Co-operative Navigation in Multi-AUV Systems

Prajjwal Dutta
Arizona State University
(Student)

Vivek Sunil Kulkarni
Arizona State University
(Student)

Niyati Abhijeet Vaidya
Arizona State University
(Student)

***Abstract*— The main objective of the project is to showcase the multi robot behavior in the underwater environment where there is very limited sensor input. An Autonomous Underwater Vehicle (AUV) and a simulated environment were designed, where a "light house" moves, providing 2D positions to an object tracking model. One robot mimics the AUV's movements, while others follow it when within their LiDAR range. In this model one of the four robots has the same positions as our AUV robot and the other robots try to follow the robot, if it appears within their defined lidar range. AUV environmental elements were left out for simplicity's sake.**

As for the Hardware, we could successfully implement the model of the AUV in the form of a blimp which can maintain a certain level and move autonomously. The further implementation of the project is still in process and can be counted on in the future scope of this project.

Keywords— Multi-robot systems, Underwater Navigation, LiDAR based tracking, Flocking Algorithm

I. Literature Review

A. *Multi-Agent Systems and Flocking*

The concept of flocking in multi-agent systems, which focuses on how robots or agents might cooperate as a group, was first proposed by Tanner, Jadbabaie, and Pappas [1][2]. Their study emphasizes how crucial it is to use fixed or dynamic communication networks for stable coordination. These concepts aid in the creation of algorithms that maintain the group's motion while preventing collisions, which is directly related to controlling several AUVs.

By improving control techniques, Beaver and Malikopoulos [3] investigated methods to increase flocking's effectiveness and energy efficiency. Their knowledge can be applied to the development of systems in which each robot operates effectively while also enhancing the overall effectiveness of the group. These ideas are especially crucial while navigating underwater obstacles including poor communication and shifting water conditions.

B. *Obstacle Avoidance and Co-ordination*

In order to avoid collisions, Arrichiello et al. [100] experimented with dispersed multi-robot systems that used laser range finders. Their use of behavior-based control techniques and extended Kalman filters for localization emphasizes how crucial distributed control and strong sensor integration are for navigating dynamic environments. These observations aid in the development of systems that can manage underwater barriers while preserving team communication.

C. *Modelling and Simulation*

In their study of AUV propulsion systems modeling, Vega et al. [4] addressed the dynamics of drag, lift, and thrust. These investigations offer a framework for recreating realistic underwater conditions when combined with Jun et al.'s [5] investigation of hydrodynamic forces. Simulation tools, such those offered by MathWorks [8][9], are crucial for confirming theoretical models and improving control algorithms prior to their practical use.

We apply these discoveries in this study to develop a decentralized multi-AUV system that effectively manages flocking and obstacle avoidance in submerged settings. Using knowledge from optimization, propulsion modeling, and flocking theory, we tackle important issues like scalable coordination, dynamic hydrodynamics, and restricted communication. We hope that our testing and simulation will help enhance autonomous underwater robotics.

II. Mathematical Model

The multi-robot behavior being studied focuses on goal-oriented motion, integrating object detection, tracking, following, and obstacle avoidance. This coordinated behavior allows the robots to move on their own and complete specific tasks, even in challenging underwater environments. Robots are made to cooperate with one another, utilizing their unique skills to increase group productivity. Prajjwal and Vivek contributed significantly to this aspect of the project.

In our hypothetical scenario, we simulate an underwater environment devoid of GPS signals and with minimal visual communication due to the extreme darkness at oceanic depths. The objective of this scenario is to observe and monitor the environment. In this context, robots rely solely on sensor data and inter-robot communication for coordination. They must navigate obstacles, maintain group cohesion, and collaboratively achieve the tasks of data collection and transmission.

A. *Assumptions and Contraints of the Model*

1) *Assumptions of the model:*

a) The lidar’s maximum range is 10 meters.

b) Three objects are the most that can be detected at once using this lidar.

c) The field of view is kept as 90 degrees (-45 to 45).

d) The radius of the robot is considered to be 1 for the multi robot flocking simulation.

2) *Constraints of the model:*

a) The system uses lidar data for communication, while the AUV uses its own IMU data for self-localization maximum range is 10 meters.

b) The AUV simulation includes multiple thrust parameters influenced by environmental forces, which also apply to object tracking and avoidance simulations. However, these forces are currently applied only to the lighthouse vehicle, which the other vehicles follow. While the following vehicles would theoretically experience the same forces, for simplicity, the environmental variables are limited to the lighthouse. This approach can be extended to other vehicles in future iterations to enhance system robustness.

TABLE I. SENSOR PARAMETERS

Sr. No.	Parameters and their values	
	Parameter	***Values***
1.	sensorOffset	[0,0]
2.	scanAngles	[-pi/4, 0, pi/4]
3.	maxRange	5
4.	robotIdx	1
5.	robotRadii	-1

c) In the real scenario, the environment would be unbounded, as would the AUV simulation. However, for simulation purposes, a bounded map has been used to implement obstacle avoidance, tracking, and the flocking algorithm. This map is designed as a maze, with walls acting as obstacles.

B. Variable and Parameters of Simulations

1) AUV Simulation: Considering blimp as our AUV, we have disigned the model used in our simulations.

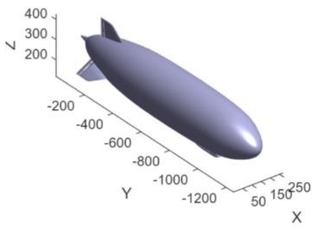


Fig 1 Designed Blimp model in MATLAB

a) Reference Parameter: A reference continuous signal for defining waypoints for generating the AUV movement trajectory.

b) Translation Controller: The translation controller is responsible for managing the movement of the AUV by controlling its position and velocity. A Proportional-Integral-Derivative (PID) control system, which guarantees precise and steady navigation, makes this possible.

c) Thrust Parameter: The thrust parameter plays a crucial role in determining the propulsion force and moments required for the AUV's controlled movement. These calculations are based on the inputs provided by the translation controller, which uses PID control to regulate position and velocity.

d)Environmental Forces and Moments: Hydrodynamic and hydrostatic forces are calculated using mathematical models that account for drag and lift forces. These parameters simulate the interaction of the AUV with the surrounding water, enabling realistic modeling of resistance, buoyancy, and lift effects in the environment ensuring accurate representation of underwater dynamics.

e) The total force and moment on the AUV are calculated by combining thruster-generated propulsion forces with hydrodynamic forces from water interaction. This integration ensures accurate modeling of the AUV's motion, accounting for environmental effects and control inputs, enabling effective maneuvering and operation in complex underwater environments.

2) Object Tracking and Avoidance: The fundamental parameters for this model remain consistent with those outlined in the previous point, ensuring a cohesive and accurate representation of the AUV's dynamics.

AUV Simulation: Considering blimp as our AUV, we have *designed* the model used in our simulations. The lidar angles and the ranges where real-time values are implemented within the logic are the sensing parameters.

3) Multi robot Flocking: Multi-robot flocking involves coordinating a group of robots to move together in a cohesive and organized manner, often mimicking natural flocking behaviors seen in birds or fish. Key parameters for implementing multi-robot flocking include:

a) Number of Robots: This refers to the total number of robots engaged in flocking behavior. The flocking algorithm's complexity is affected by the group's size. Stronger collision prevention and communication systems are needed in larger groups.

b) Robot Identification Number: Each robot is assigned a unique identification number (ID) to ensure it can be distinctly recognized within the group. This ID has several uses, including facilitating accurate tracking and role assignment throughout operations, both of which are critical for task coordination.

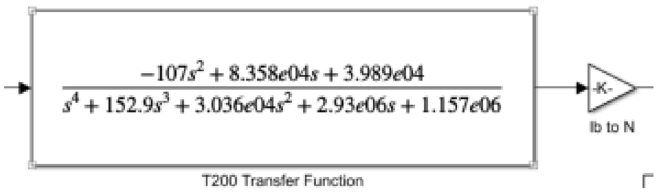
c) Number of teams: It is used to describe the separation of the robot group into smaller groups for distributed operation or task specialization.

The model's assumptions and constraints reflect deep-sea exploration challenges and current technology. LiDAR and IMU compensate for GPS and vision absence, with constraints like a 10-meter range and 90-degree view ensuring efficiency. Thrust parameters and hydrodynamic forces support accurate modeling, while a bounded map and simplified forces aid scalable testing.

C. Mathematical Model

1) AUV Model:

a) Thrust Transfer Function: The input-output relationship of the T200 and T100 thrusters is modeled by calculating their transfer functions using MATLAB. These functions help with accurate force control by illustrating how thrust reacts to control signals.



$\frac{-107s^2 + 8.358e04s + 3.989e04}{s^4 + 152.9s^3 + 3.036e04s^2 + 2.93e06s + 1.157e06}$
-K-
lb to N
T200 Transfer Function

Fig 2 Calculated Transfer Function

b) Thrust Transformation: Using MATLAB, the transfer functions for T200 and T100 thrusters are calculated, modeling their input-output relationship. These functions represent how thrust responds to control signals, aiding precise force control.

c) 6 DOF Block: Based on the dynamics of underwater motion, a six degrees of freedom (6 DOF) block transforms the changed forces into the move parameters of the vehicle, including linear and angular velocities.

d) Output data: After post-processing the movement parameters, the model outputs Inertial Measurement Unit (IMU) data and side-slip velocities, which are essential for navigation and control.

2) Object Detection and Avoidance Model: This model incorporates the sensor's position and orientation, object detection based on range and angle, field-of-view and maximum range constraints, occupancy grid checks, and sorting/limiting of detections.

*a) Sensor Position and Orientation:*The position of sensor defined by [x, y, θ] updated as follows:

$$x_{\text{sensor}} = x + dx \cdot \cos(\theta) - dy \cdot \sin(\theta)$$
$$y_{\text{sensor}} = y + dx \cdot \sin(\theta) + dy \cdot \cos(\theta)$$
$$\theta_{\text{sensor}} = 0$$

*b) Object detection:*For each object (i),

Distance to Object, $r_i = \sqrt{(x_{oi} - x_{\text{sensor}})^2 + (y_{oi} - y_{\text{sensor}})^2}$

Angle to object, $\phi_i = \text{wrapToPi}(\text{atan2}(y_{oi} - y_{\text{sensor}}, x_{oi} - x_{\text{sensor}}) - \theta_{\text{sensor}})$

c) Sorting and limiting detections: After filtering by range and field of view, the detections are sorted by distance, and only the nearest detections are kept.

3) Grouping Mathematical Calculation : Robots are organized into teams and interact based on detected positions of other robots in their environment.

a) Robot Dynamics: Each robot's position is defined in a 2D plane (x, y) and orientation (θ). Using standard differential drive kinematics, the positions are updated over time

b) Detection: A sensor model is designed to detect other robots in its vicinity. It involves the constant values of maximum detection range and field of view.

c) Team Co-ordination: After assigning each robot to a team, the swarm controller determines the motion by detecting other robots in the same team. Each robot moves according to the average position and orientation of the team.

III. Theoretical Analysis

Theoretical analysis is fundamental to understanding and designing complex systems. It helps predict behavior, address potential challenges, and develop effective solutions before transitioning to physical implementation. By offering a structured framework for validation and optimization, it ensures our models align with real-world conditions while remaining efficient and scalable. Prajjwal and Niyati have contributed extensively to this section, providing reliable values critical for our project's system implementation.

A. Mathematical Calculation for AUV

1) Motor Transfer Function Calculation:

Electrical Equation: $V_a(s) = I_a(s)R_a + sL_aI_a(s) + E_b(s)$

Where: $E_b(s) = K_b\omega(s)$ is the back EMF voltage

2) Mechanical Equation for calculating torque of the motor is:

$$T_m(s) = Js\omega(s) + B\omega(s) + T_L(s)$$

Where:$T_m(s) = K_tI_a(s)$ is the torque generated by the motor

3) Transfer function and Thrust Controller: Based on the mechanical and electrical characteristics of the T100 and T200 motors, we used MATLAB to determine their transfer functions. These transfer functions, which take into account variables like motor torque, resistance, inductance, and back EMF, show the link between the motor's input voltage and angular velocity.

$$Tansfer\ function, \frac{\omega(s)}{V_a(s)} = \frac{K_t}{(Js + B)(R_a + sL_a) + K_bK_t}$$

We used MATLAB's auto-tuning capability to tune a PID controller after receiving the transfer function. In order to reduce the discrepancy between the reference and real position and velocity measurements, the PID block was set up. For simulation, the thrust values were normalized to a set range and converted into Pulse Width Modulation (PWM) signals using a 1D lookup table. The transfer function was then applied to these values to generate accurate outputs for simulation and system control.[4]

$$V = RI + K_MT$$

Where, V: Input voltage to motor

R : Motor Resistance

I : Motor current

K_M : Motor constant relating thrust to current

In case of steady state equations, Thrust can also be expressed as a function of angular velocity and thrust coefficients such as:

$$T = K_T\omega + K_{T0}$$

Where, K_T : Thrust coefficient

K_{T0} : Constant offset for thrust

ω: : angular velocity of propeller

4) Hydrodynamic Forces: Here's how we determined the mathematical model for drag and lift forces in your simulation setup using the given photos and the hydrodynamic force model from the relevant document.

a) Drag force: It can be calculated as by the following equation:

$$D_x = \frac{1}{2}\rho A_f C_D u^2$$

Where,ρ: Fluid density

A_f: Frontal area of the object

C_D: Drag coefficient (calculated using MATLAB function Calculate_Drag_Coeff)

u: Relative velocity of the object in the fluid

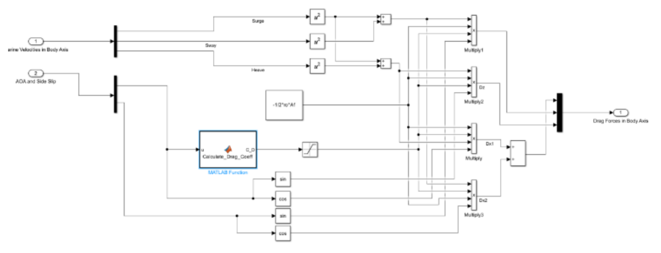


Fig 3 Drag coefficient calculation in Simulink

b) Lift force: The equation for calculating lift of the AUV is:

$$L_y = \frac{1}{2} \rho A_f C_L u^2$$

Where, C_L: Lift coefficient

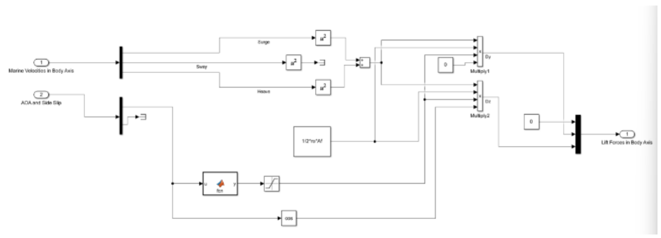


Fig 4 Lift coefficient calculation in Simulink

c) Steady State Forces and Moments: Drag and lift forces are calculated using velocity components (u2, v2, w2), which are transformed into the respective body axes through trigonometric calculations. These forces are then input into moment computation blocks, where the thrust distribution and the center of mass matrix are used to calculate the final moments, ensuring accurate force application.

$$\tau = M(q)q'' + C(q,q')q' + D(q,q') + G(q)$$

Where, $M(q)q''$: Inertia Matrix
$C(q,q')q'$: Coriolis and Centrifugal Matrix
$D(q,q')$: Hydrodynamic forces
$G(q)$: $Gravity\ and\ \ Buoyancy\ forces$

B. *Object Detector Mathematical Calculations*

*1) Sensor Position and Orientation:*The sensor has a position defined by pose = [x, y, θ], where:

- x and y represent the sensor's position in the world coordinates (in meters).
- θ represents the orientation of the sensor (in radians), relative to some reference axis.

The sensor also has an offset from the base position defined by $sensorOffset = [dx, dy]$. The sensor's position in world coordinates, taking into account the offset, is calculated as:

$$sensorPose = \begin{matrix} x_{sensor} \\ y_{sensor} \\ \theta_{sensor} \end{matrix} = \begin{matrix} x + dx \cdot cos(\theta) - dy \cdot sin(\theta) \\ y + dx \cdot sin(\theta) + dy \cdot cos(\theta) \\ \theta \end{matrix}$$

2) Object Detection Calculation: For each object, we assume the object's position is given by $objects(:,1:2)$ (i.e., $[x\_o, y\_o]$), and its label is in the third column $objects(:,3)$.

The distance r from the sensor to the object can be computed using the Euclidean distance formula:

$$r = \sqrt{}((xo - xsensor)2 + (yo - ysensor)2)$$

The angle ϕ of the object relative to the sensor's forward direction is given by:

$$\phi = wrapToPi(atan2(yo - ysensor, xo - xsensor) - \theta sensor)$$

Where $wrapToPi$ ensures that the angle ϕ is within the range $[-\pi, \pi]$.

3) Field of View and Maximum Range Filtering: The object detector has a $field\ of\ view$ (FoV) of $fieldOfView$ and a maximum range of $maxRange$. These parameters are used to filter out objects that fall outside the detection region:

- The object must be within the field of view, i.e., the absolute value of the angle ϕ must be less than or equal to half the field of view:

$$|\ \phi\ | \leq fieldOfView/2$$

- The object must also be within the maximum detection range:

$$r \leq maxRange$$

If the object satisfies these two conditions, it is considered a valid detection.

4) Sorting Detections and Limiting to Maximum Number of Detections: The detected objects are sorted by distance in ascending order, and only the nearest $maxDetections$ objects are returned:

$$detections = [sortedRanges, sortedAngles, sortedLabels]$$

If the number of valid detections exceeds the $maxDetections$ threshold, only the closest $maxDetections$ detections are retained.

C. *Grouping Mathematical Calculations*

*1) Interaction and Coordination:*The core control law in the provided code is the $swarmTeamController$ function, which defines how each robot moves relative to the other robots in its team. The controller includes*:*

a) Turning Behavior: If a robot detects no other robots within its field of view, it turns in place.

b) Avoidance and Attraction Behavior: When robots detect others from the same team, they adjust their velocities to either attract or repel from the detected robot, trying to maintain a certain range.

This can be modeled using a potential field approach, where each robot tries to avoid collisions and maintain a certain distance from others:

$$U_{attraction} = k_{attr} . range$$
$$U_{repulsion} = k_{repel} . range$$

Where:

$U_{attraction}$ *and* $U_{repulsion}$ are the potential fields representing attractive and repulsive forces.

k_{attr} *and* k_{repel} are constants that control the strength of attraction and repulsion.

$range$ is the distance between two robots.

The robot's velocity is adjusted based on these potentials to maintain the desired swarm formation.

2) *Team Coordination:*Each robot is assigned to a team, and the swarm controller calculates the motion based on the detection of other robots within the same team. The team's average position and orientation guide each robot's motion:

For each robot r_i in team j:

$$angle_{avg} = \frac{1}{N} \sum_{i=1}^{N} \theta_i$$
$$range_{avg} = \frac{1}{N} \sum_{i=1}^{N} r$$

Where:

θ_i is the relative angle of robot r_i to other robots in the same team.

r_i is the distance to robot

N is the number of robots in the team.

3) *Discrete Time Integration:The robot's positions and velocities are updated over time using a discrete-time approximation:*

$$x(t + \Delta t) = x(t) + v(t) \cdot \Delta t \cdot cos(\theta(t))$$
$$y(t + \Delta t) = y(t) + v(t) \cdot \Delta t \cdot sin(\theta(t))$$
$$\theta(t + \Delta t) = \theta(t) + \omega(t) \cdot \Delta t$$

This matches the simulation loop in the code, where the positions and velocities are updated at each time step.

IV. Simulation and Validation Of Our Model

This part of our project was undertaken by Niyati and Vivek. She was tasked with using the results of our preliminary mathematical modeling and some teamwork to create the models for the blimp to create and adjust Simulink models to validate our findings.

A. AUV Simulation:

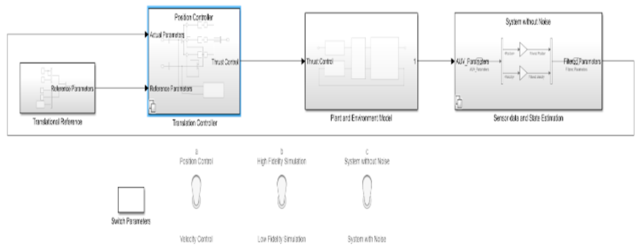


Fig 5AUV model using Simulink

The AUV weighs 10 kg with a submerged volume of 0.01 m² and is powered by five motors: two T200 for forward/backward motion, two T100 for sideways movement, and one T200 for vertical motion. The Translational Reference block generates waypoints, which are compared in real-time with actual positions. The error feeds into a PID controller for position and velocity control, producing thrust outputs after passing through a dead zone (-0.5 to +0.5). These outputs are converted to PWM signals and processed in the Propulsion Forces & Moments block using motor transfer functions to generate thrust values in x, y, and z directions, along with thrust moments for propulsion dynamics.

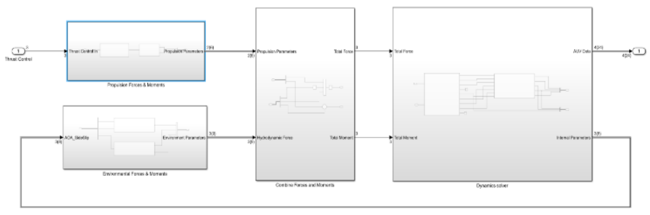

Thrust Control
Propulsion Forces & Moments
Environment Forces & Moments
Combine Forces and Moments
Dynamics solver

Fig 6 Thrust Controller

Also, the hydrostatic and hydrodynamic forces like Drag and lift forces are calculated using the above stated equations in the Theoretical analysis section, in the following blocks. [10]

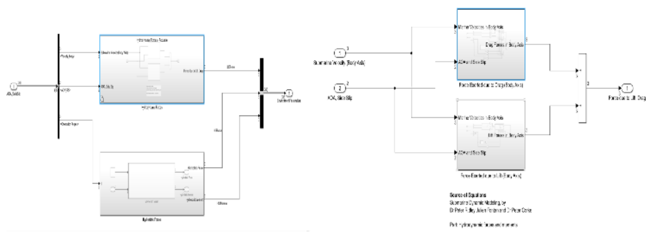


Fig 7Hydrodynamics forces applied in the simulation

These forces are added together to get the total force and moment on the AUV and finally sent to the 6 DOF body to euler angles block of simulink. This block gives outputs to the input as velocity, position, body angles, angular rates, angular acceleration, body acceleration and velocity. This is post processed to get the Angle of Attack, side slip IMU data and orientation angles.

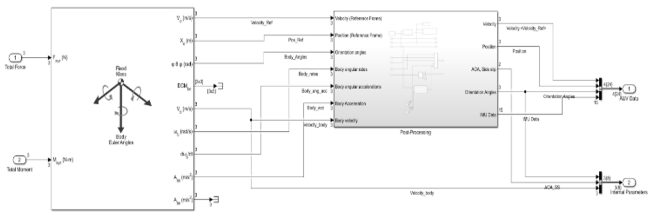


Fig 8 Getting Actual parameters

Finally, the sensor data is used for the state estimation of the system as described in III Theoretical Analysis. This filtered parameter of position and velocity is considered as the real time actual parameter of the AUV which is used in the error analysis in the beginning of the simulation. Thus, full feedback controlled AUV system is modeled.

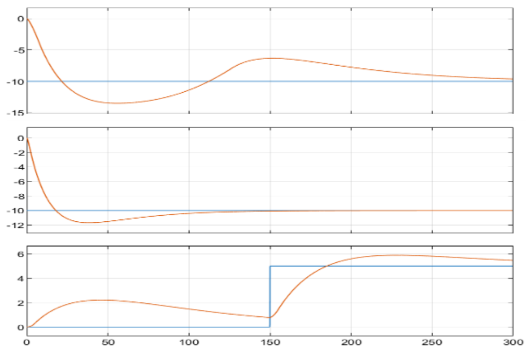


Fig 9 2D graph of x, y, z positions

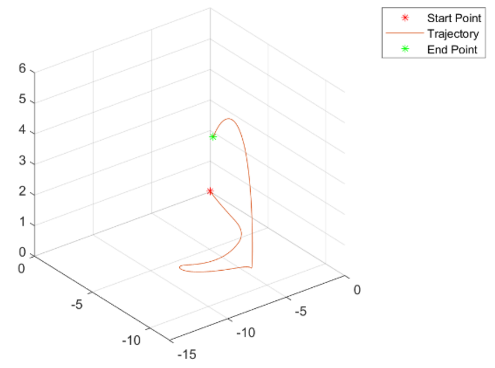


Fig 10 3D graph of x, y, z positions

B. *Object Tracking and Obstacle Avoidance Model*

For this model firstly we take the pose of the AUV from the previous model to the workspace and feed that into this model. So, the underwater environmental parameters are accounted for in this simulation as well for the lighthouse robot. For simplification purposes we are avoiding implementing these variables for all the other robots.

From the poses, instantaneous angles are extracted and compared with LiDAR angles, and LiDAR range is checked for each robot. If the average detection angle is < 0, the obstacle avoidance logic rotates the robot in place to locate the lighthouse. Once detected, the robot moves toward the lighthouse at a constant velocity. The tracking algorithm then ensures robots follow the lighthouse's center, maintaining a set distance from it and each other.

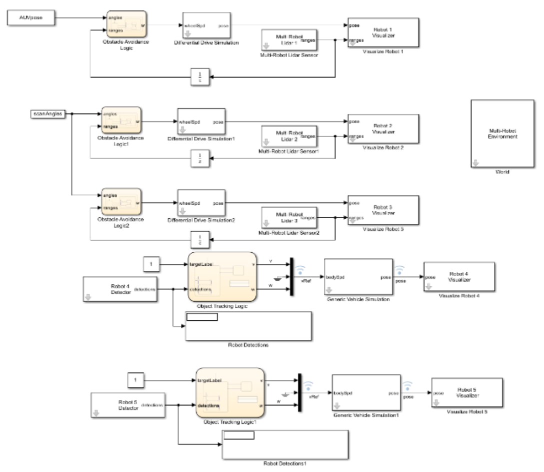


Fig 11 Obstacle tracking and avoidance Simulink Model

Fig. 12 show robots 4 and 5 optimally following robot 1, the AUV robot, while two other robots, treated as obstacles, are efficiently avoided by the lighthouse and following robots (parts b and c). This forms the foundation for the grouping simulation, further detailed in the accompanying video.

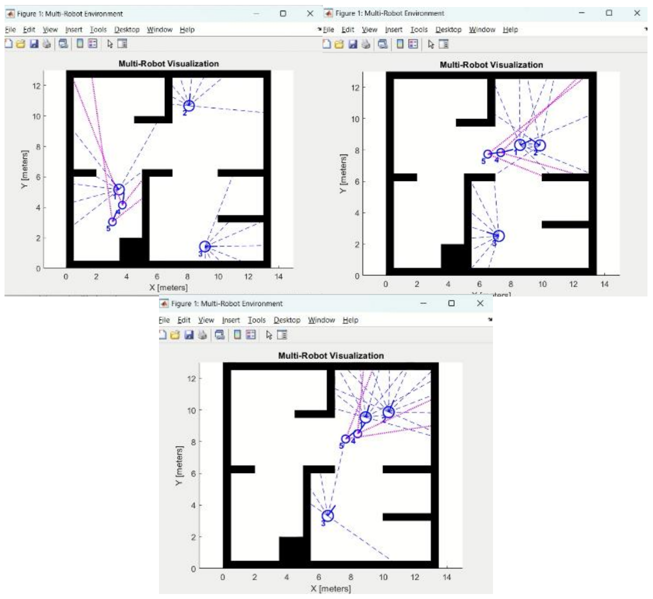


Fig 12 Output figures (a), (b) and (c)

C. *Grouping Model Simulation*

The grouping model, adapted from the widely used flocking model for multi-robot systems, is designed for underwater environments. It simulates 50 point-mass robots equipped with LiDARs, IMUs, and 90-degree fields of view. Parameters are aligned with object tracking and obstacle avoidance models to ensure consistency across simulations.

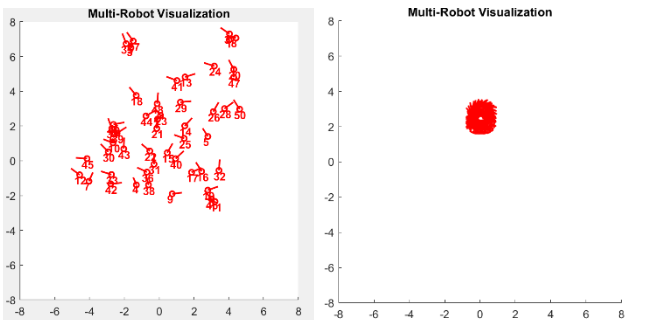


Fig 13 Grouping Model Visualization

The robots start at random locations and converge into a single group. The code can be modified to form multiple groups as needed. Fig13 a and b illustrate the initial and final grouping stages, with variations shown for different group configurations. Fig15 and Fig16 show the trajectories of all the robots in x position V/s Y position and position V/S time respectively. Note that with time when the robots form a group, the trajectories of all the robots converge and become stable as discussed in the theoretical analysis.

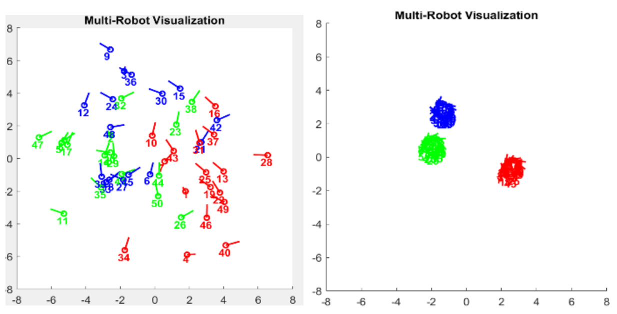


Fig 14 Initial stage (a) and final stage (b) of grouping

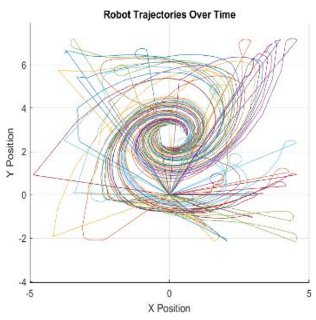


Fig 15 Trajectories (x and y position)

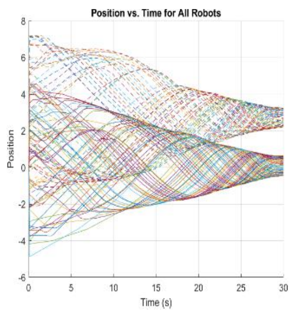


Fig 16 Trajectories (Position vs time)

V. Results

In this project we successfully implemented an AUV model, using the coordinates of the moving model in the underwater environment. The coordinates are used for modeling obstacle avoidance and robot following behavior for starting a multi-robot system. Finally, a grouping algorithm is developed on the basis of the general flocking algorithm for multirobot systems. This model shows how the robots will be able to group together in an unknown environment without visual and GPS support only through lidar and will be able to collect and transmit the required data collectively.

VI. Conclussion

After the successful implementation of the simulation, we tried to start implementing the hardware. We were only able to build and steadily fly a blimp (Because the environment of air and water are similar to an extent). We just changed the density of water with air and implemented the AUV model to the blimp. Also, instead of 5 motors we only used 3. But there are several more parts to implement in the hardware. So, the hardware implementation is not complete. Even in the simulation we have ignored environmental effects on the other robots other than the lighthouse which is significant. Also, applying the real model in the simulation is a part of the future scope of the project. So, this is just the successful starting point of the development of this project but there are many ways to improve the model and as well as implementing the hardware completely.

VII. Video Links

Simulation: https://youtu.be/16jqQIGX92Q
Hardware : https://youtube.com/shorts/o3sujP4HZNI

References

[1] H. G. Tanner, A. Jadbabaie and G. J. Pappas, "Flocking in Fixed and Switching Networks," in IEEE Transactions on Automatic Control, vol. 52, no. 5, pp. 863-868, May 2007, doi: 10.1109/TAC.2007.895948.

[2] H. G. Tanner, A. Jadbabaie and G. J. Pappas, "Stable flocking of mobile agents, part I: fixed topology," 42nd IEEE International Conference on Decision and Control (IEEE Cat. No.03CH37475), Maui, HI, USA, 2003, pp. 2010-2015 Vol.2, doi: 10.1109/CDC.2003.1272910.

[3] Logan E. Beaver, Andreas A. Malikopoulos,An overview on optimal flocking,Annual Reviews in Control, Volume 51, 2021, Pages 88-99, ISSN 1367-5788,

[4] Vega, Emanuel P., Olivier Chocron, and Mohamed Benbouzid. "AUV propulsion systems modeling analysis." International Review on Modelling and Simulations 7.5 (2014): 827-837.

[5] Jun, B.H., Shim, H.W. and Lee, P.M., 2011. Approximated generalized torques by the hydrodynamic forces acting on legs of an underwater walking robot. International Journal of Ocean System Engineering, 1(4), pp.222-229.

[6] Zhang, Qingrui, Hao Dong, and Wei Pan. "Lyapunov-based reinforcement learning for decentralized multi-agent control. Distributed Artificial Intelligence: Second International Conference, DAI 2020, Nanjing, China, October 24–27, 2020, Proceedings 2. Springer International Publishing, 2020.

[7] OpenAI, personal communication, December 9, 2024

[8] MathWorks Student Competitions Team (2024). Modeling and Simulation of an Autonomous Underwater Vehicle (https://github.com/mathworks/AUV-modeling-and-sim), GitHub. Retrieved December 9, 2024.

[9] MathWorks Student Competitions Team (2024). Mobile Robotics Simulation Toolbox (https://github.com/mathworks-robotics/mobile-robotics-simulation-toolbox), GitHub. Retrieved December 9, 2024.

F. Arrichiello, S. Chiaverini, and V. K. Mehta, "Experiments of obstacles and collision avoidance with a distributed multi-robot system," *Proceedings of the IEEE International Conference on Information and Automation*, Shenyang, China, 2012, pp. 727-732, doi: 10.1109/ICInfA.2012.6246918