Development of Multi-Agent Coordination and Control for Surface and Underwater Vehicles

A Project Report Submitted by

Arashdeep Singh

in partial fulfillment of the requirements for the award of the degree of

M.Tech

Indian Institute of Technology Jodhpur

Inter-disciplinary Research Division(IDRD)

August, 2025

Declaration

I hereby declare that the work presented in this Project Report titled **Development of Multi-Agent Coordination and Control for Surface and Underwater Vehicles** submitted to the Indian Institute of Technology Jodhpur in partial fulfilment of the requirements for the award of the degree of M.Tech, is a bonafide record of the research work carried out under the supervision of Dr.Jayant Kumar Mohanta. The contents of this Project Report in full or in parts, have not been submitted to, and will not be submitted by me to, any other Institute or University in India or abroad for the award of any degree or diploma.

Signature

Arashdeep Singh

M23IRM003

Certificate

This is to certify that the Project Report titled **Development of Multi-Agent Coordination and Control for Surface and Underwater Vehicles**, submitted by Arashdeep Singh(M23IRM003) to the Indian Institute of Technology Jodhpur for the award of the degree of M.Tech, is a bonafide record of the research work done by him under my supervision. To the best of my knowledge, the contents of this report, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Signature

Dr.Jayant Kumar Mohanta

Acknowledgements

I want to convey my sincere gratitude to my thesis supervisor, Dr. Jayant Kumar Mohanta, for his invaluable advice, constant encouragement, genuine concern, prompt assistance, and the creation of a great environment for conducting research. He has offered me pleasant and kind support throughout the project so that I may finish my research project. His belief in my abilities and constant encouragement helped me push through challenging moments and remain dedicated to my research goals.

I would like to thank Mr. Jay Khatri (Ph.D.) and Mr. Ravindra Kumar (Research Assistant) for their constant guidance, assistance, and suggestions on the project work.

Abstract

This study aims to create a framework for decision making and obstacle avoidance for marine robots capable of surface and underwater operations. Core elements of this framework are a multi-agent coordination system for task allocation and role switching and an obstacle-avoidance module. Utilizing priority-based task allocation, agents with flexible tasks may optimally handle mission-critical operations. Using dynamic leader-follower configurations, this system promotes cooperative behavior among several agents when necessary, therefore improving mission efficiency. This multi-agent agent system depends on autonomous systems. One important idea for autonomy is obstacle avoidance, particularly concave obstacle avoidance and passing through enclosures in complicated underwater settings. Our main emphasis was on solving the difficulties of concave obstacle avoidance and distributed self-task allocation.

Contents

List of Figures

List of Tables

1 Introduction

As advances in technology are growing at a faster pace, humans desire to make machines capable of working without human intervention. These are known as unmanned systems. And we want unmanned systems to work for us in almost every domain, be it medical, construction, etc. One such domain is underwater operations.

In the vast, ever-changing underwater world, relying solely on one vehicle is like trying to navigate uncharted waters by yourself. A fleet of autonomous surface vehicles (ASVs) and underwater vehicles (AUVs) cooperating is used in a multi-agent system. In addition to increasing efficiency, this coordinated approach adds resilience because, in the event that one agent experiences an issue, other agents can adjust and step in to ensure mission continuity.

Complex missions require flexibility. Underwater conditions may change rapidly, most likely due to currents, unexpected marine life, or evolving mission parameters. Dynamic task allocation enables each vehicle to reassess its environment and capabilities on a continuous basis, switching roles and priorities in real-time. This smooth distribution of tasks ensures that the most suitable agent is always playing the most critical role, optimizing performance and conserving energy.

For robots, the underwater world is full of dangers like rocks and big marine life. Avoiding these dangers is essential for robot safety and mission success. Advanced algorithms help robots to safely navigate around these dangers. Which ensures progress towards the goal.

2 Literature survey

- **Sahoo et al.** highlights developments of AUVs, mentioning the work for which they are developed, their capability, and their design. Apart from this authors discussed various under water communication modems and their capability comparison. Also, the paper briefly introduces modelling, control, and path planning for AUVS. [1]

- **T. I. Fossen** This book provides detailed modeling and control techniques for marine vehicles, including ASVs and AUVs. Fossen's work is instrumental in understanding ASV/AUV motion dynamics and control. The book discusses mainly two theories, maneuvering Theory and Seakeeping Theory, for modelling dynamics of marine vehicles. [2]

- **Rui Hu et al.** focused on hybrid aerial underwater vehicle's buoyancy control. They have used a piston-based actuator to control the buoyancy of the vehicle. Further they have employed PID controller. They have created a prototype names as Nezha III. They are controlling pitch and heave based on buoyancy. They also analyzed the effect of speed of piston movement. [3]

- **Mohd Aras et al.** discuss the hull design, propulsion system, and dynamic analysis of an underwater vehicle. They have developed the actual setup. Also they have discussed the parameters effecting performance of AUV like pressure, environmental forces. [4]

- **Aditya Natu et al.** discuss the design and development of AUV named VARUNA 2.0. They have employed a PID controller. Also they have discussed about task detection and task execution. They have emolyed two batteries one for back up and discussed operating time off the vehicle. They have also discussed Computer vision applications and challenges in underwater. [5]

- **Leong Wai Lunn et al.** focused on developing AUV that autonomously navigates for a predefined task. They have employed TNMM module with camera and depth sensor. They also discuss the design and propulsion dynamics of AUVs. They also demonstrated the prototype developed based on the target acquisition test and navigation test. [6]

- **Wang et al.** discusses task allocation for homogenous and heterogenous agents. They have divided the existing algorithms into centralized and decentralized approaches. They also introduced parallel task allocation for multiple tasks. They also show simulation of SOM, Ants, GA, PSO and Auction algorithms. They showcased layered planning to improve the hierarchical planning of tasks. [7]

- **Esther Bischoff et al.** have introduced greedy-constructive heuristic which yields feasible initial solutions independent of the problem size. For task execution, they have introduced a local improvement heuristic. They have simulated the introduced method, discussing the efficiency of the method introduced on multi-task allocation. [8]

- **Y. Ma, Y. Liu, L. Zhao, and M. Zhao,** have discussed the problems faced by multi-agent control. They have discussed consensus, formation, and flocking as major divisions of a multi-

agent system. Then they discuss problems like Connectivity, Distributed estimation, Containment control, and Coverage control. [9]

- **Z. Fang et al.** have discussed problems of multi-agent reinforcement learning in uncertain marine applications. So, they explored the use of multi-agent generative adversarial imitation learning (MAGAIL) in autonomous underwater vehicles (AUVs). They have used Unmanned Underwater Vehicle (UUV) simulator with gazebo to simulate MAGIL for formation control and obstacle avoidance. They have used decentralized training with decentralized execution framework. [10]

- **M. Liu and Y. Feng** examines group consensus in mixed-order multi-agent systems, accommodating both first- and second-order agents under fixed, directed topologies. Stability and control strategies are discussed, providing valuable insights for designing task-switching and priority-based allocation strategies in complex agent architectures [11].

- **Y. Bai et al.** Bai's research on multi-agent coverage control addresses static and dynamic obstacle avoidance using Voronoi partitions. Adaptive control mechanisms are discussed, ideal for multi-agent navigation in environments with obstacles like underwater terrains. This study supports task allocation frameworks ensuring coordinated movement and obstacle avoidance [12].

- **L. Wen, J. Yan, X. Yang, Y. Liu, and Y. Gu** present a method for trajectory planning in ASVs, focusing on efficient and safe navigation in obstacle-dense environments. The proposed framework decouples the planning process into path searching and trajectory optimization, with a specific focus on energy-efficient, collision-free routes. This study is relevant for ASV applications requiring reliable path planning under dynamic constraints and fuel efficiency considerations [13].

- **Lei Cao, Guo-Ping Liu, Da-Wei Zhang** proposes a predictive control-based leader-follower consensus approach for networked multi-agent systems. The method compensates for communication delays and ensures consensus and stability in environments with dynamic topologies. The system's effectiveness is validated via experiments using air-bearing simulators [14].

- **J. Xu et al.** apply deep reinforcement learning in multi-AUV attack-defense scenarios, enabling AUVs to develop cooperative strategies under limited perception. The proposed framework achieves effective multi-agent coordination, demonstrating the suitability of reinforcement learning for decision-making in high-stakes underwater missions [15].

- **B.Xu et al.** presents a novel approach to formation control for Autonomous Underwater Vehicles (AUVs) under dynamic switching topology. The authors propose a double-layer distributed formation prediction control scheme, which includes an upper-level adaptive distributed observer and a lower-level predictive controller. This method addresses the challenges posed by external time-varying disturbances and ensures robust formation maintenance. [16]

- **Huber et al.** worked on multiple moving obstacle avoidance specially focusing on convex and star shaped obstacles. Their work is inspired from harmonic potentials. They validated their work in

grocery store pick and place task. Their work is based on modulating the original dynamics of the system. [17]

- **Huber et al.** worked on concave obstacle avoidance by modifying original non linear dynamics of the dynamical system. The main idea is rotating the initial dynamics into a tangent space which is safe to move around obstacles. The proposed method is highly reactive and effective in dynamic environments. They demonstrated proposed approach on 7-DoF robot arm around dynamic obstacles. [18]

- **Khansari et al.** discuss about dynamical systems for real time obstacle avoidance. They discussed the problem of sudden obstacles. They discussed about Harmonic Potential function. They worked on modifying the normal component of velocity field according to obstacle which will result in tangent component making robot to slide on the boundary of the obstacle. [19]

3 Problem definition and Objective

There have been a lot of ongoing developments related to marine robotics. Many problems still need more focus and alternative approaches. One such problem is the underwater search operations. Let us understand this problem in more detail. There is a surface vehicle and several underwater vehicles. As in underwater search operations, we will try to explore every possible and feasible direction. Let there be eight directions, such as NW, NE, SW, etc. Now, the question arises: How can it be effectively done? One answer is a multi-agent system with dynamic task allocation. This task allocation will help agents decide what best they can perform. Thus, we can properly utilize our resources. So the solution involves autonomus vehciles in multi- agent frame work with dynamic task assignment. For the effective working of marine vehicles for surface and underwater operations, It is necessary to have strong control systems, decision-making algorithms, communication, and hardware development. Moreover, for multi-agent systems, it becomes essential to develop a framework that will decide how they will behave in dynamic situations. Like agents are in a formation with leader follower mechanism. And suddenly, the leader gets destroyed by the enemy. Now, to complete the assigned task, there should be a new leader.

A major challenge in achieving autonomy is obstacle avoidance, especially concave obstacles and enclosures. As in the underwater scenario, there is much more possibility of gaps between objects like caves. Currently, available methods try to avoid them entirely by ignoring the gap. However, for better maneuverability and efficient operations, we want the capability of moving in and out of enclosures.

Apart from that, we have primarily underactuated underwater vehicles as we utilize fins for orientation control to get more working time. However, it introduces the problem of controlling underactuated robots.

Objectives Based on above problem statement these are my primarliy objectives :-

- Modelling of AUV,ASV:- Finding accurate dynamic models to represent both the autonomous underwater vehicle (AUV) and the autonomous surface vehicle (ASV) is essential for simulating real-world behavior, enabling effective control.
- Control of Underactuated Vehicles:- Creating control systems to track specified trajectories, prevent obstacles, and preserve stability.
- Obstacle Avoidance and Navigation: AUVs with autonomous obstacle detection and Avoidance ensures safe operation and efficient navigation. This capability is critical for uninterrupted multi-agent operations in dynamic underwater environments.
- Task Switching and Priority-Based Allocation: I wish to allow every AUV to assume duties depending on its capacity, therefore enabling dynamic task switching and reallocation depending on mission priorities. This will guarantee that high-priority tasks get attention from the most competent agents and maximize resource use.
- Simulation: Building and integrating consistent simulation soultion that will support to justify the suggested solutions. The main task will be to create a simulation environment which will have capability to simulate underwater and surface vehicles at same time.

4 Methodology

4.1 System Modeling and Control

Develop and understand mathematical models for Autonomous Surface Vehicles (ASVs) and Autonomous Underwater Vehicles (AUVs). These models incorporate hydrodynamic forces, moments, and external disturbances. Control surfaces like rudders and thrusters are modeled to reflect underactuated control and environmental interactions.

4.2 Control for AUVs

Implement control strategies for underactuated agents based on their dynamics. I will be using cascaded control with PID as outer loop and to remove non-linearity will use Feedback linearization in the inner loop.

4.3 Multi-Agent Coordination and Control Strategy

Task Allocation and Switching

- **Priority-Based Task Assignment**: Develop a task assignment framework where tasks are allocated based on each agent's capabilities and mission priority requirements, ensuring high-priority tasks are given to the most capable agents.
- **Dynamic Task Switching**: Implement a mechanism for dynamic task-switching, allowing agents to reassign tasks based on real-time conditions, such as obstacle detection, to maintain formation and mission success in uncertain environments.
- **Event-Triggered Control (ETC)**: To reduce communication load, employ ETC to trigger updates only upon significant state deviations, minimizing data exchange while maintaining system performance under limited bandwidth conditions.

4.4 Concave Obstacle Avoidance

To make the robot move through caves and archways, we need to work on concave obstacle avoidance. To make it possible, we will use the concept of rotating dynamics away from the obstacle in a pseudo-tangent direction.

5 Mathematical Modelling and Design

5.1 Modelling of Underwater Vehicle

Following the Manuvering Theory Given by T. I. Fossen [2], I have worked on kinematics and dynamics of surface and underwater vehicles.

Kinematics of Six DOF Underwater Vehicle:- It describes how the vehicle's position (x, y, z) and orientation (roll ϕ, pitch θ, yaw φ) change over time based on its control inputs like linear velocities (u, v, w) and angular velocities (p, q, r). Essentially, it captures the complex motion dynamics of the vehicle as it navigates underwater.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\varphi} \end{bmatrix} = \begin{bmatrix} c(\varphi)c(\theta) & -s(\phi)c(\phi) + c(\varphi)s(\theta)s(\phi) & s(\varphi)s(\phi) + c(\varphi)c(\phi)s(\theta) & 0 & 0 & 0 \\ s(\varphi)c(\theta) & c(\varphi)c(\phi) + s(\phi)(\theta)(\varphi) & -c(\varphi)s(\phi) + s(\theta)s(\varphi)c(\phi) & 0 & 0 & 0 \\ -s(\theta) & c(\theta)s(\varphi) & c(\theta)c(\phi) & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & s(\phi)t(\theta) & c(\phi)t(\theta) \\ 0 & 0 & 0 & 0 & c(\phi) & -s(\phi) \\ 0 & 0 & 0 & 0 & s(\phi)se(\theta) & c(\phi)se(\theta) \end{bmatrix} \begin{bmatrix} u \\ v \\ w \\ p \\ q \\ r \end{bmatrix}$$

4-DOF Dynamic Model of an Autonomous Underwater Vehicle (AUV)

The equations of motion for an AUV in surge, sway, heave, and yaw (4-DOF) are given by:

Surge Equation:

$$m_{11}\dot{u} - m_{22}wr + m_{33}vq + d_{11}u = \tau_u \tag{1}$$

Sway Equation:

$$m_{22}\dot{v} + m_{11}wr - m_{33}uq + d_{22}v = \tau_v \tag{2}$$

Heave Equation:

$$m_{33}\dot{w} + m_{11}vq - m_{22}ur + d_{33}w = \tau_w - (W - B) \tag{3}$$

Yaw Equation:

$$m_{44}\dot{r} + m_{22}wu - m_{11}vq + d_{44}r = \tau_r \tag{4}$$

5.2 Modelling of Surface Vehicle

For the small boat with 3 degrees of freedom (surge, sway, yaw), the kinematics and kinetics can be expressed in both equation form and vector representation under these assumptions:-

- The motion in roll, pitch, and heave is ignored. This means that we ignore the dynamics associated with the motion in heave, roll, and pitch, i.e., (z = 0), (p = 0), and (q = 0).
- The vessel has homogeneous mass distribution and an xz-plane of symmetry so that ($I_{xy} = I_{yz} = 0$).
- The center of gravity (CG) and the center of buoyancy (CB) are located vertically on the z-axis.

The kinematic equations in the body-fixed frame can be written as:

$$\dot{x} = u\cos\varphi - v\sin\varphi \tag{5}$$

$$\dot{y} = u\sin\varphi + v\cos\varphi \tag{6}$$

$$\dot{\varphi} = r \tag{7}$$

Vector Representation of Kinematics

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\varphi} \end{bmatrix} = \begin{bmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ r \end{bmatrix}$$

The **dynamic equations** in the body-fixed frame can be written as:

$$m(\dot{u} - vr) = T - X_r \tag{8}$$

$$m(\dot{v} + ur) = Y_\delta - Y_r \tag{9}$$

$$I_z\dot{r} = N_\delta - D_r \tag{10}$$

5.3 Multi-Agent Coordination and Control

I will discuss the intuitive foundation and mathematical formulation of a priority-aware, event-triggered multi-agent coordination and control algorithm. I detail how each agent's internal awareness , capability evaluation, task prioritization, and event-driven re-planning combine to yield adaptive team behavior.

Agent Awareness Each agent i maintains a local state vector:

$$\mathbf{s}_i = \begin{bmatrix} E_i \\ \mathbf{p}_i \\ \mathbf{v}_i \\ \vdots \end{bmatrix} \tag{11}$$

where:

- E_i is the remaining battery energy of agent i.
- $\mathbf{p}_i \in \mathbb{R}^n$ is the current position.
- $\mathbf{v}_i \in \mathbb{R}^n$ is the current velocity.
- Additional entries may represent sensor health, payload, or other relevant states.

This awareness enables each agent to estimate its suitability for any task.

Capability Evaluation For a task j characterized by a requirement vector $\mathbf{r}_j$, define the capability function:

$$C_i^j = f(\mathbf{s}_i, \mathbf{r}_j) = w_E \, g_E(E_i, r_j^E) + w_d \, g_d\big(\|\mathbf{p}_i - \mathbf{p}_j^*\|\big) + \cdots, \tag{12}$$

where:

- r_j^E is the energy requirement for task j.
- $\mathbf{p}_j^*$ is the task location.
- $g_E(E_i, r_j^E) = \min\{1, E_i/r_j^E\}$ normalizes energy availability.
- $g_d(\|\mathbf{p}_i - \mathbf{p}_j^*\|)$ normalizes distance.
- $w_E, w_d \geq 0$ are weights with $w_E + w_d + \cdots = 1$.

Thus, $C_i^j \in [0, 1]$ quantifies how capable agent i is of performing task j.

Task Priority and Utility Each task j has a priority $\rho_j > 0$. The utility of assigning agent i to task j is defined as:

$$U_i^j = \rho_j \, C_i^j. \tag{13}$$

The global objective is to maximize total utility:

$$\max_{\mathcal{A}} \sum_{(i,j)\in\mathcal{A}} U_i^j, \tag{14}$$

subject to assignment constraints (e.g., each agent assigned to at most one task).

Event-Triggered Decision Making Agents monitor for events $\mathcal{E}$ (e.g., $E_i < E_{\min}$, obstacle detection, task completion). Define an event indicator:

$$\delta_e(t) = \begin{cases} 1, & \text{if an event in } \mathcal{E} \text{ occurs at time } t, \\ 0, & \text{otherwise.} \end{cases} \tag{15}$$

When $\delta_e(t) = 1$, the system re-computes C_i^j, U_i^j, and resolves the assignment.

Algorithm Flow

1. **Initialization:** Agents broadcast initial states $\mathbf{s}_i(0)$.
2. **Capability Computation:** Compute C_i^j for all agent-task pairs.
3. **Utility Matrix:** Form $U \in \mathbb{R}^{N \times M}$ with entries U_i^j.
4. **Assignment:**
$$\mathbf{X}^* = \arg \max_{\mathbf{X} \in \{0,1\}^{N \times M}} \text{trace}(\mathbf{X}^T U),$$
5. **Execution:** Agents execute assigned tasks.
6. **Monitoring & Events:** If $\delta_e(t) = 1$, return to Step 2.

Algorithm 1 Multi-Agent Task Allocation with Capability Sharing, Task Switching, and Formation Control

Initialize task allocator $T.A$
Initialize agents $A_1, A_2, \ldots, A_n$
Initialize mission parameters and environment
$T \leftarrow$ AllocateTasks($T.A$) {Task allocator floats tasks(may with priority) to agents}
for each agent A_i **do**
  Set initial position P_i and state S_i
  A_i.tasks $\leftarrow T$ {Receive task list, with or without priority numbers}
end for
while mission is active **do**
  for each agent A_i **do**
    $S_i \leftarrow$ GetState(A_i)
    $C_i \leftarrow$ AssessCapability(A_i)
    A_i.share $\leftarrow (A_i.\text{name}, C_i)$ {Agents share their capabilities}
    A_i.info $\leftarrow$ GatherInfo($A_{1,2,\ldots,n}$) {Each agent gathers others' capabilities}
    A_i.assigned_tasks $\leftarrow$ SelectBestTasks(A_i.info, A_i.tasks) {Assign tasks based on priority and capability score; if tasks have no priority, assign based only on capability score}
  end for
  Agents confirm selected tasks with each other, then break off communication
  if EventTriggered() **then**
    {Trigger event for reassessment} **for** each agent A_i **do**
    $C_i \leftarrow$ ReassessCapability(A_i)
    A_i.share $\leftarrow (A_i.\text{name}, C_i)$ {Share updated capabilities}
    A_i.info $\leftarrow$ GatherInfo($A_{1,2,\ldots,n}$)
    A_i.assigned_tasks $\leftarrow$ ReallocateTasks(A_i.info) {Re-allocate tasks}
    end for
    if HighPriorityAgentDestroyed() **then**
      {Task switching triggered if high-priority task agent is lost} Reshuffle tasks to reassign high-priority tasks to the most capable available agents
    end if
  end if
  $G \leftarrow$ FormGroups($A_1, A_2, \ldots, A_n$)
  for each group G_k in G **do**
    $L_k \leftarrow$ SelectLeader(G_k) {Most capable agent becomes leader}
    $L_k \leftarrow$ CoordinateActions(G_k) {Leader coordinates with followers}
  end for
  MonitorProgress()
end while
CompleteMission() =0

Algorithm 2 Capability Assessment and Task Assignment

Input: *agents* (list of agents with attributes), *task_details* (list of tasks with details)
Output: *task_assignments* (list of task-to-agent assignments)
Define information shared by each agent:
Each agent shares:

- Position (x, y, z coordinates)
- Battery level and battery capacity (normalized)
- Communication and movement capabilities
- Vehicle type (e.g., 'surface' or 'underwater')

Define a strategy for assessing capability:
Function $CalculateEnergyPriority(agents, task\_details)$
Initialize $energy\_priority \leftarrow []$
for each *agent* in *agents* **do**
  Initialize $agent\_energies \leftarrow []$
  for each *task* in *task_details* **do**
    $distance \leftarrow$ EuclideanDistance$(agent.position, task.position)$
    Append *distance* to *agent_energies*
  end for
  Normalize *agent_energies* to range [0,1] and append to *energy_priority*
end for
return *energy_priority*
Function $AssessCommunication(agent)$
$communication\_score \leftarrow w_1 \times agent.range + w_2 \times agent.bandwidth + w_3 \times (1 - agent.latency) + w_4 \times agent.reliability$
return *communication_score*
Function $AssessMovement(agent)$
$movement\_score \leftarrow u_1 \times agent.speed + u_2 \times agent.maneuverability + u_3 \times agent.terrain\_adaptability + u_4 \times agent.endurance$
return *movement_score*
Function $AssessCapability(agent)$
$normalized\_battery \leftarrow agent.battery\_level / agent.battery\_capacity$
$communication\_score \leftarrow AssessCommunication(agent)$
$movement\_score \leftarrow AssessMovement(agent)$
$overall\_score \leftarrow v_1 \times normalized\_battery + v_2 \times communication\_score + v_3 \times movement\_score$
return *overall_score*
Function $GenerateTaskAssignments(agents, task\_details)$
$energy\_priority \leftarrow CalculateEnergyPriority(agents, task\_details)$
Initialize $task\_assignments \leftarrow []$
for each $i, task$ in *task_details* **do**
  $task\_type \leftarrow task.type$
  $eligible\_agents \leftarrow []$
  for each $agent\_index, agent$ in *agents* **do**
    if agent.vehicle_type == 'surface' or (agent.vehicle_type == 'underwater' and task_type == 'underwater') **then**
      $capability\_score \leftarrow AssessCapability(agent) + energy\_priority[agent\_index][i]$
      Append $(agent, capability\_score)$ to *eligible_agents*
    end if
  end for
  Sort *eligible_agents* by *capability_score* in descending order
  if two or more agents have the same highest *capability_score* **then**
    Break ties based on:

    - Higher battery level
    - Closer Euclidean distance to task position

  end if
  Assign the agent with the highest (or tie-broken) score to *task*
  Append $(task.id, selected\_agent.id)$ to *task_assignments*

5.4 Obstacle Avoidance

Obstacle avoidance is a key component for autonomy of robots. It is responsible for several key components like Safety, Efficiency, Lifespan, and autonomous navigation. Let us discuss some keywords that are often used in research on obstacle avoidance:-

Real-Time Obstacle Avoidance: Real-time obstacle avoidance refers to a robot's ability to detect and navigate around obstacles as they appear, without pre-planning a path. This capability is crucial for robots operating in dynamic environments where obstacles can suddenly appear or move. The robot uses sensors to continuously monitor its surroundings and adjust its path instantaneously to avoid collisions.

Dynamical Systems in Obstacle Avoidance: A dynamical system in obstacle avoidance is a mathematical framework used to describe the robot's motion. It involves equations that govern the robot's behavior over time, allowing it to react to changes in its environment smoothly and efficiently. This approach ensures that the robot can avoid obstacles by continuously adjusting its trajectory based on the current state of the system.

Cartesian Space: This refers to the robot's movement in a three-dimensional space, defined by x, y, and z coordinates. Obstacle avoidance in Cartesian space involves planning the robot's path directly in this coordinate system, ensuring it doesn't collide with obstacles in its environment.

Joint Space: This involves controlling the robot's individual joints to achieve the desired end-effector position. Obstacle avoidance in joint space means planning the movements of each joint to avoid collisions, which can be more complex but allows for more precise control of the robot's posture.

Concave Obstacle: These are obstacles with inward curves or indentations. Imagine a U-shaped barrier; the inner part of the U is a concave region. Navigating around concave obstacles can be challenging because the robot might get trapped in the concave area if not properly planned.

Convex Obstacle: These are obstacles with outward curves, like a sphere or a cube. Convex obstacles are generally easier for robots to navigate around because there are no indentations where the robot can get stuck. The robot can simply move around the outer boundary of the obstacle.

There has been intense research on obstacle avoidance using artificial potential field and learning based methods. In this thesis we have mainly focused on avoiding obstacles using dynamical systems as it offers several benefits like:-

- **Robustness**: Dynamical systems can handle unexpected changes and disturbances in the environment without needing to replan the entire path.
- **Smooth Motion**: The continuous nature of dynamical systems ensures smooth and natural movements.
- **Real-Time Adaptation**: They allow for real-time adjustments, making the robot highly responsive to dynamic environments

Artifical Potential Method(APF) is widely used for obstacle avoidance. But I have done comparison with modern dynamic based methods I find it lacking the real time obstacle avoidance capbility. Below is the table of comparison of APF with dynamics based methods.

Table 5.1: Comparison of Different Obstacle avoidance Methods

Feature	APF	DMM	ROAM
Smooth Navigation	Struggles with local minima (robot can get stuck)	Ensures continuous motion by modifying velocity	Ensures smooth motion while respecting robot dynamics
Handling Dynamic Obstacles	Difficult without modifications	Can adapt to moving obstacles in real-time	Can handle dynamic obstacles better than DMM
Mathematical Stability	Can cause oscillations near obstacles	Provides stability using modulation	Ensures stability with pseudo-tangents
Implementation Complexity	Simple and easy to implement	Requires computing modulation matrices	More complex due to directional space calculations

5.4.1 Dynamic Modulation Method for Obstacle Avoidance

The **Dynamic Modulation Method** (DMM) is a reactive obstacle avoidance technique that deforms a robot's desired velocity field to generate safe and smooth navigation trajectories. The core idea is to modulate the motion vector so that it lies within the tangent space of nearby obstacles while preserving convergence to the goal.

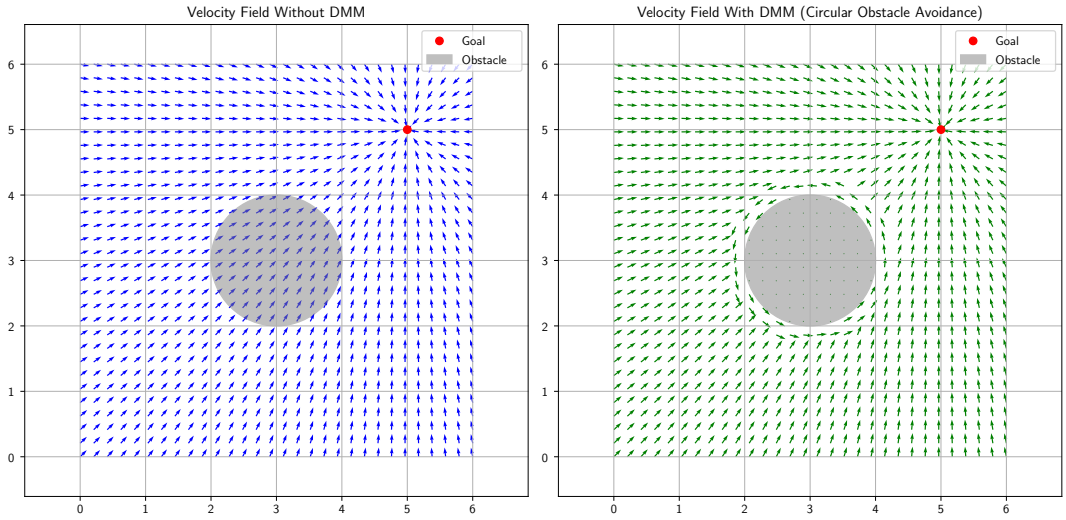

Figure 5.1: Velocity modulation via obstacle surface projection

Assumptions:

- The environment contains N smooth, compact, and convex obstacles, each represented by a differentiable implicit function $\phi_i(\mathbf{x})$, such that the obstacle boundary is $\mathcal{B}_i = \{\mathbf{x} \mid \phi_i(\mathbf{x}) = 0\}$.
- The robot has access to a desired velocity field $\mathbf{V}_d(\mathbf{x})$, e.g., generated from a goal attractor or planner.
- Obstacle avoidance occurs in a local frame assuming no global replanning is required.

Velocity Field Representation:

$$\dot{\mathbf{x}} = \mathbf{V}_d(\mathbf{x}) \in \mathbb{R}^n \tag{16}$$

In the presence of obstacles, the velocity is modulated using a local deformation:

$$\dot{\mathbf{x}}_{\text{mod}} = \mathbf{M}(\mathbf{x})\mathbf{V}_d(\mathbf{x}) \tag{17}$$

where $\mathbf{M}(\mathbf{x}) \in \mathbb{R}^{n\times n}$ is a symmetric, positive semi-definite modulation matrix constructed based on obstacle geometry.

Obstacle Surface Representation: Each obstacle i is defined by a level set:

$$\phi_i(\mathbf{x}) = 0, \quad \text{with} \quad \nabla\phi_i(\mathbf{x}) = \mathbf{n}_i(\mathbf{x}) \tag{18}$$

where $\mathbf{n}_i(\mathbf{x})$ is the normalized surface normal at point $\mathbf{x}$.

Single Obstacle Modulation Matrix: For a single obstacle, the modulation matrix is:

$$\mathbf{M}_i(\mathbf{x}) = \mathbf{I} - (1 - \lambda_i(\mathbf{x}))\mathbf{n}_i(\mathbf{x})\mathbf{n}_i(\mathbf{x})^T \tag{19}$$

where the scalar $\lambda_i(\mathbf{x}) \in [0, 1]$ controls the influence of the modulation:

$$\lambda_i(\mathbf{x}) = \text{clip}\left(\left(\frac{d_i(\mathbf{x})}{d_{s,i}}\right)^2, 0, 1\right) \tag{20}$$

with $d_i(\mathbf{x}) = \|\phi_i(\mathbf{x})\|$ and $d_{s,i}$ the safe distance threshold.

Multiple Obstacle Modulation (Sequential Composition): In environments with multiple obstacles, a sequential application of modulation is used:

$$\dot{\mathbf{x}}_{\text{mod}} = \mathbf{M}_N(\mathbf{x}) \cdots \mathbf{M}_2(\mathbf{x})\mathbf{M}_1(\mathbf{x})\mathbf{V}_d(\mathbf{x}) \tag{21}$$

Alternatively, the modulation matrices can be combined via weighted blending:

$$\mathbf{M}_{\text{blend}}(\mathbf{x}) = \sum_{i=1}^{N} w_i(\mathbf{x})\mathbf{M}_i(\mathbf{x}) \tag{22}$$

with normalized weights:

$$w_i(\mathbf{x}) = \frac{\gamma_i(\mathbf{x})^{-1}}{\sum_{j=1}^{N} \gamma_j(\mathbf{x})^{-1}}, \quad \gamma_i(\mathbf{x}) = d_i(\mathbf{x}) + \epsilon \tag{23}$$

where ϵ is a small constant to prevent division by zero.

Geometric Intuition:

- When $\mathbf{x}$ is far from all obstacles, $\lambda_i(\mathbf{x}) \to 0$ and $\mathbf{M}(\mathbf{x}) \approx \mathbf{I}$.

- When $\mathbf{x}$ is near an obstacle, $\lambda_i(\mathbf{x}) \to 1$, and the component of $\mathbf{V}_d$ in the direction of $\mathbf{n}_i$ is suppressed.
- The tangent motion dominates, leading to a natural flow around the obstacle surface.

Stability Guarantee: As shown in [19], when the original system $\dot{\mathbf{x}} = \mathbf{V}_d(\mathbf{x})$ is globally asymptotically stable at a point $\mathbf{x}^*$ and obstacles are properly modeled, the modulated system retains asymptotic convergence toward $\mathbf{x}^*$ while avoiding all obstacles.

Below is the algorithm for multiple obstacle avoidance using DMM. Here Lidar is used for obstacle detection and PID is used as controller.

Algorithm 3 DMM-Based Obstacle Avoidance with Multiple Obstacles

Current velocity $\mathbf{v}$, LiDAR data $L = \{l_i\}$, orientation quaternion q
Modified velocity $\mathbf{v}'$
function DMM_MultiObstacle_Avoidance($\mathbf{v}, L, q$)
  Extract yaw angle θ from quaternion q
  Initialize empty list of obstacle normals $\mathcal{N} = []$, distances $\mathcal{D} = []$
  for each LiDAR point $l_i \in L$ **do**
    Extract range r_i and angle α_i
    Transform to local coordinates: $\mathbf{p}_i = (r_i \cos\alpha_i, r_i \sin\alpha_i)$
    if $r_i < d_{\text{safe}}$ **then**
      Compute normal: $\mathbf{n}_i = \frac{\mathbf{p}_i}{\|\mathbf{p}_i\|}$
      Append $\mathbf{n}_i$ to $\mathcal{N}$ and r_i to $\mathcal{D}$
    end if
  end for
  if $\mathcal{N}$ is not empty **then**
    Initialize blended modulation matrix: $\mathbf{M}_{\text{blend}} \leftarrow \mathbf{0}_{2\times 2}$
    for $i = 1$ to $|\mathcal{N}|$ **do**
      Let $\mathbf{n}_i = \mathcal{N}[i]$, $d_i = \mathcal{D}[i]$
      Compute weight: $w_i = \frac{1}{d_i + \epsilon}$
      Compute $\lambda_i = \text{clip}\left(\left(\frac{d_i}{d_{\text{safe}}}\right)^2, 0, 1\right)$
      Compute modulation matrix: $\mathbf{M}_i = \mathbf{I} - (1 - \lambda_i)\mathbf{n}_i\mathbf{n}_i^\top$
      $\mathbf{M}_{\text{blend}} \leftarrow \mathbf{M}_{\text{blend}} + w_i \cdot \mathbf{M}_i$
    end for
    Normalize weights: $\mathbf{M}_{\text{blend}} \leftarrow \frac{\mathbf{M}_{\text{blend}}}{\sum w_i}$
    Compute modulated velocity: $\mathbf{v}' \leftarrow \mathbf{M}_{\text{blend}} \cdot \mathbf{v}$
  else
    $\mathbf{v}' \leftarrow \mathbf{v}$ {No nearby obstacles}
  end if
  Publish or return modified velocity $\mathbf{v}'$
end function

5.4.2 Rotational Obstacle Avoidance Method

The **Rotational Obstacle Avoidance Method (ROAM)** is a real-time, dynamic motion planning algorithm designed for navigating autonomous agents through complex environments populated with static or dynamic obstacles. Unlike traditional trajectory planners that compute a full path ahead of time, ROAM adopts a local, velocity-based strategy where the robot continuously adapts its motion based on perceived surroundings.

At the heart of ROAM lies the concept of velocity field modulation. A baseline converging dynamical system towards the goal is defined as:

$$\dot{\xi} = f(\xi), \tag{24}$$

where $\xi \in \mathbb{R}^n$ denotes the agent's position and $f(\xi)$ represents the nominal velocity field guiding the agent to the goal. In the presence of obstacles, ROAM rotates velocity field into a safe direction. This rotation ensures smooth avoidance by projecting the desired motion onto the local tangent space of obstacles while scaling the motion near boundaries to enforce safety.

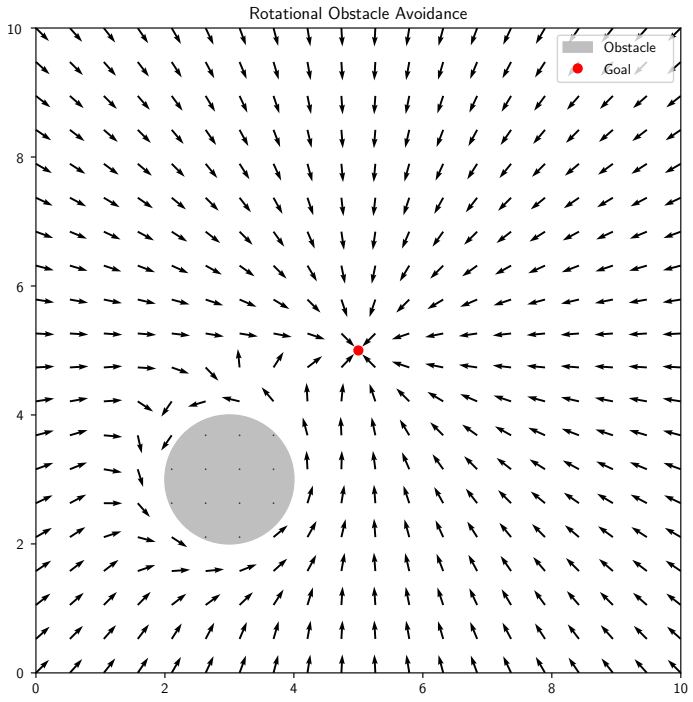

Figure 5.2: Rotational Obstacle Avoidance

Comparison with Dynamic Modulation Method (DMM)

Although ROAM and the *Dynamic Modulation Method (DMM)* share similar foundations — both modulate a nominal velocity field to achieve obstacle avoidance — they differ in design philosophy and

computational structure:

- **Geometric vs. Rotational Modulation Focus:** DMM constructs the modulation matrix by explicitly using geometric features such as the surface normal and tangent basis derived from implicit obstacle representations. ROAM, in contrast, focuses on rotating the convergence direction to avoid obstacles while preserving goal-directed behavior. Rather than modulating repulsion geometrically, ROAM modifies the vector field orientation smoothly within the tangent space.

- **Reactivity vs. Rotation-Informed Flow:** While DMM emphasizes modulation via orthogonal decomposition and scaling of eigenvalues (often leading to sharp redirection), ROAM applies smooth rotational transformations to blend between the convergence vector and an obstacle-aware pseudo-tangent direction, leading to more fluid trajectories. ROAM is not merely reactive but structurally modifies the dynamics based on obstacle proximity and orientation.

- **Modulation Design Philosophy:** DMM relies on constructing a modulation matrix using a basis formed by the normal and tangent directions, enforcing strict repulsion via eigenvalue scaling. ROAM, on the other hand, introduces a rotation angle between the convergence direction and the modulated vector, ensuring motion stays within the tangent plane and avoids obstacle penetration while allowing for graceful redirection.

- **Multi-Obstacle Handling:** DMM often includes explicit combination rules for handling multiple obstacles, usually by modulating the final velocity based on distance-weighted superposition. ROAM, due to its rotational nature, can blend rotations associated with each obstacle, resulting in a composite flow field that still respects the tangent constraints and avoids abrupt discontinuities.

In summary, DMM is highly geometric and modulation-centric, ideal for applications requiring formal guarantees and strong obstacle avoidance behavior. ROAM leverages rotational dynamics to maintain smooth and goal-consistent navigation, often producing more natural paths, especially in cluttered environments.

Below is the description of mathematical formulation for modifying the original convergence dynamics $c(\xi)$ to ensure that it remains in a safe, constrained direction. By blending the reference direction $r(\xi)$ and convergence dynamics $c(\xi)$, and solving for the correct weighting factor b, the method ensures that the final motion follows a proper geometric structure, avoiding unwanted behavior such as direct collisions with obstacles.

5.4.2.1 Directional Space and Directional Weighted Mean

The purpose of the directional space transformation is to project a global direction vector $\mathbf{v}_i \in \mathbb{R}^N$ back into the tangent space orthogonal to a reference direction $\mathbf{b} \in \mathbb{R}^N$, typically representing the obstacle normal. This transformation reduces the dimensionality of the problem from N to $N-1$, providing a compact representation of the directional deviation from the reference.

Let:

- $\mathrm{v}_i \in \mathbb{R}^N$ be a unit velocity vector representing a direction.

- $\mathbf{b} \in \mathbb{R}^N$ be the reference direction, typically the normalized obstacle normal.
- $B \in \mathbb{R}^{N \times N}$ be a basis matrix where:
  - The first column is $\mathbf{b}$
  - The remaining $N-1$ columns form an orthonormal basis of the null space orthogonal to $\mathbf{b}$
- $_i = \mathrm{B}^\top \mathbf{v}_i$ be the transformed velocity vector in the new basis.

The transformation into direction space K :

$$K = \{\kappa_i \in \mathbb{R}^{N-1} where \|\kappa_i\| < \pi\} \tag{25}$$

The magnitude of $\hat{v}_i$ is equal to angle between the original vector and reference vector.
The Transformation of initial vector in direction space is : [20]

$$\kappa_i(b) = k(\mathbf{v}_i, b) = \begin{cases} \arccos\left(\hat{v}_{i[1]}\right) \dfrac{\hat{v}_{i[2:]}}{\|\hat{v}_{i[2:]}\|}, & \text{if } \hat{v}_{i[1]} \neq 1 \\ 0, & \text{if } \hat{v}_{i[1]} = 1 \end{cases} \tag{26}$$

Where: $\hat{v}_{i[1]}$ is the angle between the reference vector and transformed vector $\hat{v}_i$
$\arccos(\hat{v}_i[1]))$ gives the angle θ between the original vector and the reference direction.
$\frac{\hat{v}_i[2:]}{\|\hat{v}_i[2:]\|}$) is a normalized component of $\hat{v}_i$ excluding the first element.
If $\hat{v}_i[1] = 1$:
The function directly returns 0, meaning no deviation from the reference direction.

The function $\kappa_i(b) = k(v_i, b)$ computes a directional weighting factor based on the transformed vector $\hat{v}_i$.

The Mean Directional Space Vector for N Obstacles is given by:

$$\bar{\kappa} = \sum_{i=1}^{N_v} w_i \kappa_i \tag{27}$$

where w_i depends upon the distance from obstacle.

5.4.2.2 Inverse Directional Space

Directional Space Transformation: As defined in Section 5.4.2.1, the directional space vector $\kappa_i \in \mathbb{R}^{N-1}$ is given by:

$$\kappa_i(\mathbf{b}) = k(\mathbf{v}_i, \mathbf{b}) = \begin{cases} \arccos\left(\hat{v}_{i[1]}\right) \dfrac{\hat{\mathbf{v}}_{i[2:]}}{\|\hat{\mathbf{v}}_{i[2:]}\|}, & \text{if } \hat{v}_{i[1]} \neq 1 \\ 0, & \text{if } \hat{v}_{i[1]} = 1 \end{cases} \tag{28}$$

Inverse Directional Space: To recover the global direction vector from directional space $\kappa_i \in \mathbb{R}^{N-1}$, we perform the inverse transformation:

1. Compute the angle

$$\theta = \|\kappa_i\| \tag{29}$$

2. Define the first element of $\hat{\mathbf{v}}_i \in \mathbb{R}^N$ as:

$$\hat{v}_{i[1]} = \cos(\theta) \tag{30}$$

3. The remaining components are given by:

$$\hat{\mathbf{v}}_{i[2:]} = \sin(\theta) \cdot \frac{\kappa_i}{\|\kappa_i\|} \tag{31}$$

4. Form the full vector:

$$\hat{\mathbf{v}}_i = \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \cdot \frac{\kappa_i}{\|\kappa_i\|} \end{bmatrix} \tag{32}$$

5. Apply the inverse transformation using the basis matrix B:

$$\mathbf{v}_i = B\hat{\mathbf{v}}_i \tag{33}$$

Basis Matrix Construction: The matrix $B \in \mathbb{R}^{N \times N}$ is constructed such that:

- The first column is $\mathbf{b}$
- The second column is a unit vector $\mathbf{u}$ orthogonal to $\mathbf{b}$
- The third column is $\mathbf{b} \times \mathbf{u}$, ensuring a right-handed orthonormal basis

These columns are normalized to ensure orthogonality and unit length.

Normalization: Finally, the recovered vector $\mathbf{v}_i$ is normalized to ensure:

$$\|\mathbf{v}_i\| = 1$$

This guarantees that the direction vector remains on the unit sphere in $\mathbb{R}^N$.

5.4.2.3 Pseudo Tangent Direction

The desired pseudo tangent direction $e(\xi)$ is obtained by rotating the convergence dynamics $C(\xi)$ away from the reference direction. [18]

Desired Modification The desired pseudo-tangent vector $e(\xi)$ is obtained by rotating the original convergence dynamics $c(\xi)$ away from the reference direction $r(\xi)$ until it lies in the tangent plane. [20] $r(\xi)$ is a reference direction, possibly aligned with Obstacle's surface. Instead of moving directly along $c(\xi)$, we modify it so that it conforms to constraints in the angular space. [18]

Understanding Angular Space and Tangent Hyper-Sphere The angular space is $k(n, \cdot)$ In this space, any tangent vector is always at a fixed distance $R_e = \frac{\pi}{2}$ from the normal direction. The set of all tangent directions forms a hyper-sphere in the direction space. [20]

The key point here is that the transformation of $c(\xi)$ must be such that it respects this geometric structure. Thus, to rotate $c(\xi)$ away from $r(\xi)$, we intersect the line connecting $r(\xi)$ and $c(\xi)$ with a circle

of radius $R_e \in [\pi/2, \pi]$. [18] [20]

Constraints for Obtaining $e(\xi)$

The pseudo-tangent vector $e(\xi)$ is computed using the following conditions:

$$\text{if } \|k(-n, c)\| \geq R_e, \text{ then } e = c \qquad [18] \tag{34}$$

This means that if the magnitude of $k(-n, c)$ is already large enough (i.e., at least R_e), then no adjustment is needed, and we set $e = c$. Otherwise, we need to blend $c(\xi)$ and $r(\xi)$ using a weighted sum:

$$k(-n, e) = (1 - b)k(-n, r) + bk(-n, c) \qquad [18] \tag{35}$$

where:

- $k(-n, e)$ is the transformed direction after applying the correction.
- $(1 - b)$ and b are weights that mix the reference direction r and the original convergence direction c.
- $b \in \mathbb{R}_{>0}$ (a positive real number).

The final condition ensures that the transformation stays within the required geometric limits:

$$\|k(-n, e)\| = R_e \tag{36}$$

This means the final adjusted vector $e(\xi)$ must lie at exactly the required distance R_e from the normal.

Solving for b Since the equation 36 involves a quadratic dependence on b, the parameter b can be found by solving a quadratic equation.

Interpretation and Impact on Behavior The modification ensures that $e(\xi)$:

- **Lies in the tangent plane**: Instead of following $c(\xi)$ directly, it respects the geometric constraints.
- **Avoids obstacles**: The correction ensures that the direction of motion is never directly toward an obstacle.
- **Maintains smooth motion**: The transition between $c(\xi)$ and $e(\xi)$ is continuous, preventing abrupt changes in motion.
- **Handles special cases properly**:
  - If $c(\xi)$ and $r(\xi)$ coincide, the correction is straight forward.
  - If they are significantly different, the method finds an appropriate blend.

5.4.2.4 Rotation Towards Tangent Direction

Rotate initial dynamics $f(\xi)$ towards the peudo tangent direction. In this step, the initial dynamics $f(\xi)$

is rotated towards the pseudo tangent $e(\xi)$. This rotation operation is performed in the direction space with respect to the convergence dynamics $c(\xi)$: [20]

$$\dot{\xi} = k^{-1}\Big(c,\ \big(1 - \lambda(\xi)\big)\, k\big(c, f\big)\ +\ \lambda(\xi)\, k\big(c, e\big)\Big) \qquad [18] \tag{37}$$

where k^{-1} is conversion from lower dimensional space to higher dimensional space i.e from directional space to original vector space.

$$\lambda(\xi) = \left(\frac{1}{\Gamma(\xi)}\right)^{q} \qquad [18] \tag{38}$$

$$R^r = \min\Big(R^e - \|k(-n,\, r)\|,\ \tfrac{\pi}{2}\Big) \qquad [18] \tag{39}$$

$$q = \max\Big(1,\ \tfrac{R^r}{\Delta k(c)}\Big)^{s} \qquad [18] \tag{40}$$

$$\Delta k(c) = \big\|\, k(-n,\, r)\ -\ k(-n,\, c)\ \| \qquad [18] \tag{41}$$

where the rotation weight $\lambda(\xi)$ is determined based on the inverse of the distance $\Gamma(\xi)$. This ensures the rotation has a decreasing influence as the distance increases, allowing for a smooth transition in the avoidance behavior. Additionally, the smoothing factor q is introduced, which gradually decreases to zero when the convergence dynamics vanish around the robot. This effectively cancels the rotation avoidance effect in regions where the tangent $e(\xi)$ is not defined.

5.4.2.5 Evaluation of Speed

The directional space mapping $k(\cdot)$ and its inverse mapping $k^{-1}(\cdot)$ operate on unit vectors in the original space. As a result, the algorithm modifies the *direction* of the initial dynamics, rather than its magnitude. [20] To incorporate magnitude control in a decoupled manner, we introduce an additional component using $h(\xi) : \mathbb{R}^n \to [0, 1]$. [18] Formally,

$$\|\dot{\xi}\|\ =\ h(\xi)\, \|f(\xi)\| \qquad [18] \tag{42}$$

The stretching factor $h(\xi)$ is designed to slow down the motion when pointing towards an obstacle, and the effect decreases with increasing distance from the obstacle. We define:

$$h(\xi)\ =\ \min\left(1,\ \frac{\big\|\Delta k(c)\big\|}{R^r},\ \left(1 - \frac{1}{\Gamma(\xi)}\right)^{2}\right) \qquad [18]\ [20] \tag{43}$$

5.4.2.6 Weighted Global Velocity Computation from Multiple Obstacles

When multiple obstacles influence a robot's motion, each obstacle contributes to the global velocity vector based on its proximity. Closer obstacles are more important and thus are given higher weights. This is achieved by assigning each obstacle a weight inversely proportional to its distance. The final global velocity is then computed as the weighted average of the individual velocity contributions from each

obstacle.

Let:

- N be the total number of obstacles,
- d_i be the distance to the i^{th} obstacle,
- $w_i = \frac{1}{d_i}$ be the weight assigned to the i^{th} obstacle,
- $\vec{v}_i = \begin{bmatrix} v_{ix} \\ v_{iy} \end{bmatrix}$ be the velocity contribution from the i^{th} obstacle,
- $\vec{v}_{\text{global}}$ be the resulting global velocity vector.

Then the global velocity is given by:

$$\vec{v}_{\text{global}} = \begin{cases} \dfrac{\sum_{i=1}^{N} w_i \vec{v}_i}{\sum_{i=1}^{N} w_i}, & \text{if } \sum_{i=1}^{N} w_i \neq 0 \\ \displaystyle\sum_{i=1} w_i \vec{v}_i, & \text{otherwise} \end{cases} \tag{44}$$

$$\vec{v}_i = \begin{bmatrix} v_{ix} \\ v_{iy} \end{bmatrix}, \quad \vec{v}_{\text{global}} = \begin{bmatrix} v_x \\ v_y \end{bmatrix} \tag{45}$$

This formulation ensures that obstacles closer to the robot exert a stronger influence on the robot's final movement direction.

Convergence vector $c(\xi)$ For this study $c(\xi)$ is chosen to be

$$c(\xi) = \xi - \xi^a \tag{46}$$

where ξ^a is the attractor point.

Reference vector r

$$r = p_{\text{obs}} - p_{\text{current}} = \begin{bmatrix} x_{\text{obs}} \\ y_{\text{obs}} \\ z_{\text{obs}} \end{bmatrix} - \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x_{\text{obs}} - x \\ y_{\text{obs}} - y \\ z_{\text{obs}} - z \end{bmatrix} \tag{47}$$

where

p_{obs} represents position of obstacle

p_{current} represents current position of robot

Algorithm 4 Rotational Obstacle Avoidance (ROAM) Algorithm

Current position $P = (x, y)$ and orientation θ, PID velocity $v = (v_x, v_y)$ and convergence vector $c = (-e_x, -e_y, 0)$, LiDAR data L, waypoints, safe distance d_{safe}, ROAM params R_e, γ

Modified control commands (thrust and fin position)

function ROAM_AVOIDANCE(P, v, L)

$O \leftarrow$ extract_obstacles(L)

Compute convergence vector c from desired minus current position

if $O \neq \emptyset$ **then**

$n \leftarrow -\big($normal of first obstacle$\big)$

for each obstacle $o_i \in O$ **do**

$r \leftarrow \big(o_i.\text{center}_x - x,\ o_i.\text{center}_y - y,\ 0\big)$

$e \leftarrow$ COMPUTEPSEUDOTANGENT(n, c, r, R_e)

$k \leftarrow$ ROTATECONVERGENCE$\big($direction_space(c, c), direction_space(c, e), γ, R_e, direction_space(n, r), direction_space(n, c), $c\big)$

$s \leftarrow$ SPEED_SCALING$\big(R_e$, γ, direction_space(n, r), direction_space$(n, c)\big)$

$v' \leftarrow s \cdot k$

Weight v' by $1/\text{dist}(o_i, P)$ and accumulate into overall v'

end for

if $w_{\text{sum}} \neq 0$ **then**

$v' \leftarrow \left(\frac{x_{\text{sum}}}{w_{\text{sum}}}, \frac{y_{\text{sum}}}{w_{\text{sum}}}\right)$

else

$v' \leftarrow (x_{\text{sum}}, y_{\text{sum}})$

end if

$\theta' \leftarrow \arctan\big(v'_y / v'_x\big)$

Clamp and normalize v' and compute thrust/fin from (v', θ')

Publish control commands

else

Thrust $\leftarrow v_x$, fin $\leftarrow \arctan(v_y / v_x)$

Publish PID control commands

end if

end function

6 Implementation

6.1 Control of Underactuated Robot

The AUV used to verify earlier introduced mathematical concepts is underactuated. With one reversible propeller in x direction. For yaw and pitch control fins have been used.

Cascaded control offers a structured way to manage these challenges by decomposing the control problem into two nested loops. The *outer loop* regulates the vehicle's position by comparing the desired position p_d to the measured position p, and uses a PID regulator to generate a commanded velocity v_d. The *inner loop* then regulates velocity: it compares v_d to the measured velocity v and uses a second PID to produce a desired acceleration a_d. This separation simplifies tuning—each loop handles one integration order of the vehicle's kinematics—and improves robustness by limiting the bandwidth of each controller.

To handle the AUV's nonlinear dynamics in the inner loop, we employ *feedback linearization*. Given the standard rigid-body model

$$M(q)\,\ddot{q} + C(q,\dot{q})\,\dot{q} + D(q,\dot{q})\,\dot{q} + g(q) \;=\; \tau \tag{48}$$

where q is the configuration vector, M the inertia matrix, C the Coriolis/centripetal terms, D the hydrodynamic damping, and g the restoring forces, we choose

$$\tau \;=\; M(q)\,a_d \;+\; C(q,\dot{q})\,\dot{q} \;+\; D(q,\dot{q})\,\dot{q} \;+\; g(q) \tag{49}$$

This exact inversion cancels all nonlinearities, yielding the linear input–output relationship $\ddot{q} = a_d$. The inner PID then effectively controls a double-integrator, vastly simplifying stability and performance analysis.

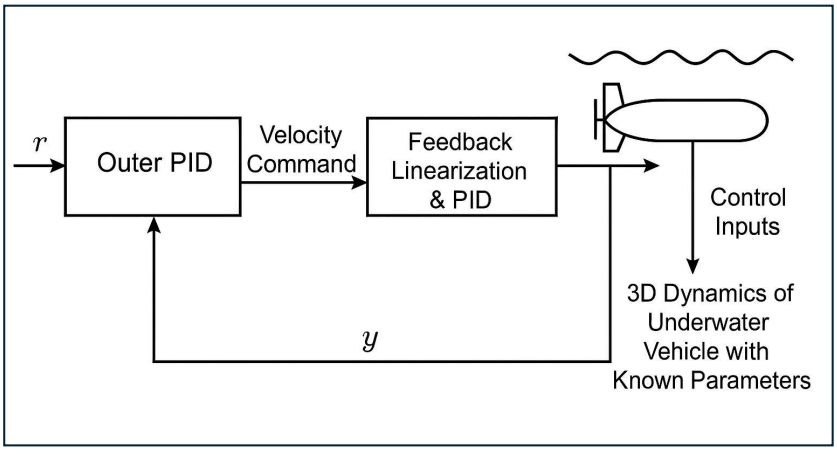

Figure 6.1: AUV State Control Flow-chart

In short, the cascaded PID structure decouples position and velocity regulation, while feedback linearization cancels the AUV's nonlinear terms. The result is a controller that is both easy to tune and capable of high-performance trajectory tracking, even in the presence of strong hydrodynamic effects.

6.2 Controller Design

AUV Parameters Physical Properties

- Mass of AUV: $m = 148.3571$ kg
- Neutral buoyancy assumed: $W = B$
- Propeller diameter: $d_p = 0.2$ m

State Variables

- u: Linear velocity in surge (x-axis)
- q: Angular velocity in pitch (y-axis)
- r: Angular velocity in yaw (z-axis)
- θ: Pitch angle
- ψ: Yaw angle

Table 6.1: AUV Parameters

AUV Parameters	
Inertia	
I_{xx}	$3.0000\ \text{kg} \cdot \text{m}^2$
I_{yy}	$41.980233\ \text{kg} \cdot \text{m}^2$
I_{zz}	$41.980233\ \text{kg} \cdot \text{m}^2$
Added Mass	
$X_{\dot{u}}$	-4.876161
$M_{\dot{q}}$	-33.46
$N_{\dot{r}}$	-33.46
Hydrodynamic Drag	
$X_{\lvert u\rvert u}$	-6.2282
$M_{\lvert q\rvert q}$	-632.698957
$N_{\lvert r\rvert r}$	-632.698957

Dynamic Equations (3 DOF)

Let the total surge, pitch, and yaw inertias be:

$$m_u = m - X_{\dot{u}} = 153.2333\ \text{kg} \tag{50}$$

$$I_{yy}^* = I_{yy} - M_{\dot{q}} = 75.440233\ \text{kg} \cdot \text{m}^2 \tag{51}$$

$$I_{zz}^* = I_{zz} - N_{\dot{r}} = 75.440233\ \text{kg} \cdot \text{m}^2 \tag{52}$$

Surge

$$m_u \dot{u} = -m_u u r + X_{|u|u}|u|u + T \tag{53}$$

Pitch

$$I_{yy}^* \dot{q} = M = M_{\text{hydro}}(\delta_p) \tag{54}$$

Yaw

$$I_{zz}^{*}\dot{r} = N = N_{\text{hydro}}(\delta_y) \tag{55}$$

Under neutral buoyancy and symmetric design, restoring moments are negligible.

Feedback Linearization

Surge (u)

$$f_u(u,r) = -m_u ur + X_{|u|u}|u|u \tag{56}$$

$$\dot{u} = \frac{1}{m_u}\left(T + f_u(u,r)\right) \tag{57}$$

Define virtual control v_u:

$$\dot{u} = v_u \quad \Rightarrow \quad T = m_u v_u - f_u(u,r)$$

Pitch (q)

$$f_q(q) = M_{|q|q}|q|q \tag{58}$$

$$\dot{q} = \frac{1}{I_{yy}^{*}}\left(\tau_{pitch} + f_q(q)\right) \tag{59}$$

Define virtual control v_q:

$$\dot{q} = v_q \quad \Rightarrow \quad \tau_{pitch} = I_{yy}^{*} v_q - f_q(q) \tag{60}$$

Yaw (r)

$$f_r(u,r) = m_u ur + N_{|r|r}|r|r \tag{61}$$

$$\dot{r} = \frac{1}{I_{zz}^{*}}\left(\tau_{yaw} - f_r(u,r)\right) \tag{62}$$

Define virtual control v_r:

$$\dot{r} = v_r \quad \Rightarrow \quad \tau_{yaw} = I_{zz}^{*} v_r + f_r(u,r) \tag{63}$$

PID Controller on Virtual Inputs

Surge PID

$$v_u = K_{p_u}(u_d - u) + K_{i_u}\int (u_d - u)\,dt + K_{d_u}(\dot{u}_d - \dot{u}) \tag{64}$$

Pitch PID

$$v_q = K_{p_q}(q_d - q) + K_{i_q}\int (q_d - q)\,dt + K_{d_q}(\dot{q}_d - \dot{q}) \tag{65}$$

Yaw PID

$$v_r = K_{p_r}(r_d - r) + K_{i_r}\int (r_d - r)\,dt + K_{d_r}(\dot{r}_d - \dot{r}) \tag{66}$$

Final Control Inputs

- Propeller Thrust:

$$T = m_u v_u + m_u ur - X_{|u|u}|u|u \tag{67}$$

- Pitch Moment via fins:

$$\tau_{pitch} = I_{yy}^{*} v_q - M_{|q|q}|q|q \tag{68}$$

- Yaw Moment via fins:

$$\tau_{yaw} = I_{zz}^{*} v_r - m_u u r - N_{|r|r}|r|r \tag{69}$$

Notes

- Restoring moments are neglected due to neutral buoyancy and symmetric configuration.
- PID gains (K_p, K_i, K_d) should be tuned based on desired tracking performance.

6.3 Obstacle Detection and Processing

I have used a lidar sensor to detect obstacles. I will explain the lidar data processing through modules namely `extract_obstacles` and `compute_obstacle_properties`. Where `extract_obstacles` is responsible for checking the presence of obstacle and distance from agent. While second function is responsible to calculate obstacle properties like normal vector, obstacle center.

6.3.1 Extracting Obstacles

The module `extract_obstacles` processes LiDAR data to identify and extract obstacles. The main steps are as follows:

1. Initialize an empty list of obstacles and a temporary list for the current obstacle.
2. Iterate through the LiDAR data points.
3. For each data point, check if the difference in distance from the previous point exceeds a given threshold.
4. If the threshold is exceeded and the current obstacle has more than two points, compute its properties and add it to the list of obstacles.
5. Clear the current obstacle list and start a new one.
6. Convert each data point to an angle-distance pair and add it to the current obstacle list.
7. After processing all data points, perform a final check on the last obstacle and add it to the list if it meets the criteria.

6.3.2 Computing Obstacle Properties

The module `compute_obstacle_properties` calculates the properties of an obstacle based on its points. The main steps are:

1. Compute the mean distance of the obstacle points.

2. Fit a line to the points using the least squares method to determine the obstacle's orientation.

3. Calculate the center of the obstacle in Cartesian coordinates.

4. Compute the normal vector to the obstacle's surface.

6.3.3 AUV state control With Obstacle Avoidance

As we are working on modulating dynamics based on obstacles. So here is the how control flow looks like with obstacle avoidance:-

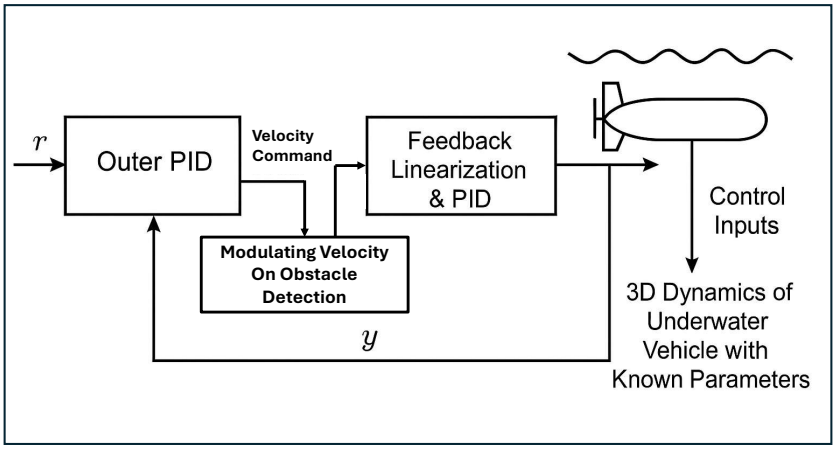

Figure 6.2: Control Flow with Obstacle Avoidance

6.4 Environment Setup

In this section, we describe the simulation environment that integrates Gazebo with ROS2 to implement a multi-agent framework with obstacle avoidance for marine robots.

Justification for Using Gazebo: Gazebo is chosen as the simulation platform due to its realistic physics, robust ROS2 integration, and extensive sensor support. Its ability to simulate complex environmental effects—such as waves, currents, and drag forces—provides a highly realistic testing ground for evaluating the performance of our autonomous systems.

Simulation World: The simulation world is designed to replicate an underwater and surface environment with carefully modeled physical properties.This includes factors like fluid dynamics and environmental disturbances to mimic real-world underwater conditions.6.1

Table 6.2: Simulation Parameters

Parameter	Value/Description
Water Density	1025 kg/m^3
Drag Coefficient	0.8
Gravity	9.81 m/s^2
Gazebo Version	Gazebo Harmonic
ROS2 Version	ROS 2 Humble

Robot Model: The robot models, such as AUVs and ASVs, are constructed using SDF/URDF. These models incorporate detailed representations of physical dimensions, mass properties, and sensor placements, ensuring accurate simulations of robot dynamics and interactions. [21]

Physics Engine: For physics simulation, we employ the Open Dynamics Engine (ODE), which provides realistic dynamics and collision handling essential for underwater simulations.

Sensor Models: The environment includes multiple sensor models (IMU, camera, LiDAR,GPS and Compass) integrated via ROS2 topics. This setup ensures comprehensive perception data for tasks such as navigation and obstacle avoidance.

Visualization Tools: The simulation setup is complemented by several visualization tools:

- ROS2 Node Graph using `rqt_graph`
- Gazebo World Screenshots
- RViz visualization of the robot and sensor data
- Velocity/Trajectory Graphs for performance analysis

ROS2 Communication: The simulation leverages ROS2 [22] for inter-node communication. The table below outlines the key nodes and their associated topics for task allocation:

Table 6.3: ROS2 Communication Overview

Node	Subscribed Topics	Published Topics
`Decision_Making`	/Task_Details /global/allocated_tasks /global/aggregated_task_allocations	/global/allocated_tasks /global/task_allocation_discussion
`Aggregator_node`	/global/task_allocation_discussion	/global/aggregated_task_allocations
`Task_publisher`	—	/Task_Details

7 Results and Analysis

7.1 Multi-Agent Task Allocation

There are 8 underwater vehicles and 1 surface vehicle. Surface vehicle is the vehcile that receives information from outer world and accordingly it floats tasks. There may be less number of tasks than the agents or it's opposite. The agents needs to discuss between themselves and decide which one task will be done by which agents. The below figure shows the flow of information between agents:-

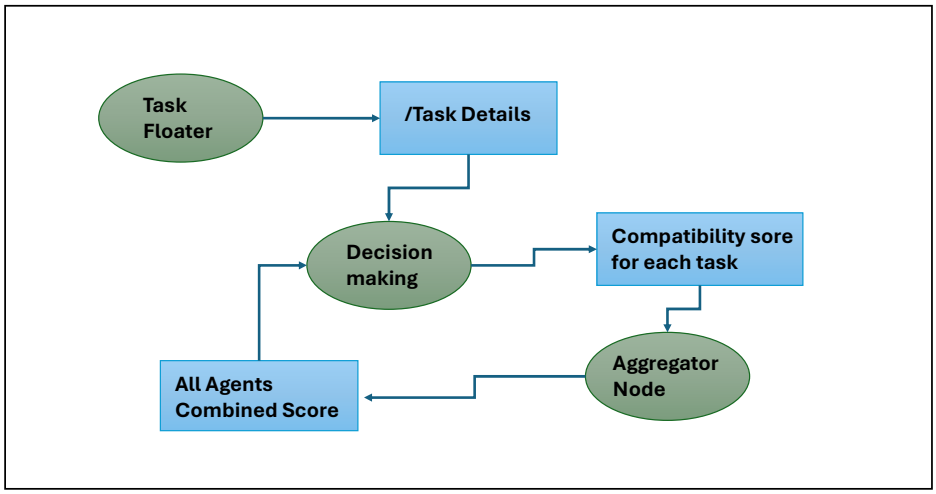

Figure 7.1: Task Allocation Flowchart

The `Task Floater` node will publish task details on the */Task_Details* topic. The `Decision Making` node of each agent will subscribe to this topic and will publish a compatibility score for each task on */global/task_allocation_discussion*. This topic will contain scores for each task from each agent, but the messages will arrive at different timestamps.

To consolidate these into a complete scorecard, the `Aggregator` node comes into play. It synchronizes the data to a common timestamp and publishes the aggregated scores to */global/aggregated_task_allocations*. Each agent's `Decision Making` node then subscribes to this topic, checks whether it is the best fit for any task, and if so, publishes the result to */global/allocated_tasks*.

Finally, the `Navigation` node of the selected agents receives the assigned task information and begins execution.

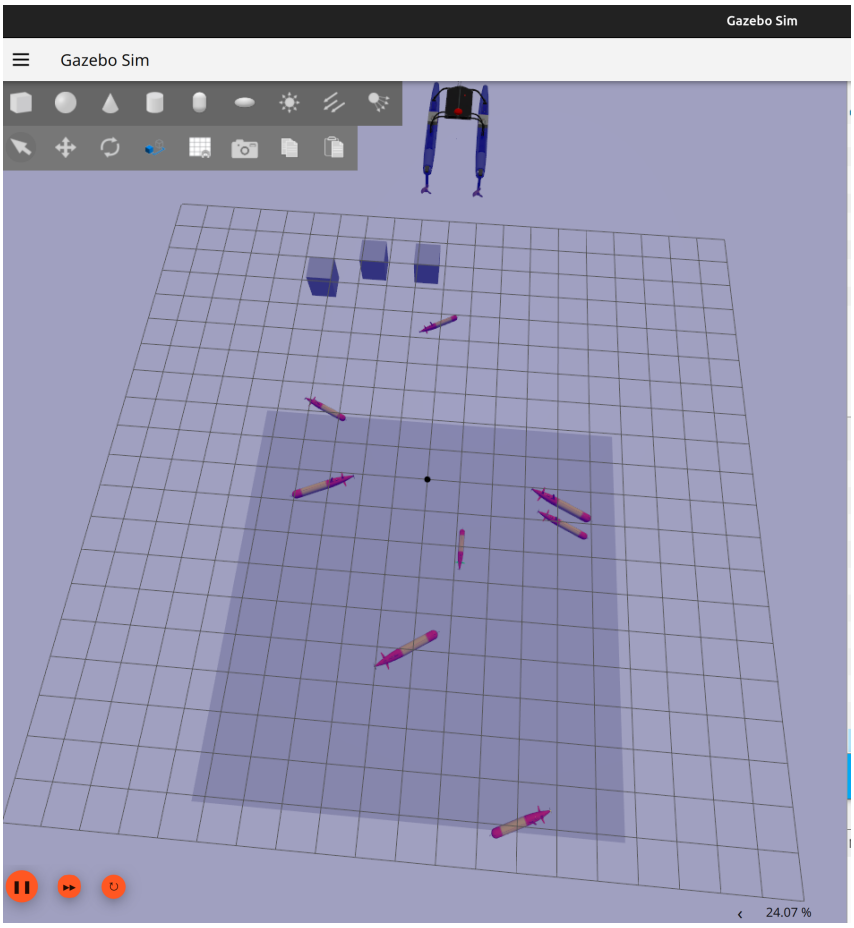

Figure 7.2: Gazebo World

Problem Definition: There are 8 agents and 4 tasks will be floated. Now agents need to communicate between themselves to decide which agent is going to do which task based on their capability.

Table 7.1: Current Position of agents

Agent Name	Position	Yaw
my_lrauv_1	(-4.99,0,-2)	2.12
my_lrauv_2	(0,-5,-2)	1.12
my_lrauv_3	(5,5,-6)	1.12
my_lrauv_4	(0,4,-2)	-1.15
my_lrauv_5	(-11,-4,-5)	-1.15
my_lrauv_6	(0,-6,-5)	1.12
my_lrauv_7	(12,0,8.5)	1.98
my_lrauv_8	(0,-2,-11)	(3.12)

Table 7.2: Task Details

Task Number	Goal Position
1	(20,10,-6)
2	(20,105,-9)
3	(100,20,-20)
4	(25,30,-8)

```
---
agent_name: my_lrauv_2
scores:
- task_id: 3
  final_distance: 28.70973778574929
stamp:
  sec: 0
  nanosec: 0
round_number: 1
---
agent_name: my_lrauv_1
scores:
- task_id: 3
  final_distance: 59.3982008479836
stamp:
  sec: 0
  nanosec: 0
round_number: 1
---
agent_name: my_lrauv_6
scores:
- task_id: 1
  final_distance: 38.59796767068401
stamp:
  sec: 0
  nanosec: 0
round_number: 1
---
agent_name: my_lrauv_6
scores:
- task_id: 2
  final_distance: 31.05947957642247
stamp:
  sec: 0
  nanosec: 0
round_number: 1
---
agent_name: my_lrauv_6
scores:
- task_id: 3
  final_distance: 30.022635648962567
stamp:
  sec: 0
  nanosec: 0
round_number: 1
---
```


Figure 7.3: Task Allocation Discussion

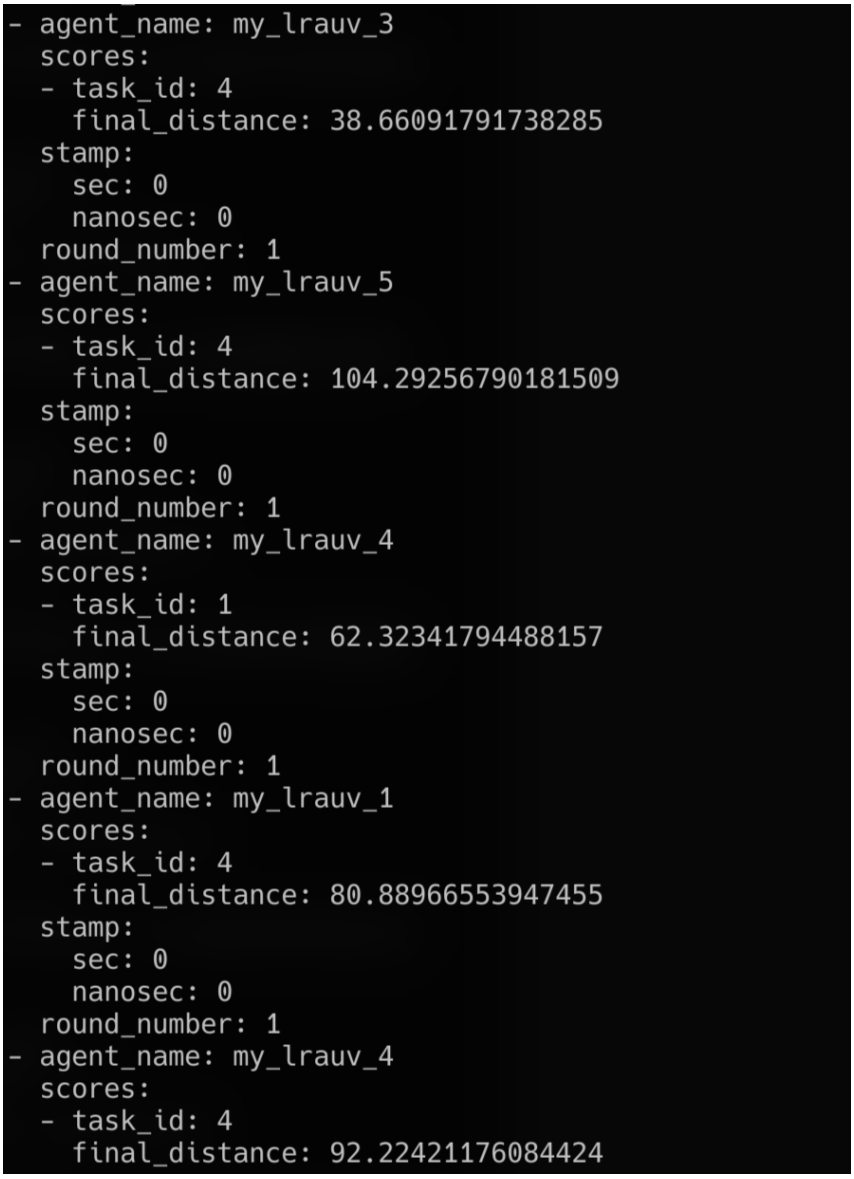

Figure 7.4: Aggregated Task Allocation

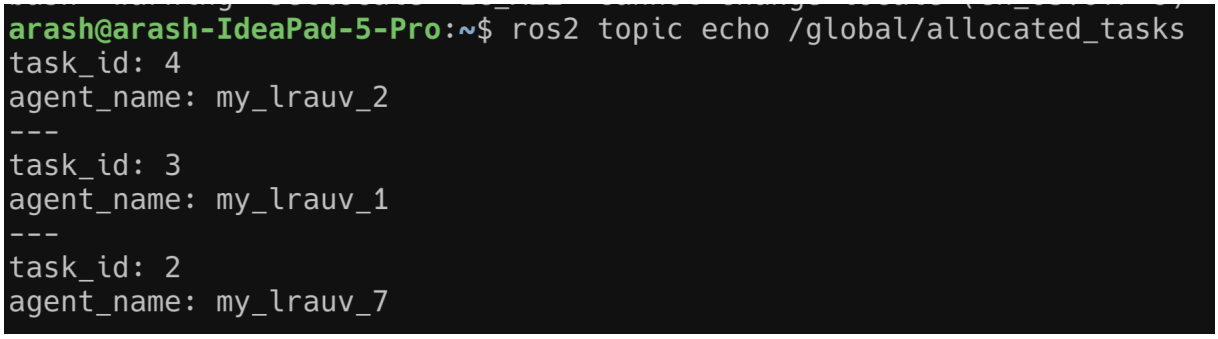

Figure 7.5: Final Decision

7.2 Dynamic Modulation Matrix

Problem Description:- I will be demonstrating cases of testing this obstacle avoidance method. A cube-shaped obstacle is used. First the AUV is given some desired goal and it's trajectory is traced. Then, the obstacle is placed in between trajectory. Then, the trajectory is traced with the presence of an obstacle.

Case 1:-

Table 7.3: Case 1-Parameters of DMM

Lambda (λ)	**Safe Distance**
1 - $\left(\frac{\text{last\_obstacle\_distance}^2}{\text{safe\_distance}^2}\right)$	15 m

Table 7.4: DMM Case 1 - Goal and Obstacle Position

Goal	Obstacle
(40,0)	(15, 0)

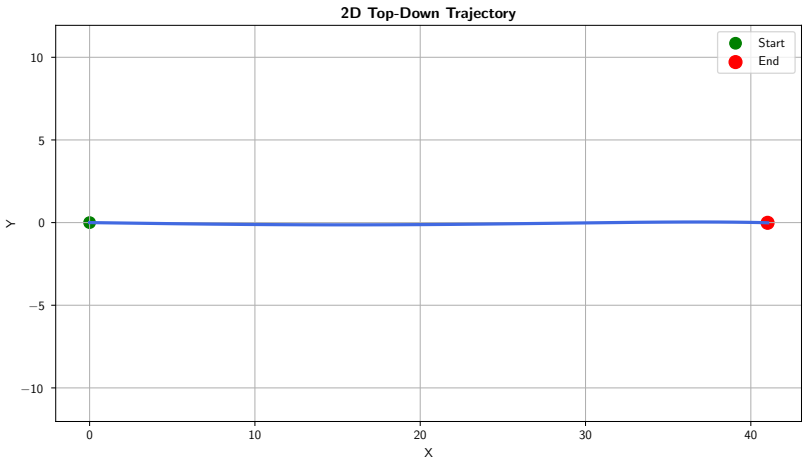

Figure 7.6: Top View Trajectory without Obstacle

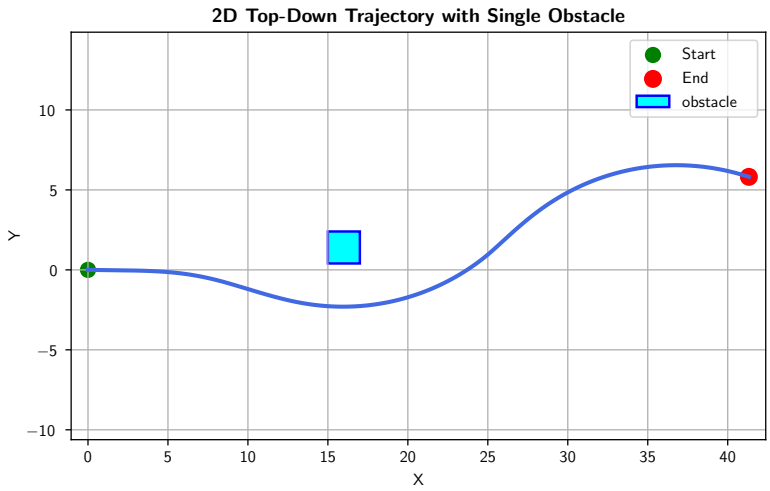

Figure 7.7: Top View Trajectory with Obstacle

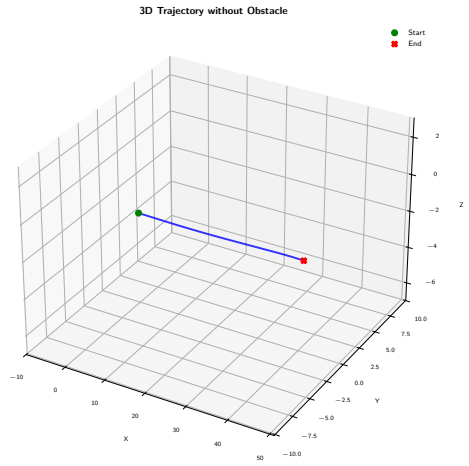

Figure 7.8: 3D View of Trajectory Without Obstacle

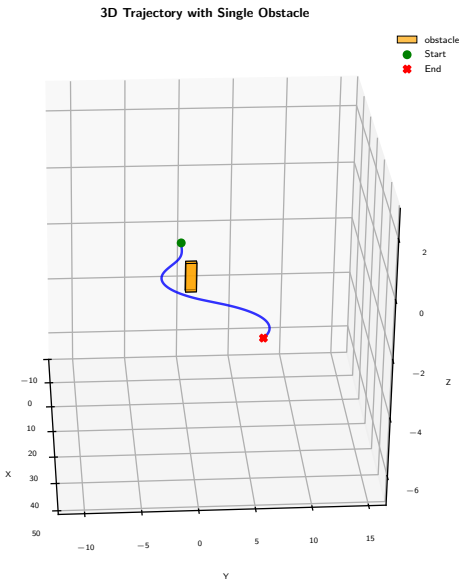

Figure 7.9: 3D view of Trajectory With Obstacle

Case 2:-

Table 7.5: DMM Case 2-Parameters of DMM

Lambda (λ)	**Safe Distance**
1 - $\left(\frac{\text{last\_obstacle\_distance}^2}{\text{safe\_distance}^2}\right)$	7 m

Table 7.6: DMM Case 2 - Goal and Obstacle Position

Goal	Obstacle
(40,0)	(15,0)

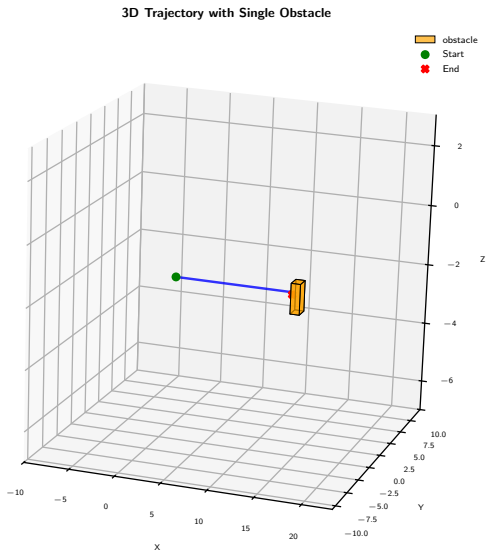

Figure 7.10: 3D view of Trajectory With Obstacle

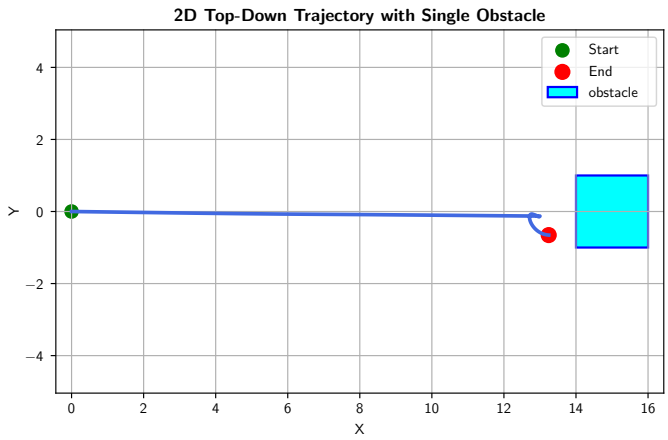

Figure 7.11: 2D view of Trajectory With Obstacle

Case 3:-

Table 7.7: Case 3-Parameters of DMM

Lambda (λ)	**Safe Distance**
1 - $\left(\frac{\text{last\_obstacle\_distance}^2}{\text{safe\_distance}^2}\right)$	15 m

Table 7.8: DMM Case 3 - Goal and Obstacle Position

Goal	Obstacle_1	Obstacle_2	Obstacle_3
(40,0)	(15,-1)	(18,1.5)	(15,3)

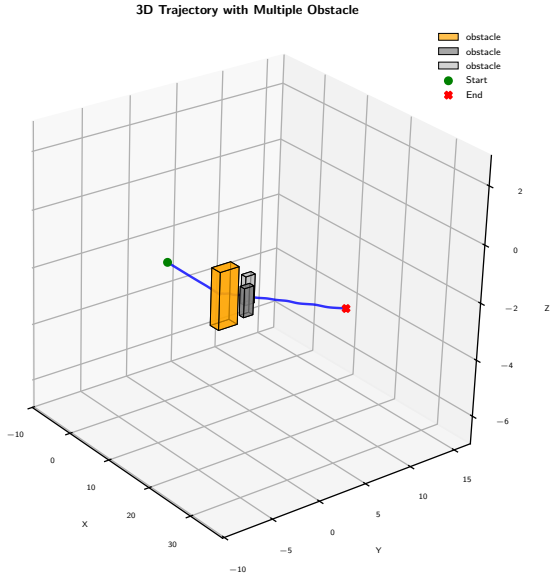

Figure 7.12: 3D view of Trajectory With Obstacles

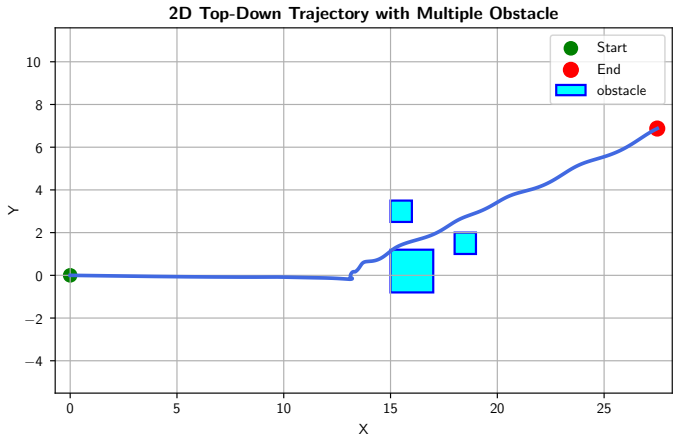

Figure 7.13: 2D view of Trajectory With Obstacles

7.3 Rotational Obstacle Avoidance Method

Problem Description: The demonstration of this method will be done in situations similar to DMM. The initial position, goal position, and obstacle placement will be the same so that both methods can be compared.

Case 1:-

Table 7.9: ROAM Case 1 - Goal and Obstacle Position

Goal	Obstacle
(40,0)	(15, 0)

Table 7.10: ROAM Parameters

Safe Distance	Gamma(γ)	R_e
4m	2	$\pi/2$

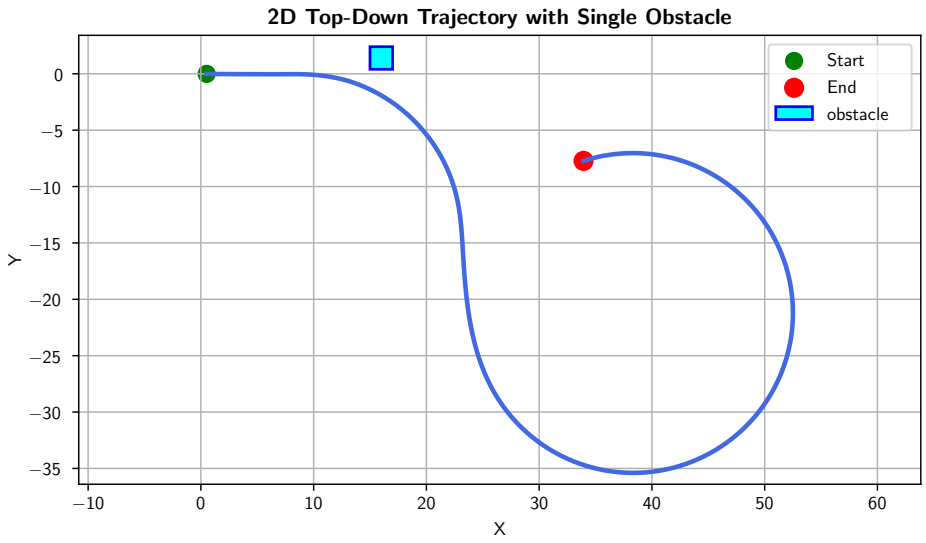

Figure 7.14: 2D View of Trajectory With Single Obstacle

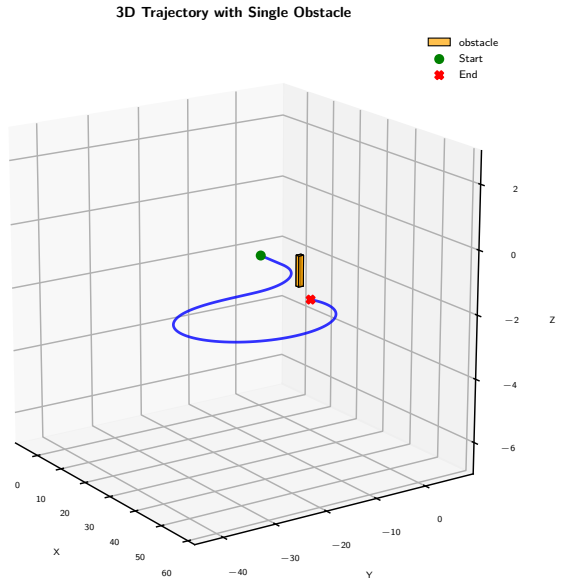

Figure 7.15: 3D View of Trajectory With Single Obstacle

Case 2:-

Table 7.11: Case 2 - Goal and Obstacle Position

Goal	Obstacle
(40,0)	(15, 0)

Table 7.12: ROAM Parameters

Safe Distance	Gamma(γ)	R_e
10m	1.5	π

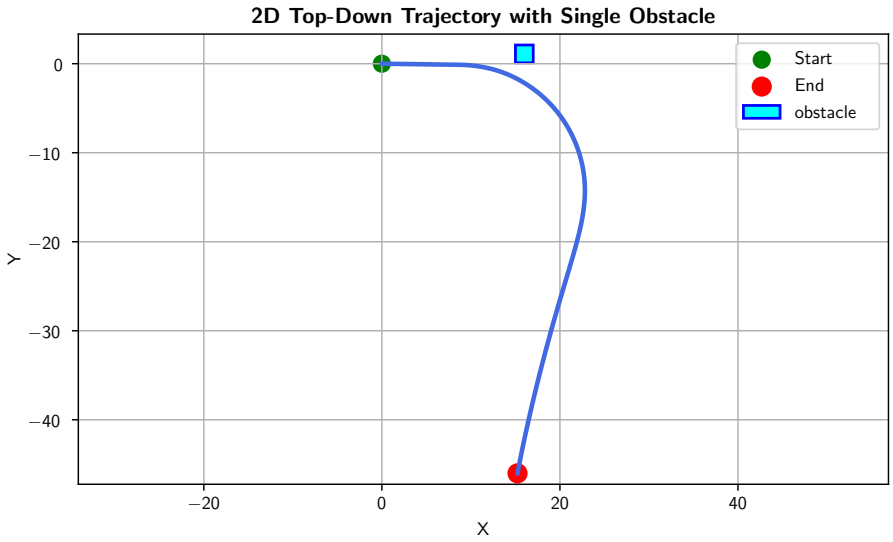

Figure 7.16: 2D View of Trajectory With Single Obstacle

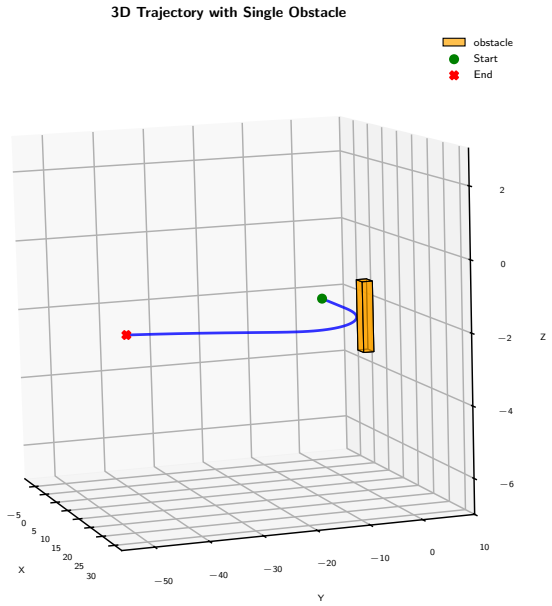

Figure 7.17: 3D View of Trajectory With Single Obstacle

Case 3:-

Table 7.13: ROAM Case 3 - Goal and Obstacle Position

Goal	Obstacle_1	Obstacle_2	Obstacle_3
(40,4)	(10, 0.8)	(13,3)	(9,5)

Table 7.14: Case 3 -ROAM Parameters

Safe Distance	Gamma(γ)	R_e
4m	2	$\pi/2$

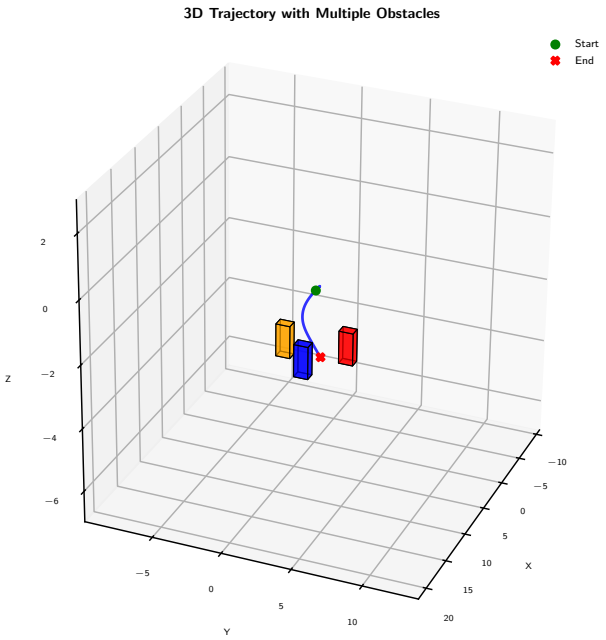

Figure 7.18: 3D View of Trajectory With Obstacles

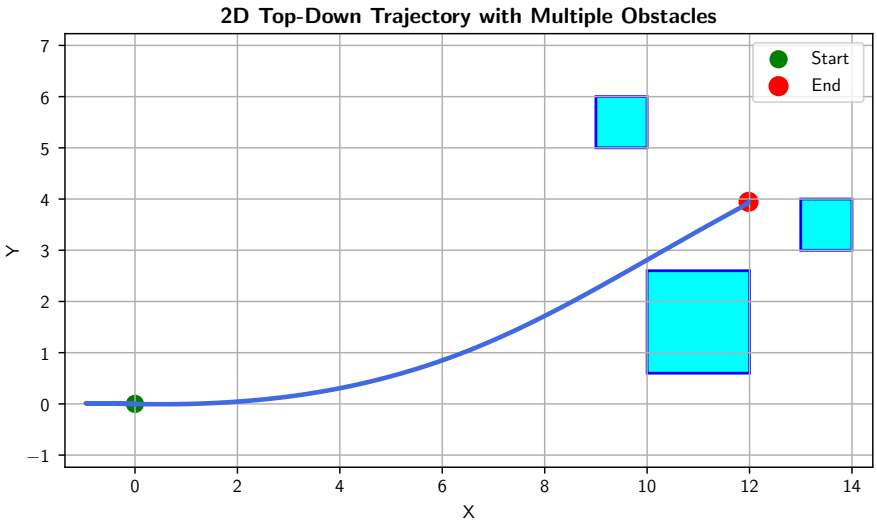

Figure 7.19: 2D View of Trajectory With Obstacles

7.4 Analysis

Comparison Between DMM and ROAM

To evaluate the performance of the obstacle avoidance methods, both **Dynamic Modulation Method (DMM)** and **Rotational Obstacle Avoidance Method (ROAM)** were tested in simulation under various obstacle configurations using **ROS2** and **Gazebo**. The focus was on assessing their effectiveness in terms of *goal convergence*, *response sharpness*, and *ability to navigate around single and multiple obstacles*.

1.Single Obstacle Scenario DMM demonstrated smooth and stable convergence toward the goal with minimal deviation in trajectory. Its modulation-based strategy allowed the agent to gently curve around the obstacle while continuously progressing toward the target.

ROAM, in contrast, produced sharper and faster reactions. The rotational strategy made the agent aggressively avoid the obstacle. While this improved speed and reactivity, it occasionally resulted in longer path lengths due to wider turns.

2.Multi-Obstacle Scenario DMM struggled in environments with closely spaced or multiple obstacles. The modulation matrix often failed to account for combined obstacle influences, causing agents to get trapped.

ROAM excelled in multi-obstacle environments. Its ability to rotate around complex shapes and dynamically adjust its trajectory allowed it to maintain motion continuity and avoid deadlocks. The method successfully navigated agents through narrow passages with better clearance and adaptability.

Comparison Summary

Table 7.15: Performance comparison between DMM and ROAM

Criterion	DMM	ROAM
Goal Convergence	High	Moderate
Obstacle Avoidance	Effective (single obstacle)	Effective (all cases)
Sharpness/Responsiveness	Smooth but slower	Sharp and fast
Performance (Multi-obstacle)	Poor	Excellent
Trajectory Smoothness	High	Moderate

While DMM is advantageous for scenarios where smooth, direct convergence to the goal is critical, ROAM proves to be more versatile and robust in cluttered environments, thanks to its quick reaction and adaptability.

8 Summary and Future plan of work

This thesis presents the design, development, and simulation of a decentralized multi-agent system that performs self task allocation and real-time obstacle avoidance in dynamic environments. The framework is built on the Robot Operating System 2 (ROS2) and simulated using the Gazebo environment, providing a realistic and modular setup for robot modeling, sensing, and control.

A key contribution is the implementation of a capability-based self task allocation strategy, where each agent autonomously evaluates its own suitability for available tasks and selects the most appropriate one without relying on centralized coordination. To ensure safe and efficient navigation in obstacle-rich environments, the system integrates two obstacle avoidance strategies: the Dynamic Modulation Method (DMM), which modulates velocity fields to smoothly deflect trajectories around obstacles, and the Rotational Obstacle Avoidance Method (ROAM), which enables agents to generate rotational motion around obstacles to maintain safe clearance and path continuity.

The system is demonstrated in simulation using Surface Vehicles and Underwater Vehicles, showcasing their ability to allocate tasks, avoid collisions, and navigate cooperatively. The use of ROS2 and Gazebo enables high-fidelity simulation, real-time interaction, and future adaptability for real-world deployment. This work contributes to scalable and resilient solutions in the domains of underwater exploration, search-and-rescue, and multi-agent robotic operations.

Future Plan of Work

- Full body obstacle avoidance for tail avoidance of AUVs
- Testing ROAM Algorithm on Dynamic Obstacles

References

[1] M. S. Mohd Aras, H. Kasdirin, M. Jamaluddin, and M. Basar, "Design and development of an autonomous underwater vehicle (auv-fkeutem)," *Proceedings of MUCEET2009 Malaysian Technical Universities Conference on Engineering and Technology, MUCEET2009, MS Garden, Kuantan, Pahang, Malaysia*, 01 2009.

[2] *Models for Ships, Offshore Structures and Underwater Vehicles*. John Wiley Sons, Ltd, 2011, ch. 7, pp. 133–186. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119994138.ch7

[3] R. Hu, D. Lu, C. Xiong, C. Lyu, H. Zhou, Y. Jin, T. Wei, C. Yu, Z. Zeng, and L. Lian, "Modeling, characterization and control of a piston-driven buoyancy system for a hybrid aerial underwater vehicle," *Applied Ocean Research*, vol. 120, p. 102925, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S014111872100393X

[4] A. Sahoo, S. K. Dwivedy, and P. Robi, "Advancements in the field of autonomous underwater vehicle," *Ocean Engineering*, vol. 181, pp. 145–160, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0029801819301623

[5] A. Natu, V. Garg, P. Gaur, U. Biswas, D. Bansal, N. Biswas, M. Ranjan, M. Goyal, R. Gupta, S. Parashar, R. Bansal, C. Kumar, R. Kumar, A. Kumar, A. Tyagi, A. Verdhan, D. Auv, Dhruv, P. Raj, and S. Sain, "Design and development of an autonomous underwater vehicle varuna 2.0," 08 2020.

[6] L. W. L. Alexander, M. M. A. Roslan, K. Isa, H. A. Kadir, and R. Ambar, "Design and development of an autonomous underwater vehicle (auv) with target acquisition mission module," in *2018 IEEE 8th International Conference on Underwater System Technology: Theory and Applications (USYS)*, 2018, pp. 1–6.

[7] C. Wang, D. Mei, Y. Wang, X. Yu, W. Sun, D. Wang, and J. Chen, "Task allocation for multi-auv system: A review," *Ocean Engineering*, vol. 266, p. 112911, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0029801822021941

[8] E. Bischoff, F. Meyer, J. Inga, and S. Hohmann, "Multi-robot task allocation and scheduling considering cooperative tasks and precedence constraints," in *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2020, pp. 3949–3956.

[9] Y. Ma, Y. Liu, L. Zhao, and M. Zhao, "A review on cooperative control problems of multi-agent systems," in *2022 41st Chinese Control Conference (CCC)*, 2022, pp. 4831–4836.

[10] Z. Fang, D. Jiang, J. Huang, C. Cheng, Q. Sha, B. He, and G. Li, "Autonomous underwater vehicle formation control and obstacle avoidance using multi-agent generative

adversarial imitation learning," *Ocean Engineering*, vol. 262, p. 112182, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0029801822014986

[11] M. Liu and Y. Feng, "Group consensus of mixed-order multi-agent systems with fixed and directed interactive topology," *IEEE Access*, vol. 7, pp. 179 712–179 719, 2019.

[12] Y. Bai, Y. Wang, M. Svinin, E. Magid, and R. Sun, "Adaptive multi-agent coverage control with obstacle avoidance," *IEEE Control Systems Letters*, vol. 6, pp. 944–949, 2022.

[13] L. Wen, J. Yan, X. Yang, Y. Liu, and Y. Gu, "Collision-free trajectory planning for autonomous surface vehicle," in *2020 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, 2020, pp. 1098–1105.

[14] L. Cao, G.-P. Liu, and D.-W. Zhang, "A leader-follower consensus control of networked multi-agent systems under communication delays and switching topologies," in *2022 41st Chinese Control Conference (CCC)*, 2022, pp. 4378–4383.

[15] J. Xu, F. Huang, D. Wu, Y. Cui, Z. Yan, and K. Zhang, "Deep reinforcement learning based multi-auvs cooperative decision-making for attack–defense confrontation missions," *Ocean Engineering*, vol. 239, p. 109794, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0029801821011562

[16] B. Xu, Z. Wang, and H. Shen, "Distributed predictive formation control for autonomous underwater vehicles under dynamic switching topology," *Ocean Engineering*, vol. 262, p. 112240, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0029801822015505

[17] L. Huber, A. Billard, and J.-J. Slotine, "Avoidance of convex and concave obstacles with convergence ensured through contraction," *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 1462–1469, Apr. 2019.

[18] L. Huber, J.-J. Slotine, and A. Billard, "Avoidance of concave obstacles through rotation of nonlinear dynamics," *IEEE Trans. Robot.*, vol. 40, pp. 1983–2002, 2024.

[19] S. M. Khansari-Zadeh and A. Billard, "A dynamical system approach to realtime obstacle avoidance," *Auton. Robots*, vol. 32, no. 4, pp. 433–454, May 2012.

[20] L. Huber, "Exact obstacle avoidance for robots in complex and dynamic environments using local modulation," PhD thesis, EPFL, Lausanne, Switzerland, April 2024.

[21] accurrent. (April) Mbari tethys lrauv. Open Robotics. [Online]. Available: https://fuel.gazebosim.org/1.0/accurrent/models/MBARITethysLRAUV

[22] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot operating system 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, p. eabm6074, 2022. [Online]. Available: https://www.science.org/doi/abs/10.1126/scirobotics.abm6074