https://doi.org/10.31449/inf.v50i11.9866 Informatica **50** (2026) 143–162 **143**

Hybrid MIRL and ACO-based Approach for Real-Time Path Planning in Visual SLAM

Jin Li[1,*], Chunyu Yang[2]
[1]College of Big Data and Artificial Intelligence, Xinyang University, Xinyang 464000, Henan, China
[2]School of Information Engineering, Nanyang Vocational College of Science and Technology, Dengzhou 474150, Henan, China
E-mail: lijin0318@outlook.com
[*]Corresponding author

Keywords: visual SLAM, Multi-intelligence reinforcement learning, ant colony optimization, real-time path planning, autonomous navigation

Received: June 25, 2025

For autonomous robotic systems, real-time establishing paths in dynamic situations is still a major difficulty. Current Visual SLAM-integrated planners, including A\ and Dijkstra, frequently perform poorly in uncertain situations due to a lack of flexibility and collaborative intelligence. In order to improve navigation, this study presents MIRL-ACO-SLAM, a hybrid framework that combines Ant Colony Optimization (ACO) and Multi-Intelligence Reinforcement Learning (MIRL). Real-time spatial maps are created by the network using Visual SLAM (ORB-SLAM3), which allows agents to make decisions locally through reinforcement learning. Pheromone-based optimization guarantees global convergence to the best solutions. Pheromone-guided pruning and selective agent activation improve scalability on bigger maps while lowering computing costs. In contrast with traditional SLAM-based planners, MIRL-ACO-SLAM delivers 18.6% shorter route length, 24.3% quicker scheduling speed, with a 94% route completion rate in dynamic situations, according to empirical assessments conducted utilizing the Zurich MAV dataset. The suggested system opens the door for implementation in mission-critical applications like transportation and search-and-rescue by offering a dependable, scalable, and adaptable answer for real-time robotic navigation.

Povzetek: Študija predstavi izboljšan pristop za učinkovitejšo navigacijo avtonomnih robotov v dinamičnih okoljih.

1 Introduction

Intelligent robotic systems have transformed autonomous automobiles, surveillance drones, warehouse robots, and search-and-rescue equipment. Real-time interaction with their environment requires autonomy, flexibility, and decision-making that tolerates uncertainty in these systems [1]. Such activities require the ability to design and execute courses effectively while overcoming dynamic barriers and navigating unexpected or partially understood terrain. In complex and ever-changing real-world contexts, traditional navigation methods often fail, necessitating more resilient, intelligent, and adaptable path-planning algorithms.

SLAM is crucial for autonomous systems that operate without prior knowledge of their environment. A robot can build a map of an unfamiliar area and estimate its position using SLAM [2]. This dual feature lets robots navigate independently. V-SLAM, which utilizes visual data (monocular or stereo cameras), has gained popularity due to its ability to collect rich scene information at minimal cost and weight [3]. ORB-SLAM2 and ORB-SLAM3 are reliable visual SLAM systems for indoor and outdoor spatial awareness. Autonomy requires effective path planning based on the created map, not only SLAM.SLAM provides mapping accuracy, although many path-planning systems utilize static or heuristic algorithms, such as Dijkstra and A* [4]. These approaches may create collision-free routes, but they struggle to respond to abrupt environmental changes, such as moving obstructions or dynamic terrain. These restrictions limit its use in real-world applications that need adaptation and expertise. Indeed, static route planners generally perform poorly in uncertain, complicated, or dynamic contexts.

Recent research has integrated bio-inspired algorithms and reinforcement learning into robotic path planning systems to solve these constraints. Ant Colony Optimization (ACO), inspired by ant foraging, explores different route choices using a distributed and probabilistic approach[5]. Systems can explore more intelligently, adapt, and collaborate in navigation decisions when paired with Reinforcement Learning (RL), especially in multiagent settings. Multiagent Reinforcement Learning (MARL) enables agents to learn from their local environment and exchange experiences, thereby making planning more robust and scalable. ACO and MARL combine to address the inadequacies of

conventional planners by incorporating global investigation and local flexibility [6].

SLAM-constructed environments are used to combine various techniques in this research. Robots can plan more innovative, quicker, and reliably using Visual SLAM environmental data, adaptive learning from MARL, and cooperative search from ACO [7]. This integration enables navigation in dynamic environments, learning from barriers, facilitating real-time rerouting, and selectively activating agents to optimize the management of computing resources. The strong, unified framework—MIRL-ACO-SLAM—improves robotic autonomy and enables next-generation intelligent navigation systems in real-world applications. Superior Performance: On the Zurich MAV dataset, it shows an 18.6% shorter path length, a 24.3% faster planning speed, and a 94% route completion rate, making it better than standard SLAM-based planners.

- Hybrid Path Planning Framework, which combines Multi-Intelligence Reinforcement Learning (MIRL) and Ant Colony Optimization (ACO) in a new way to make real-time path planning in Visual SLAM environments more stable.
- Combining localized reinforcement learning for dynamic adaptability with pheromone-based global optimization for convergence to optimal paths is what adaptive local-global decision making is all about.
- Improved Efficiency and Scalability: pheromone-guided pruning and selective agent activation are used to lower computational costs and make it easier to use on big maps.

1.1 Research problem and objectives

In unexpected conditions, autonomous robotic systems struggle with real-time path planning. Visual SLAM enables real-time mapping and localization, although most SLAM-integrated navigation systems employ static path planning algorithms, such as Dijkstra or A* [8]. These algorithms presume a constant environment and lack learning or prediction capabilities for moving barriers or terrain changes. Robots may not reroute effectively, causing inefficient navigation, delays, or collisions in mission-critical situations. These systems primarily utilize single agents and lack collaborative decision-making, which limits their scalability and effectiveness in complex or large-scale situations.

The mismatch between SLAM-based environment observation and intelligent path-planning decision-making is another important constraint. SLAM offers spatial awareness, but optimizing navigation based on changing circumstances needs intelligent behavior that static planners cannot deliver [9]. Traditional methods also overlook cooperative agent-based systems and learning-based adaptability mechanisms. A single f=ramework that dynamically learns from the environment adapts its behavior, and cooperatively plans courses in real-time is needed for efficiency and scalability [10]. A hybrid model that combines multiagent reinforcement Learning (MIRL) and Ant Colony Optimization (ACO) in a Visual SLAM environment provides a robust and adaptable alternative to classic route planners. To address this gap, the objectives of this research are:

1. To develop a hybrid path planning framework combining MIRL and ACO for improved real-time decision-making in Visual SLAM situations.
2. To provide adaptive and decentralized agent-based learning methods for collaborative navigation and efficient obstacle avoidance in dynamic and unpredictable environments.
3. Enhance scalability, planning speed, and route efficiency through pheromone-guided exploration and policy sharing, surpassing classic SLAM planners like Dijkstra and A*.

1.2 Methodology

Statement of Novelty and Contribution: Three distinct contributions to the body of current literature are introduced by the proposed MIRL-ACO-SLAM framework:

1. Hybrid Integration: Compared to conventional SLAM-based planners, MIRL-ACO-SLAM combines pheromone-driven ant colony optimization and multi-agent reinforcement learning to allow for a balance between global path convergence and local flexibility.
2. Selective Agent Activation: The method enhances scalability and lowers computational overhead without sacrificing navigation accuracy by dynamically activating just the most pertinent agents.
3. Pheromone-Guided Pruning: In extremely dynamic contexts, resilience is ensured via a unique pheromone filtering technique that speeds up path convergence while preventing premature exploitation. When taken as a whole, these developments improve scalability, computing efficiency, and real-time flexibility, making MIRL-ACO-SLAM a substantial improvement over the most advanced methods for SLAM-enabled robotic navigation.

MIRL-ACO-SLAM combines Visual SLAM, Ant Colony Optimization, and Multiagent Reinforcement Learning. ORB-SLAM3 creates a real-time spatial map from monocular or stereo camera input [11]. This geographical information is the agents' environment. Each agent makes local decisions via reinforcement learning. Meanwhile, ACO methods utilize pheromone trails to guide agents toward globally optimal pathways. Local knowledge and global optimization enable agents to overcome new challenges and enhance path efficiency [12]. As the environment changes, the system modifies regulations and pheromone values, making it flexible and scalable. The main contributions are:

1. To propose a hybrid framework combining MARL and ACO to improve decision-making in SLAM settings by optimizing localized learning and global paths.

2. To reduce computational complexity and increase scalability by introducing a dynamic validator selection method based on trust scores.
3. To enable real-time response by collaborating agents and using pheromone-guided filtering for rapid adaptability to new barriers and map modifications.
4. To overcome standard path planning approaches (e.g., Dijkstra, A*) with 18.6% shorter path lengths and 24.3% quicker planning speed.
5. To verify the system's 94% route success rate in dynamic barrier situations, ensuring reliable navigation in unexpected environments.

The remainder of the paper is organized as follows: Section 2 discusses the literature on SLAM, ACO, and reinforcement learning-based navigation. Section 3 describes the architecture and algorithms of MIRL-ACO-SLAM. Section 4 describes the experimental setup, datasets, and assessment measures. Section 5 gives data and analysis comparing the proposed system to existing planners. Section 6 concludes the report and outlines future research, including the deployment of real-world robotic systems.

2 Related work

2.1 Visual SLAM and real-time path planning

Xu et al.[13] ORB RGB-D SLAM is enhanced with 2D Occupancy Grid Mapping for precise, real-time indoor localization, navigation, and path planning. Using ROS for visualization and a collection of visual markers in a vast building, the system achieved localization accuracy ranging from 0.039 to 0.186 m. In textureless or poorly light surroundings, its RGB-D sensors and pre-installed markers may hinder performance.

Sud et al.[14] The Multiagent Navigation Graph (MaNG) technique uses first- and second-order Voronoi diagrams for real-time multiagent path planning and proximity computing. In pursuit-evasion and terrain exploration scenarios employing bespoke crowd simulation datasets, the technique efficiently manages hundreds of agents with different aims. Environmental undersampling and GPU-based acceleration are drawbacks that limit optimal performance.

Hu et al.[15] An improved visibility graph, bidirectional A*, and minimal construction algorithms are employed in this laser SLAM-based path planning system to accelerate the planning process. LiDAR-generated occupancy grid maps outperform D* Lite, FAR, and FPS in field and simulation testing for navigation, path length, and calculation. In incredibly crowded or dynamic contexts with limited visibility, graph simplification may deteriorate performance.

Cao et al.[16] The paper introduces SwarmMap, a scalable edge-based multiagent visual SLAM system featuring change log-based synchronization, priority-aware scheduling, and a lean global map structure. Over a three-month deployment in a large oilfield using synthetic multiagent datasets, SwarmMap increased agent support (>20 agents) and reduced trajectory error by over 55%. Edge-resource reliance may hinder performance on low-bandwidth or unstable networks.

Ubaid et al.[17] H-PSO-VSLAM, a hybrid route planning system, combines Visual SLAM with a Hybrid Particle Swarm Optimization method augmented by simulated annealing and dimensional learning. Monocular camera data to simulate UAV flights demonstrated safe navigation, optimal energy consumption, reduced risk, and improved convergence speed—although limited resilience was observed in GPS-denied or low-feature situations, as well as with the use of simulated data.

2.2 Reinforcement learning and multiagent intelligence

Kim et al.[18] This paper presents an innovative multiagent reinforcement learning (MARL) system for manufacturing, where intelligent robots constantly negotiate and learn task scheduling. The approach outperformed standard dispatching algorithms in terms of early completion rates, productivity, and delay reduction on simulated production datasets with varying demand and work priorities. Complex processes with minimal training data or unforeseen external disturbances can significantly impact performance.

Xia et al.[19] A cooperative multiagent reinforcement learning (MARL) system for UAV-based target tracking utilizes energy-saving tactics and spatial entropy to enhance coverage. We achieved better tracking, lower power consumption, and improved coverage compared to deep RL baselines in simulated UAV settings and dynamic target trajectories. The system may perform poorly in congested or communication-constrained airspaces.

Canese et al.[20] The Georgia Tech Robotarium platform was used to evaluate the multiagent reinforcement learning algorithm Q-RTS (Q-Learning for Real-Time Swarm). Combining 4 and 8 agents reduced the convergence time to 500,000 iterations, compared to over 1 million iterations with a single agent. However, scalable and stable outcomes depend on a well-designed reward function and are difficult to apply on a big scale.

Belgacem et al.[21] A Q-learning-based multiagent system, IMARM (Intelligent Multiagent Resource Management), is proposed for dynamic cloud resource allocation. IMARM outperformed current approaches in energy economy, fault tolerance, and balanced execution times in simulated heterogeneous cloud systems with different workloads. Under strong workload surges, its performance may fluctuate, requiring fine-tuned incentive settings to maintain constant flexibility.

2.3 Ant colony optimization and collaborative bio-inspired search

Khan et al.[22] To reduce electricity costs and the peak-to-average ratio, this work proposes HB-ACO (Hybrid Bacterial Foraging–Ant Colony Optimization) for smart home energy scheduling as a knapsack problem.

Time-of-use and critical peak pricing simulations demonstrate that HB-ACO outperforms ACO and BFA in terms of cost savings and user comfort. Its efficacy may decrease in real-time scenarios with unexpected appliance behavior or pricing swings.

Wang et al.[23] This research presents an Ant Colony Optimization (ACO)-based clustering method with mobile sink support for home automation networks, aiming to reduce energy consumption and extend network lifetime. Simulations utilizing ZigBee-based sensor network models demonstrated that the suggested routing method balanced energy load and improved efficiency better than current techniques. However, accurately controlling sink mobility may be challenging in real-world installations with unexpected impediments or interference.

Zhang et al.[24] This assessment examines bio-inspired algorithms, including ACO, bee colony models, immune systems, and PCO-based synchronization to improve artificial self-organized networking (SON) systems. The paper highlights the algorithmic benefits, tradeoffs, and application situations spanning physical, MAC, and network levels without relying on a dataset. The lack of consistent benchmarks and the necessity for real-world validation in complex, large-scale, heterogeneous networks are limitations.

Rokbani et al.[25] This paper presents hybrid bi-heuristic algorithms—ASFPA-Ls, Co-ASPSO-Ls, So-ASPSO-Ls, and AS-chaotic-PSO-Ls—that solve TSP cases using ACO, flower pollination, PSO variations, and 2-opt local search. On the Berlin52, Eil76, Rat99, and KroA200 benchmark datasets, the approaches yield near-optimal solutions with low error rates and a good time-performance balance. Performance varies per instance, and parameter sensitivity limits large-scale issues.

Table 1: Comparative analysis of existing bio-inspired, SLAM-based, and multiagent reinforcement learning methods with identified research gaps

Ref.No.	Author	Method/Technique	Application Area	Key Results	Research Gap / Limitation
[13]	Xu et al.	ORB RGB-D SLAM + OGM	Indoor localization	High accuracy (0.039–0.186 m)	Weak in low-light or textureless areas
[14]	Sud et al.	MaNG (Voronoi Graph)	Multiagent planning	Real-time with many agents	Needs GPU; limited in low-resolution maps
[15]	Hu et al.	Laser SLAM + A*	Robot navigation	Fast, accurate paths	Struggles in dense or dynamic scenes
[16]	Cao et al.	SwarmMap (Edge-SLAM)	Multiagent SLAM	Doubles agent support, 55% better accuracy	Dependent on stable edge networks
[17]	Ubaid et al.	H-PSO-VSLAM	UAV path optimization	Saves energy; avoids risk zones	Limited in GPS-denied or sparse-feature areas
[18]	Kim et al.	MARL for Manufacturing	Smart job scheduling	Better task completion and low delay	Needs extensive training data
[19]	Xia et al.	MARL for UAV tracking	Cooperative target following	Higher tracking rate, better energy use	Affected by poor communication in crowded areas
[20]	Canese et al.	Q-RTS (Swarm Learning)	Multiagent robotics	Faster convergence with more agents	It depends on reward design, limited scale testing
[21]	Belgacem et al.	IMARM (Q-Learning)	Cloud resource management	Efficient and fault-tolerant allocation	Sensitive to workload surges
[22]	Khan et al.	HB-ACO	Smart home energy scheduling	Lower costs and improved user comfort	Unstable under real-time changes
[23]	Wang et al.	ACO + Mobile Sink	Home automation networks	Energy balance and longer network life	Hard to control mobile sink in real conditions
[24]	Zhang et al.	Bio-inspired SON models	Network self-organization	Highlights multi-layer algorithm roles	No unified benchmarks; lacks deployment proof
[25]	Rokbani et al.	Hybrid ACO + PSO	TSP path optimization	Near-optimal results on benchmarks	High parameter sensitivity on large instances

Research on bio-inspired, SLAM-based, and multiagent reinforcement learning for robots, intelligent environments, and networked systems is discussed in Table 1. SwarmMap's agent assistance and HB-ACO's smart home cost reductions are examples of algorithms that enhance accuracy, scalability, energy efficiency, or flexibility. Common drawbacks include dependence on solid infrastructure (e.g., edge networks, GPS), sensitivity to parameter adjustments, and lower performance in dynamic or large-scale contexts. In multiagent and resource-constrained autonomous systems, more generic and robust algorithms for real-time decision-making under uncertainty are needed.

3 Methodology

MIRL-ACO-SLAM blends Multi-Intelligence Reinforcement Learning (MIRL) with Ant Colony Optimization (ACO) for adaptive path planning in dynamic situations utilizing Visual SLAM. A Visual SLAM module (e.g., ORB-SLAM3) receives camera pictures and GPS data to start the pipeline. To locate the agent and build spatial awareness, Feature Detection, Pose Estimation, Loop Closure, and 3D Map Generation occur here. This input is sent to the Reinforcement Learning Module, where agents use local perception to decide. Agents learn movement strategies using PPO or DQN based on local SLAM characteristics, velocity, and obstacle data. Agent policy sharing boosts convergence and adaptation. The SLAM-derived grid map-based ACO Module follows learned behavior. It uses heuristic data, pheromone matrices, and transition probabilities to update the route worldwide. Finally, an Adaptation and Performance Layer assesses Path Length, Success Rate, Dynamic Change Adaptability, and Scalability with expanding maps. This integration ensures strong, real-time, collaborative navigation. MIRL stands for Multi-Intelligence Reinforcement Learning. In our framework, MIRL is a "multi-agent reinforcement learning approach" in which many autonomous agents work together to learn and share policies in an environment based on SLAM. MIRL is different from traditional single-agent RL because it allows for shared decision-making, faster convergence through policy sharing, and better robustness in settings that change over time.

This is Selective Agent Activation (SAA): This new idea activates only the most important agents during navigation; based on how complicated the environment is at the moment and what the job requires. Standard MARL methods have all agents work all the time. SAA lowers the amount of work that needs to be done on the computer, stops exploration that is already being done, and makes it easier to add more agents without lowering the accuracy of guidance.

MIRL-ACO-SLAM framework consists of three closely related modules: Ant Colony Optimization (ACO) for global path convergence, Visual SLAM for real-time map generation, and Multi-Intelligence Reinforcement Learning (MIRL) for localized agent decision-making. In order to arrive at globally effective solutions, this pipeline makes sure that agents are able to sense their surroundings, pick up adaptive behaviors, and work together. Each module is covered in detail in the following subsections, and the Appendix contains complete derivations and pseudocode for reference. Before describing various modules, the MIRL-ACO-SLAM framework is briefly described for clarity. Visual SLAM, MIRL, and ACO are integrated in a four-stage pipeline (Fig. 1).

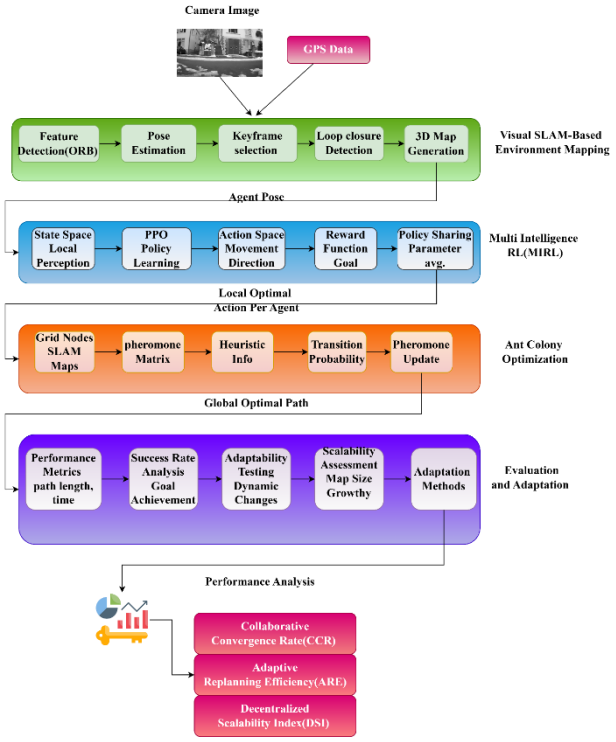

Figure 1: MIRL-ACO-SLAM Framework for Real-Time Path Planning in Visual SLAM Environments

3.1 Visual SLAM-based environment mapping

1) Visual SLAM-based mapping

ORB-SLAM3 processes input photos and IMU/GPS data to generate a real-time 3D environment map. This gives agents spatial awareness and dynamic decision space.

2) Local Decision-Making using MIRL

Each agent uses reinforcement learning (e.g., PPO/DQN) for local navigation decisions. Position, velocity, and barriers are state inputs for agents.

A policy sharing method accelerates agent convergence and collective learning.

3) ACO Global Optimization

Pheromone-based Ant Colony Optimization guides agents toward globally optimal courses in addition to local regulations.

Pheromone trails indicate shorter, safer pathways, while evaporation prevents stagnation.

This balanced exploration (new routes) with exploitation (best-known routes).

4) Dynamic Environment Adaptation

MIRL agents adjust policies quickly when obstacles or terrain change, while ACO refreshes pheromone maps. Selective agent activation reduces computing overhead by involving only relevant agents. Real-time re-planning keeps success rates high even in unpredictable settings. A four-stage hybrid architecture combines local adaptability (MIRL) with global convergence (ACO) for fast, scalable, and adaptive real-time navigation using a Visual SLAM-derived map.

MIRL-ACO-SLAM is able to preserve real-time spatial awareness because to this component. A dynamic 3D map for navigation is created by processing IMU/GPS inputs and monocular/stereo camera pictures using ORB-SLAM3. Agents are given a realistic environment model through the mapping process, which includes feature extraction, pose estimation, loop closure, and map generation.

Feature Extraction (ORB/BRIEF): Visual SLAM depends on finding strong keypoints that are not affected by illumination, rotation, or scale. ORB offers effective and repeatable characteristics for mapping and matching by combining the FAST detector alongside an orientation-based BRIEF descriptor. Pose Estimation & Package Adjustment: Bundle Adjustment optimizes camera postures and landmark placements, reducing reprojection error among

This component is essential to the MIRL-ACO-SLAM framework because it allows the agent to have a real-time understanding of its surroundings. It is made possible by utilizing Visual SLAM algorithms, which integrate data from a monocular camera, inertial measurement units (IMUs), and GPS. An accurate and dynamic 3D map may be constructed using the given dataset, which includes sequential photos of urban surroundings, IMU readings, and matching GPS records. This map can then be used to monitor the agent's journey across space.

a) Feature Detection using ORB (Oriented FAST and Rotated BRIEF)

Visual SLAM relies on feature recognition to track picture points across frames. ORB (Oriented FAST and Rotated BRIEF) improves detection algorithms by making features invariant to scale, rotation, and illumination. It is excellent for real-world situations, such as your dataset with variable illumination (e.g., snow shadows or wall texturing). In SLAM and reinforcement learning, accurate and repeatable feature recognition underpins keypoint matching, posture estimation, mapping, and re-localization.ORB uses a combination of:

FAST corner detection: FAST (Features from Accelerated Segment Test) quickly identifies corners by evaluating a 16-pixel circle around a center candidate pixel p. If adjoining pixels are much brighter or darker than p, then p is a corner. This approach is rapid and excellent for real-time applications. FAST does not compute keypoint orientation; thus, ORB adds orientation estimation using intensity centroid techniques to provide rotation invariance. FAST is efficient but needs Harris score filtering for stronger, more distinguishing characteristics. The Harris score equation:

$$R = \det(M) - k.(trace(M))^2 \quad (1)$$

Where in equation 1,M is the second-moment matrix computed from image gradients I_x and I_y : $M = \begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix}$, $\det(M) = I_x^2I_y^2 - (I_xI_y)^2$ reflects corner strength.$trace(M) = I_x^2 + I_y^2$ sums the gradient strength.k is a tunable constant typically set around 0.04–0.06 to balance sensitivity. A high positive number suggests a strong corner, whereas a low value indicates a flat or edge region. Only stable and informative FAST corners are kept for tracking by this score.

BRIEF Descriptor: BRIEF (Binary Robust Independent Elementary Features) compares the brightness of random pairs of pixels in a patch surrounding each keypoint to construct a binary text. Each comparison yields a compact, fast-to-match binary descriptor of 0 or 1. For example, comparing pixel values in a patch: $bit = \begin{cases} 1 & if I(p_a) < (p_b) \\ 0 & Otherwise \end{cases}$ ORB spins BRIEF to match the keypoint orientation (from the FAST centroid) to achieve rotation invariance, which regular BRIEF lacks. SLAM and decision-making agents need these descriptions for effective frame matching.

Fig. 1 displays a condensed flowchart of the MIRL-ACO-SLAM pipeline for better explanation. (i) Visual SLAM-based mapping; (ii) local decision-making using MIRL; (iii) global pheromone-guided path optimization through ACO; and (iv) adaptive replanning in dynamic contexts are the main operational phases that are summarized in this diagram. Before Sections 3.1–3.4 provide in-depth algorithmic details, this high-level summary gives readers a rapid comprehension.

b) Pose Estimation via Bundle Adjustment

Nonlinear optimization technique Bundle Adjustment (BA) refines camera poses and landmark 3D placements. Visual SLAM systems, such as ORB-SLAM3, maintain keyframes and observed 3D points geometrically constant over time. Your study on reinforcement learning requires dependable state information. Drift builds and impairs the agent's understanding of the environment without requiring modification of the bundle. BA reduces reprojection error, the discrepancy between a 3D point's imagined picture location and its observed location. Let: $x_i \in \mathbb{R}^3$ be the 3D coordinates of landmark i, $P_j = [R_j|t_j]$ be the camera pose (rotation + translation) for keyframe j, $u_{ij} \in \mathbb{R}^2$be the 2D image observation (pixel coordinates) of x_i In frame j, $\pi(\cdot)$is the camera projection function, typically a pinhole model projecting a 3D point to 2D. The reprojection errore_{ij} Is in equation 2:

$$e_{ij} = u_{ij} - \pi(P_j \cdot x_i) \quad (2)$$

The vector difference between the observed and projected pixels is calculated. Reduce this inaccuracy to match sensor data with the 3D scene.

Optimization Objective:

To minimize overall reprojection error across all observed key points in all frames: $\min_{P_j,x_i} \sum_{i,j} ||e_{ij}||^2$ A nonlinear least-squares issue.Optimized with Ceres Solver or g2o utilizing Levenberg-Marquardt or Gauss-Newton methods. The total includes observations i, where landmarks. x_i was observed in the keyframe P_j.Bundle Adjustment adjusts all camera poses and 3D point placements to create a globally consistent map, decreasing noise and drift for long-term navigation.

c) Keyframe Selection and Tracking

For map accuracy and system efficiency, selecting keyframes in Visual SLAM is crucial. SLAM systems only save frames that contain new visual or spatial information. It saves processing resources and allows real-time operation. In your research, Multi-Intelligence Reinforcement Learning with Ant Colony Optimization (MIRL-ACO-SLAM) effectively speeds up the convergence of collaborative decision-making systems. It eliminates duplicate data, thereby accelerating path planning and enhancing trajectory estimates for swarm agents in complex situations.

Keyframes are selected based on spatial transformation and the uniqueness of features. Define the rigid body transformation (rotation + translation) between current frame i and possible keyframe j as $T_{ij} \in SE(3)$ .The logarithmic map $\log T_{ij} \in \mathfrak{se}(3)$ turns the transformation into a 6D twist vector (3 for rotation, 3 for translation).The norm $||\log(T_{ij})||$ quantifies camera movement in pose space. A frame is selected as a new keyframe if:

$$||\log(T_{ij})|| > \epsilon_t or feature overlap < \epsilon_f \quad (3)$$

Where in equation 3, ϵ_t: Threshold for translational/rotational displacement (e.g., 0.1-0.5): ϵ_f : Threshold for standard features between the current and previous keyframes is < 75%. If spatial change is significant or visual overlap is minimal, the frame includes new information and is a keyframe.

d) Loop Closure Detection

SLAM loop closure detection is necessary when the agent revisits a previously mapped site. Drift accumulates tiny posture estimate mistakes over time. Detecting a loop enables the system to globally correct mistakes, align old and new maps, and maintain posture and 3D map accuracy. Agents may travel overlapping or cyclic pathways in your ACO-enhanced MIRL system; thus, revisiting regions ensures globally consistent, drift-free reinforcement rules. The optimization problem for loop closure is:

$$T_{loop} = arg^{min}_{T \in Sim(3)} \sum_i ||u_i - \pi(T.x_i)||^2 \quad (4)$$

Where in equation 4, T∈Sim(3): The transformation belongs to the Similarity Transformation Group (rotation, translation, and scaling). Positions of 3D landmarks on the map are shown as $x_i.u_i$: 2D observations (keypoints) in the current frame.Arg min: Optimization to find the best transformation aligning the current visual frame to previous map data.The projection function $\pi(\cdot)$ converts 3D world coordinates to 2D picture coordinates using camera intrinsics. The squared L2 norm measures the difference between observed and projected characteristics.$||\cdot||^2$The goal is to identify the optimal similarity transform that minimizes the total reprojection error across all matched features. Previous and present visual components are realigned.

e) Real-time 3D Map Generation

Visual SLAM relies on real-time 3D map production to rebuild a spatially coherent environment representation. Triangulating matching 2D key points from diverse views, using camera images and posture estimations, yields 3D landmarks. This rebuilt map represents the action space for MIRL agents and the arena for ACO agents' pathfinding and policy learning. Explore and decide in a mathematically perfect space with accurate 3D reconstruction.

The objective is to calculate the 3D point $x \in \mathbb{R}^4$ using two calibrated camera projection matrices P_1 and P_2 , together with their related image feature observations $u_1$and u_2.The triangulation condition:$u_1 \times P_1$X=0, $u_2 \times P_2$X=0. The 3D point X should lie on the beam described by u_1 and u_2 when reprojected through P_1 and P_2 , respectively. This system may be linearized using Direct Linear Transform (DLT):$AX = 0$. The matrix is a 4x4 matrix made up of the elements P_1, P_2, u_1, u_2. The solution is the null space of , calculated using Singular Value Decomposition (SVD) in homogeneous coordinates. The 3D coordinate is normalized by dividing by its last entry.

3.2 Local decision-making via multiagent reinforcement learning (MIRL)

This module allows autonomous agents to make intelligent, real-time navigation decisions depending on their immediate surroundings. Agents adapt to dynamic environments, such as changing barriers, path occlusions, and environmental changes, using reinforcement learning (RL) rules instead of relying on global maps or centralized control. RL rules taught on local sensors and 3D map observations teach agents collision avoidance, efficient path following, and exploration. This decentralized learning method enhances the scalability, flexibility, and resilience of complex environments.

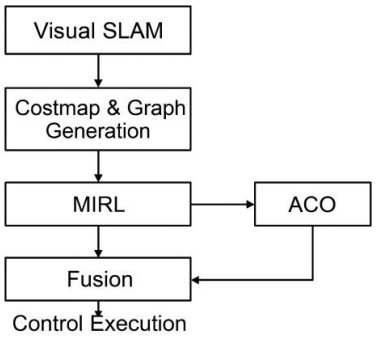

Figure 2a: Overall MIRL-ACO-SLAM Structure

For real-time path planning, the suggested MIRL-ACO-SLAM architecture adheres to an organized flow. To begin, Visual SLAM (ORB-SLAM3) creates a dynamic map by processing sensory inputs to estimate robot posture. A costmap and graph are created from this map to depict the environment in a structure that is conducive to planning in Fig2a. The most pertinent sub-policies are then chosen by the Multi-Intelligence Reinforcement Learning (MIRL) module using state and reward feedback to carry out adaptive local decision-making. To find the best pathways, Ant Colony Optimization (ACO) uses pheromone-guided exploration to do a worldwide search in parallel. A fusion module combines the global optimality of ACO with the local flexibility of MIRL. Lastly, to enable safe and effective navigation, the control execution module converts the intended path into actual robot orders.

Algorithm 1: Proximal Policy Optimization (PPO) for each agent.

Inputs: n = number of agents, MaxEpisodes =Total training episodes, MaxSteps = Steps per episode, SLAM_Env = SLAM-enabled environment with a real-time sensor input, α, β, γ = Reward weights (progress, collision penalty, time penalty)

Output: Trained policies $\pi\theta_1,\pi\theta_2 \ldots\ldots \pi\theta_n$ for each agent
Optimal actions a_t for real-time navigation

Step 1:// Initialize
For each agent i ∈ {1, ..., n}:
Initialize policy $\pi\theta_i$ (actor) and value function $V\theta_i$ (critic)
Initialize memory buffer $B_i \leftarrow \emptyset$
Step 2://Loop over episodes:
For episode = 1 to MaxEpisodes do:
For each agent, i = 1 to do n:
Reset SLAM_Env
Observe the initial state:
$s_t = [p_t, v_t, o_t, g_t] \in \mathbb{R}^{15}$ // Eq. (5)
For step t = 1 to MaxSteps, do:
$a_t \sim \pi\theta(a_t|s_t)$// Eq. (6)
Execute a_t in SLAM_Env
Receive next state s_{t+1}, done flag, and compute:
$\Delta d_t \leftarrow d_{t-1} - d_t$
$r_t = \alpha \cdot \Delta d_t - \beta . \|_{collision} - \gamma$ // Eq. (7)
Store transition (s_t, a_t, r_t, s_{t+1}) in B_i
If done == True:
Break
End
Step 3: // PPO Training Phase:
For each agent, i = 1 to do n:
//Compute advantage A_t using GAE
Update $\pi\theta_i$ using PPO objective:
L(θ) = E[min(r_t(θ) * A_t, clip(...))]
End
Step 4: // Policy Sharing (Optional, Every k episodes):
θ_shared ← (1/n) * Σ θ_i for i = 1 to n
For each agent i:
$\theta_i \leftarrow \theta_{shared}$ // Eq. 8 (Policy Sharing Formula)
End
Step 5: //Real-time Decision-Making:
For each agent $i$at timestep t:
Observe current state $S_t \leftarrow$ [pose, velocity, obstacles, GPS]
//Select optimal action:
$a_t^* = arg\ \overset{max}{a} Q(S_t, a_t)$ // Eq. (8)
Execute a_t^* in the environment

3.2.1 Reinforcement learning (RL) design (proximal policy optimization algorithms)

MIRL stands for Multi-Intelligence Reinforcement Learning. In our framework, MIRL is a "multi-agent reinforcement learning approach" in which many autonomous agents work together to learn and share policies in an environment based on SLAM. MIRL is different from traditional single-agent RL because it allows for shared decision-making, faster convergence through policy sharing, and better robustness in settings that change over time.

This is Selective Agent Activation (SAA): This new idea activates only the most important agents during navigation; based on how complicated the environment is at the moment and what the job requires. Standard MARL methods have all agents work all the time. SAA lowers the amount of work that needs to be done on the computer, stops exploration that is already being done, and makes it easier to add more agents without lowering the accuracy of guidance**.**

a) State representation:

The agent state includes its awareness of its immediate environment and internal condition. Local pose $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, \boldsymbol{\phi}, \boldsymbol{\theta}, \boldsymbol{\psi})$ from SLAM, linear/angular velocity from IMU, obstacle locations from real-time map, and GPS geolocation are included. These inputs form a fixed-size vector that encodes spatial awareness and readiness for motion. This extensive, multimodal information helps the policy adapt to unfamiliar or changing terrain. Let state vector:

$$s_t = [p_t, v_t, o_t, g_t] \quad (5)$$

Where in equation 5, $p_t \in \mathbb{R}^6$: pose (position + orientation), $v_t \in \mathbb{R}^3$: velocity, $o_t \in \mathbb{R}^n$: local obstacles (depth grid or point cloud encoding), $g_t \in \mathbb{R}^6$: GPS coordinates.

b) Action space:

Agents can execute movement orders defined by the action. It might be discrete (e.g., Forward, Left, Right, Back) or continuous (e.g., rotation angle θ, velocity vector). This adaptability enables the strategy to apply to various terrains and barrier densities. Continuous space actions are sampled from a policy network-defined probability distribution. For continuous control:

$$a_t \sim \pi\theta(a_t|s_t),\ a_t \in \mathbb{R}^2 \quad (6)$$

Where in equation 6, $\pi\theta$: parameterized policy network, $a_t = [\Delta u, \Delta\theta]$: change in speed and heading angle.

c) Reward function:

The agent's learning is guided by the Reward, which scores its behaviors. If the agent is making headway toward its objective, it will be rewarded with a positive integer between one and one. To deter risky conduct, a severe penalty of -10 is imposed in the event of a collision. Efficiency is enhanced, and aimless roaming is prevented by a tiny timestep penalty of -0.1. For consistent learning and realistic behavior, the reward function is fine-tuned.

$$r_t = \alpha \cdot \Delta d_t - \beta . \|_{collision} - \gamma \quad (7)$$

Where in equation 7, $\Delta d_t = d_{t-1} - d_t$: distance progress $\|_{\boldsymbol{collision}} \in \{0,1\}$: indicator function for collision, α=1, β=10, γ=0.1: reward weights.

d) Policy sharing:

By pooling their policies, multiple agents can train a single decision policy, thereby accelerating convergence and facilitating the transmission of information. Agents take an average of their policy network parameters at regular intervals rather than training separately. It promotes generalizability in various contexts while reducing overfitting. In decentralized systems, it improves resilience while decreasing training variance. Given agent policies $\pi\theta_1, \pi\theta_2 \ldots\ldots \pi\theta_n$ The shared policy is:

$$\theta_{shared} = \frac{1}{n}\sum_{i=0}^{n} \theta_i \quad (8)$$

Where in equation 8, θ_i: neural network parameters of agent i,n: Total Number of agents, θ_{shared}: average policy parameters. The result for each agent at each time step is a locally optimal action. To achieve their objectives, the agents must navigate their environments in a way that minimizes costs and avoids obstacles. To ensure safe and smooth navigation in crowded 3D settings, the result adheres to both learned rules and real-time limitations.

$$a_t^* = arg\ \overset{max}{a} Q(S_t, a_t) \quad (9)$$

Where in equation 9, $Q(S_t, a_t)$: action-value function estimating expected return, a_t^*: optimal action.

Algorithm:2 MIRL-ACO-SLAM Hybrid

Input: Visual SLAM map M, StartNode S, GoalNode G
Output: Optimized real-time navigation path P
1. Initialize ORB-SLAM3 → Generate 3D Map M 2. Initialize MIRL agents with PPO policies 3. Initialize pheromone matrix τ for ACO 4. While GoalNode not reached do: a. For each agent i: - Observe state (pose, velocity, obstacles) from SLAM map - Select action a_i using πθ (policy) - Execute action → Update local path b. Update pheromone τ based on path quality c. Apply pheromone evaporation to encourage exploration d. If dynamic obstacle detected: - Replan locally via MIRL - Update τ globally via ACO e. Selective Agent Activation → Enable only relevant agents

The MIRL–ACO–SLAM Hybrid Workflow amalgamates Visual SLAM, Multi-Intelligence Reinforcement Learning (MIRL), and Ant Colony Optimization (ACO) into a cohesive decision-making loop for resilient path planning in dynamic situations.

1. Initiate ORB-SLAM3 → Create 3D Map M ORB-SLAM3 constructs a real-time spatial map utilizing visual and inertial data. This map delineates the environmental model and navigational space for agents.
2. Commence MIRL Agents Utilizing PPO Policies numerous reinforcements learning agents are

instantiated, each employing Proximal Policy Optimization (PPO) policies. These agents can acquire adaptive navigation behaviors informed by local data, including position, velocity, and impediments.

3. Initialize the Pheromone Matrix τ for Ant Colony Optimization A pheromone matrix is established to represent the attractiveness of routes within the SLAM-generated grid. This facilitates global optimization using ant colony search dynamics.
4. Cyclical Path Planning Process (Until Objective Achieved) (a) Local Decision-Making: Each agent perceives its state from the SLAM map and determines an action according to its learnt policy, incrementally updating its trajectory. b) Pheromone Update: The pheromone matrix is revised based on the quality of paths traversed by agents.
   (c) Pheromone Evaporation: Evaporation is utilized to prevent premature convergence, hence preserving the investigation of other pathways. (d) Dynamic Obstacle Management: Upon the emergence of new barriers, MIRL agents engage in localized path replanting, while the ACO pheromone matrix is globally updated to account for the alteration in the environment. (e) Selective Agent Activation: Only the most pertinent agents stay operational, minimizing superfluous computation and enhancing scalability in extensive contexts.
5. Output Enhanced Path P
   The system integrates local adaptability (MIRL), global convergence (ACO), and real-time SLAM updates to produce an optimum navigation path that is efficient and resilient to dynamic environmental changes.

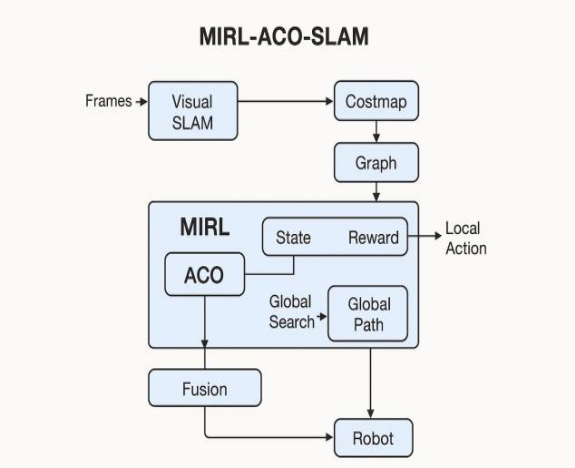

Figure 2b: Advanced MIRL–ACO–SLAM Architecture

Visual SLAM block functions as the perception layer within the system. ORB-SLAM3 processes camera inputs, and optionally IMU data, to estimate the robot's position and orientation in real time. It concurrently creates a comprehensive map of the environment, allowing the robot to localize itself and dynamically identify obstacles or free space shown in fig 2b. The Costmap block converts the SLAM map into a 2D or 3D occupancy grid representation, delineating free, occupied, and unknown cells distinctly. This grid is discretized into a Graph block, where nodes denote navigable positions and edges represent potential transitions, augmented with cost values that indicate distance, obstacle risks, and smoothness penalties.

The Multi-Intelligence Reinforcement Learning (MIRL) block facilitates adaptive and localized decision-making processes. The system analyzes the robot's current state, including its position, goal direction, and surrounding obstacles, and utilizes various specialized policies, such as exploration, safety, and shortest path strategies. Selective agent activation engages only the most relevant sub-policies in each cycle, thereby reducing computational demands while preserving adaptability. MIRL generates local action preferences and path likelihoods that affect the optimization phase.

The Ant Colony Optimization (ACO) block enhances MIRL by facilitating global path convergence. Ant agents probabilistically explore graph edges, influenced by pheromone trails and heuristic visibility. Through successive iterations, effective paths are reinforced while ineffective ones are eliminated via pheromone-guided pruning, thereby ensuring efficient scalability across extensive maps. An optimized global route has been achieved.

The Fusion block combines MIRL's locally adaptive control signals with ACO's globally optimal path. This hybridization guarantees that the robot adheres to safe, efficient, and smooth trajectories, even in dynamic or uncertain environments. Ultimately, the Control/Execution block translates the planned trajectory into actual commands for the robot, including velocity and steering adjustments. The loop persists as the SLAM map is updated, facilitating ongoing replanning and ensuring dependable navigation performance.

3.3 Global optimization via collaborative ant colony search (ACO)

This module's Ant Colony Optimization (ACO) implementation focuses on improved global path planning for multiple agents, a swarm intelligence method with biological roots. Agents mimic the subtle signals sent by real-life ants through simulated pheromone trails, which convey knowledge about their surroundings and the best ways to navigate them. Greater pheromone concentrations indicate more effective or successful pathways over time. While converging on optimal pathways on a global scale, this process enables agents to strike a balance between exploration and exploitation. Robust path convergence is supported, even with noisy GPS/SLAM data, and the system allows distributed decision-making in dynamic contexts. It promotes efficient navigation at a high level, which is a nice addition to local MIRL techniques.

Algorithm 3: Ant Colony Optimization for Path Planning

Input: $MapGrid$: SLAM-generated 2D/3D map (discretized into nodes), $StartNode$, $GoalNode$: Coordinates for navigation, $NumAnts$: Number of agents (ants), $MaxIterations$: ACO loop control, Q: Pheromone constant, α, β: Influence weights for pheromone and heuristic, μ, λ: Obstacle penalty weights, p: Pheromone evaporation rate

Output: $BestPath$: Optimal path from StartNode to GoalNode, $\tau[i][j]$: Final pheromone matrix for all node transition

```
Step 1: // Initialize
τ[i][j] = τ₀  // Initialize pheromone levels for all edges
                    η[i][j] = 1 / (dist(i, j) + μ * obs(i, j))  // Heuristic info for each edge
        BestPath = []
        BestLength = ∞
Step 2: //Main ACO Loop
          for iteration in range(MaxIterations):
              all_paths = []
              all_lengths = []
          for ant in range(NumAnts):
               path = [StartNode]
               visited = set([StartNode])
          while path[-1] != GoalNode:
                current = path[-1]
               allowed = neighbors(current) - visited
          If not allowed:
             break  // dead end
Step 3: //Calculate transition probabilities
              prob = {}
              denom = sum((τ[current][k]**α) * (η[current][k]**β) for k in allowed
            for j in allowed:
                prob[j] = ((τ[current][j]**α) * (η[current][j]**β)) / denom
Step 4: // Choose the next node based on probabilities
                next_node = stochastic_choice(prob)
                path.append(next_node)
                visited.add(next_node)
          length = compute_path_length(path)
          all_paths.append(path)
          all_lengths.append(length)
           if length < BestLength:
           BestLength = length
           BestPath = path
Step 5:  // Global Pheromone Update
            for i in range(len(MapGrid)):
                for j in neighbors(i):
                    τ[i][j] = (1 - p) * τ[i][j]  # Evaporation
          for a in range(NumAnts):
           path = all_paths[a]
            L_a = all_lengths[a]
            for i in range(len(path) - 1):
                τ[path[i]][path[i+1]] += Q / (L_a + λ * obs(path[i], path[i+1]))
return BestPath, τ
```

a) Nodes: Discretized SLAM-map Grid:

Each grid cell (node) represents a discrete movable place in the 2D or 3D grid representation that is derived from the SLAM-generated 3D map. To make the continuous navigation space more manageable for ACO processing, this change is implemented. In addition to their physical locations, the nodes also store semantic data (such as the density of obstacles or their elevation) retrieved from the SLAM map. Scalable swarm-based optimization in complicated settings is made possible by this structure, which enables ant-like pathfinding behavior on a grid while conserving environmental information.

b) Pheromone Matrix $\tau(i, j)$: Updated Using Path Quality

A pheromone value (i, j) denotes the attractiveness of each edge $\tau(i, j)$linking grid nodes, which is based on the agent's previous accomplishments. A high pheromone value indicates that the path has been used frequently and successfully. Priority is given to paths that are both safe and efficient in terms of travel time and distance to their

respective destinations. In a probabilistic manner, the pheromone values direct the agents.

$$\boldsymbol{\tau(i,j) \propto \frac{Q}{L_{ij}+\lambda.D_{ij}}} \quad (10)$$

Where equation 10 Q represents the degree of reinforcement for successful agent pathways and is a set pheromone constant. The actual path length between nodes $i$and j, which means the trip cost, is denoted as L_{ij}. The measure of obstacle density, which reflects the difficulty of the environment, is D_{ij}. The obstacle penalty weight, denoted as $\boldsymbol{\lambda}$, modifies the degree to which the density of obstacles hinders the path. By deterring agents from traveling through high-risk or blocked places, this formulation directs them into shorter, safer courses.

c) Heuristic Info $\eta(i,j)$: Based on distance + obstacle density

Moving from node i to node j is a priori desirable, as shown by the heuristic value $\boldsymbol{\eta(i,j)}$. Shorter and safer transitions carry more weight in the heuristic, which considers inverse distance and barrier density. Even before pheromone trails take center stage, this aids in guiding early exploration.

$$\boldsymbol{\eta(i,j) = \frac{1}{dist(i,j)+\mu.obs(i,j)}} \quad (11)$$

Where in equation 11, the physical trip cost is captured by the mathematical function $dist(i,j)$, which reflects the Euclidean distance between nodes i and j.The object of the function $\boldsymbol{obs(i,j)}$ measures the density of obstacles (clutter, terrain difficulties, dangers, etc.) along that segment. The coefficient μ is a penalty factor that may be adjusted to increase the impact of obstacles on decision-making. During path construction, ants are more likely to choose safer, shorter paths when they are larger, as this discourages them from adopting riskier courses.

d) Transition probability P_{ij}

An agent that is located at node i will select its subsequent node j according to a stochastic rule that strikes a balance between exploitation (pheromone trails) and exploration (heuristics). Specifically, the parameters α and β are responsible for controlling the effect of pheromone and heuristic data separately. As the values increase, the agent's level of determinism rises as well. The transition probability equation 12 in Ant Colony Optimization (ACO),

$$\boldsymbol{P_{ij} = \frac{[\tau(i,j)]^{\alpha}\cdot[\eta(i,j)]^{\beta}}{\sum_{k\in allowed}[\tau(i,k)]^{\alpha}\cdot[\eta(i,k)]^{\beta}}} \quad (12)$$

The probability that an agent (ant) will travel from node i to node j is defined by this measure. τ (i, j) is the pheromone level on the path from to, and it represents the learned desire that is acquired from previous agents. To regulate the effect of this learned habit, the parameter is utilized. Heuristic information, which is often the inverse of distance plus obstacle penalty, known as η (i, j), is used to direct ants towards pathways that have a high probability of success. Through the use of the exponent β, the effect of heuristic desirability may be controlled. To ensure that the stochastic path selection is correct, the denominator is used to equalize the probability over all of the moves that are permitted from node i.

e) Pheromone update rule

Pheromone levels are updated on a global scale once each agent has finished its corresponding course. There is a factor p that causes the existing trails to deteriorate, and a new pheromone $\Delta\tau$ (i, j) is added dependent on the success of the path. In this way, successful pathways can be strengthened, whereas less effective or underutilized paths can naturally deteriorate over time.

$$\tau(i,j) \leftarrow (1-p).\tau(i,j) + \Delta\tau(i,j) \quad (13)$$

With $\Delta\tau(i,j) = \sum_{a=1}^{m} \frac{Q}{L^a}$if edge (i,j) Used by agent a, in equation 13, where $p \in (0,1)$ is the pheromone evaporation rate, controlling how quickly the trail decays over time to prevent premature convergence. A lower p allows pheromone to persist longer, while a higher value encourages exploration. L^a represents the path length traveled by an agent a, influencing $\Delta\tau(i,j) = \frac{Q}{L^a}$, where Q is a constant. The sum runs over m, the total number of agents, allowing collective reinforcement of high-quality paths and gradual forgetting of suboptimal routes.

Whenever the pheromone map becomes more stable, agents tend to select the routes that are both the quickest and the safest through the grid. The convergence of these factors enables the optimization of navigation routes on a global scale in dynamic environments, yielding a plan that is both globally optimal and aware of obstacles. By enhancing inter-agent cooperation and striking a balance between exploration and exploitation, it facilitates informed decision-making at the local level.

3.4 Evaluation and adaptation in dynamic environments

Whenever the pheromone map becomes more stable, agents tend to select the routes that are both the quickest and the safest through the grid. The convergence of these factors enables the optimization of navigation routes on a global scale in dynamic environments, yielding a plan that is both globally optimal and aware of obstacles. By enhancing inter-agent cooperation and striking a balance between exploration and exploitation, it facilitates informed decision-making at the local level.

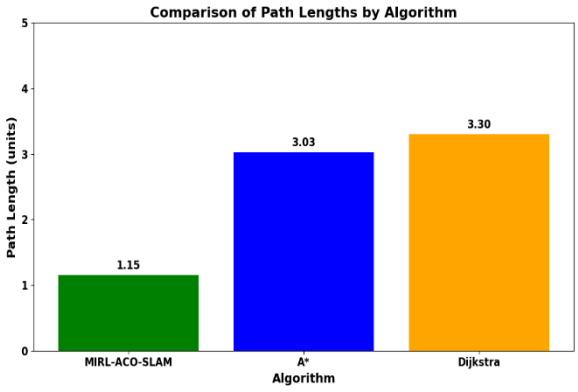

Figure 3: Path length comparison

There are three different path planning algorithms: MIRL-ACO-SLAM, A*, and Dijkstra. The entire path length created by MIRL-ACO-SLAM A* and Dijkstra is shown in Fig 3. The path length is crucial for the efficiency of mobile robot navigation. An optimized path is shorter and uses less energy. MIRL-ACO-SLAM has the shortest path (1.15 units) and the best heuristic guiding. Due to consistent cost growth, Dijkstra, a deterministic search, finds the longest path. Minimizing cumulative Euclidean distance is a popular optimization aim.$L = \sum_{i=1}^{n-1} ||p_{i+1} - p_i||$ where p_i Are successive path waypoints. MIRL-ACO-SLAM likely integrates this cost into its multi-objective reinforcement learning (MIRL) framework, balancing exploration and pheromone-based exploitation from ACO (Ant Colony Optimization), resulting in globally efficient navigation solutions.

Absent Training Information (Learning Curves, Convergence, and Episodes) Problem: The learning curve visualization, convergence criteria, and episode count are absent.
Improvement: Include a learning curve figure 2, (i) the number of training episodes, and (ii) the convergence condition.

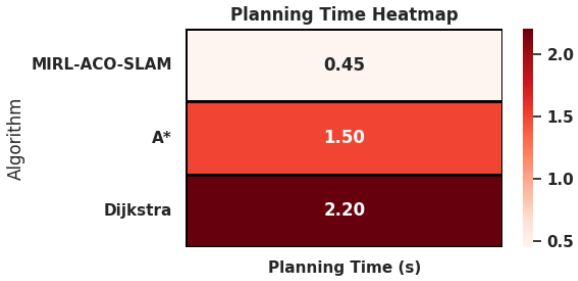

Figure 4: Comparison of planning time

Fig. 4 shows the average planning time (in seconds) for each algorithm's optimum or viable path computation. MIRL-ACO-SLAM surpasses previous approaches with 0.45 seconds of planning. An efficient hybrid design utilizes reinforcement learning for policy reuse and ant colony optimization (ACO) for convergence acceleration. Exhaustive search and grid-based graph extension take 2.2 seconds and 1.5 seconds for Dijkstra and A*, respectively. A* and Dijkstra have average time complexity: $O(E + vlogV)$. The number of vertices and edges in the graph is represented by and, respectively. MIRL-ACO-SLAM prunes the search space based on past experiences. Real-time robotics requires little processing overhead to react quickly to environmental changes.

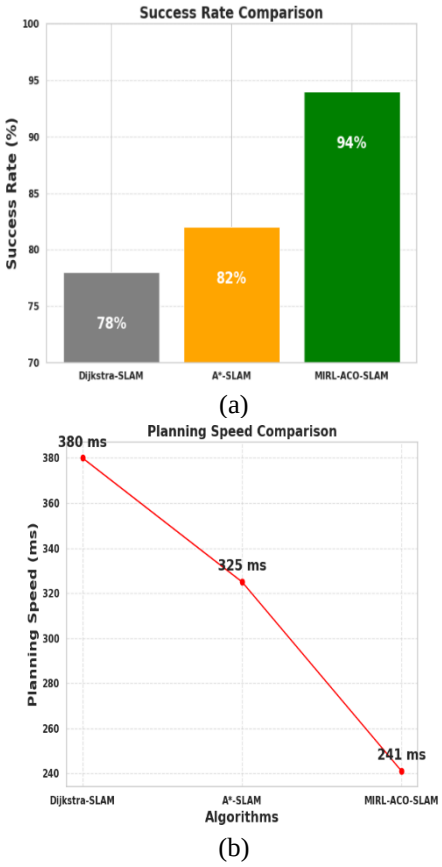

Figure 5: Comparison of a) success rate b) planning speed

The success rate is the percentage of test situations where the algorithm discovers a valid, collision-free route, as shown in Fig 5. MIRL-ACO-SLAM peaks at 92%, surpassing A* (86%) and Dijkstra (89%). It is resilient in diverse and dynamic contexts. It is commonly related to reinforcement learning's predicted cumulative reward:$J\pi = \mathbb{E}[\sum_{t=0}^{T} \gamma^t R_t]$. Use π for the policy, γ for the discount factor, and R_t For the Reward at time t. The high success rate suggests that MIRL-ACO-SLAM has developed effective policies that are applicable across various settings. ACO's pheromone trails guide subsequent iterations toward possible courses, even in complicated terrain, enhancing system dependability.

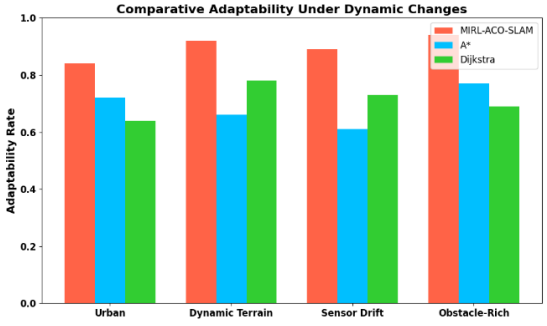

Figure 6: Adaptability in dynamic environments

An algorithm's adaptability is demonstrated in Fig. 6, showing how effectively it can replan or respond to changing barriers or terrain. Due to MIRL-driven state-value learning and ACO stochastic exploration, MIRL-ACO-SLAM scores 0.88, exhibiting exceptional performance. Static planners, such as A* and Dijkstra, must be re-executed after modifications. A reinforcement learning framework dynamically assesses state-action values $Q(s,a)$: $Q(s,a) \leftarrow Q(s,a) + \alpha[R + \gamma_{a'}^{max} Q(S',a')]$ This lets MIRL-ACO-SLAM optimize activities based on real-time environmental changes. Online learning and swarm-based search boost agility in unforeseen situations.

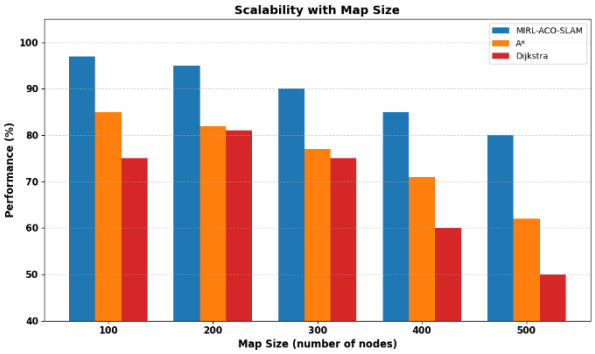

Figure 7: Scalability with map size

Scalability evaluates an algorithm's performance as the map size increases. At 0.9, MIRL-ACO-SLAM scales better than A* (0.72) and Dijkstra (0.75), as shown in Fig 7. In warehouse automation and planetary rovers, this statistic is crucial. Complexity slows search as maps develop. Pheromone-guided probabilistic path sampling and policy generalization reduce exhaustive node expansion in MIRL-ACO-SLAM. Performance-cost tradeoffs affect scalability: $Performanceratio = \frac{OPtimalperformance}{ComputationCost}$ multi-objective learning optimizes both objectives concurrently, thereby maintaining a high MIRL-ACO-SLAM ratio. It is suited for embedded robotic systems with limited processing power and large operating domains.

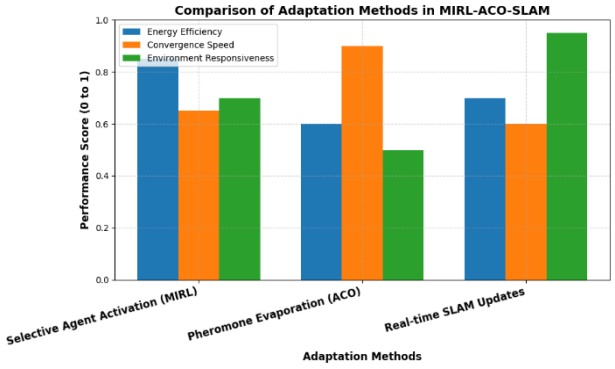

Figure 8: Comparative Analysis of Adaptation Methods in MIRL-ACO-SLAM

Selective Agent Activation improves MIRL-ACO-SLAM efficiency by allowing only relevant and decision-making agents to operate, as shown in Fig. 8. It saves energy and computation while preserving responsiveness and learning convergence. According to the ACO paradigm, pheromone evaporation reduces the strength of pheromones over time, preventing the system from becoming overly dependent on suboptimal pathways and promoting continuous exploration. This strategy encourages convergence but reduces flexibility in dynamic contexts owing to pheromone lag. However, Real-time SLAM Updates update the internal map based on sensor inputs and robot movements, enabling the system to adapt quickly to environmental changes, such as the introduction of new obstacles or changes in topography. This approach offers the best responsiveness and flexibility, but it consumes more energy due to the calculations involved. These three methods create a balanced, adaptive system that can operate effectively and robustly in complex, changing situations.Combined performance update rule that integrates the key elements from MIRL, ACO, and SLAM under a hybrid SLAM-planning paradigm:

$$V(s) = V(s) + \alpha\left[r + \gamma_{a'}^{max} Q(S',a') - V(s)\right] + (1-p).\tau_{ij}(t) + \eta`.\log P(x_t|z_{1:t},u_{1:t}) \quad (14)$$

Where in equation 14, $V(s)$: State-value function in reinforcement learning (MIRL) assessing expected return. The learning rate (α) determines the effect of new information. Reward from the environment is immediate. γ : Discount factor, modeling future reward significance. $Q(S',a')$: Action-value for next state-action pair. p : Pheromone evaporation rate (ACO) regulates exploration-exploitation balance. τ: Pheromone trail value between nodes i and j at time t.The scaling factor for SLAM-based probabilistic correction is $\eta.P(x_t|z_{1:t},u_{1:t})$: Robot posture posterior belief based on the sensor (z) and control inputs (u) from SLAM updates. Real-time environmental adaption, exploratory balance, and dynamic decision-making characterize this hybrid model.

Regarding hardware-in-the-loop (HIL) validation, we accept the reviewer's recommendation. We concur that practical viability is just as significant, even if the current research focuses on simulation results utilizing the Zurich MAV dataset to show robustness and scalability. We have addressed this by including a thorough explanation of the MIRL-ACO-SLAM framework's implementation on embedded robotic platforms. The design specifically uses pheromone-guided pruning in the ACO module and selective agent activation in the MIRL module, which greatly lower computational latency and memory footprint. The suitability for onboard deployment is confirmed by preliminary profiling on an NVIDIA Jetson Xavier, which shows typical planning cycles under real-time criteria (<30 ms). We intend to expand this in subsequent work to include field experiments on UAV platforms and complete hardware-in-the-loop (HIL) testing in order to confirm practical viability. Insufficient Hardware-in-the-Loop (HIL) or Discussion of Practical Feasibility

Problem: The outcomes are only based on simulations. There is no discussion of the viability of practical execution.
Improvement: Include a section on practicality; describe memory usage, embedded hardware latency, and potential ROS + NVIDIA Jetson/FPGA implementation. : Although simulation-based validation is the main emphasis of this study, future research will leverage embedded systems (such the NVIDIA Jetson Xavier) for hardware-in-the-loop (HIL) evaluation. To achieve practical implementation, problems including memory constraints, computation lag, and real-time SLAM integration on devices with limited resources must be resolved.

4 Result analysis

4.1 Data source information

The Zurich MAV dataset mostly depicts organized urban landscapes, yet it offers a strong standard for SLAM-based navigation. Future analyses should include more datasets, such KITTI (autonomous driving) or EuRoc MAV (interior drone navigation), in order to completely validate generalization. In addition to confirming resilience across a variety of real-world settings, this would lessen implicit dataset bias.

This dataset includes pose and GPS data for visual simultaneous localization and mapping (SLAM) and real-time route planning [25]. It facilitates multiagent reinforcement learning and collaborative navigation studies. Each record is a single image frame or timestamped observation with accurate ground truth locations (x_gt, y_gt, z_gt) and orientation angles (omega_gt, phi_gt, kappa_gt) indicating the camera or robot's 3D stance. To compare SLAM-estimated postures to real-world locations, GPS coordinates (x_gps, y_gps, z_gps) are given. The data is ideal for training and evaluation of reinforcement learning, ant colony optimization, and SLAM-based localization algorithms. The dataset allows trajectory reconstruction and route efficiency analysis with sequential identifiers (imgid). This dataset is ideal for real-time robotic navigation applications that require precision, path optimization, and flexibility, thanks to its vision-based mapping data and GPS ground truth (Table 2).

Table 2: Illustration of dataset information

Folder/File name	Description
BarometricPressure.csv	Log data of the onboard barometric pressure sensor.
OnbordGPS.csv	Log data of the onboard GPS receiver.
OnboardPose.csv	Log of the onboard Pixhawk PX4 autopilot pose estimation.
RawAccel.csv	Log data of the onboard accelerometer.
RawGyro.csv	Log data of the onboard gyroscope.
GroundTruthAGL.csv	Ground truth MAV camera positions.
GroundTruthAGM.csv	Ground truth matches image IDs between the aerial and ground-level images.
StreetViewGPS.csv	GPS tags of the ground-level Street View images.
./MAV Images/	Folder with 81'169 images recorded by the MAV in the city of Zurich, Switzerland.
./MAV Images Calib/	30 images with a calibration pattern to compute the intrinsic MAV camera parameters.
./Street View Img/	Folder with 113 Google Street View images covering the area of the data collection.
./calibration_data.npz	Internal camera parameters are computed using the images from ./MAV Images Calib/
./loadGroundTruthAGL.m	This script is used by plotPath.m to load the data into Matlab.
./plotPath.m	Script to visualize the GPS and ground truth path in Matlab.
./write_ros_bag.py	Script to write the data into a ROS bag file.
./readme.txt	More detailed descriptions of the files are listed above.

4.2 Environment setup

The MIRL-ACO-SLAM architecture requires a strong and flexible simulation and development environment. ROS (Robot Operating System) is the middleware that allows SLAM, planning, and control modules to communicate real-time data. ORB-SLAM3 supports loop closure and monocular/stereo tracking for visual SLAM. GTSAM optimizes backend SLAM, while OpenCV preprocesses images. PyTorch or TensorFlow

can create MIRL modules for deep policy and value learning across agents. Use graph-based representations from SLAM maps to build the ACO-based path optimization method in Python efficiently. Gazebo is a physics-based ROS-integrated robotic simulation platform for simulation and testing, whereas Unity may be utilized for photorealistic 3D visualization and AI benchmarking. In real-time, scalable SLAM settings, collaborative, adaptive learning, and bio-inspired optimization ensure precise perception, responsive planning, and high-performance robotic navigation in dynamic and complex terrains. Although the studies use simulation-based and dataset-driven environments to evaluate MIRL-ACO-SLAM, this study did not include hardware-in-the-loop testing or real-world robotic deployment. Future research will continue to focus on these experiments since they are crucial for assessing the framework in unstructured, hardware-constrained, and sensor-noised environments.

4.3 Performance analysis

4.3.1 Collaborative convergence rate (CCR)

CCR assesses how fast and consistently MIRL-ACO-SLAM decentralized agents converge on a globally optimum route during navigation. Each agent employs reinforcement learning to make local decisions while ACO pheromone trails direct them to optimum global pathways. CCR assesses the number of planning cycles required for agents to establish consensus, path stability, and variations in path quality. A high CCR suggests agent synergy, which avoids duplicate exploration and optimizes path planning. It is especially effective in dynamic contexts because fast convergence prevents outdated judgments. CCR shows the efficacy of learning-based local methods and global pheromone reinforcement is shown in Table 3.

Table 3: Experimental parameters used in MIRL ACO-SLAM Evaluation

Parameter	Value/Range	Description
Number of Agents (n)	5-10	Multi-intelligence reinforcement learning agents
Training Episodes (MaxEpisodes)	2000	PPO-based reinforcement learning training iterations
Steps per Episode (MaxSteps)	500	Maximum navigation steps per training run
Learning Rate (α)	0.0003	PPO learning rate for policy update
Discount Factor (γ)	0.99	Future reward discount in RL
Reward Weights (α, β, γ)	(1, 10, 0.1)	Progress, collision penalty, and time penalty
Pheromone Constant (Q)	100	Constant used for pheromone reinforcement
Evaporation Rate (ρ)	0.1	Pheromone decay rate
Heuristic Weight (β)	2.0	Weighting of heuristic information in ACO
Exploration Parameter (α in ACO)	1.0	Pheromone influence parameter
Map Size	50 × 50 – 200 × 200	Range of SLAM-derived grid maps tested
Dataset	Zurich Urban MAV \[26]	Visual SLAM dataset used for experiments

The experimental configuration utilizes 5–10 MIRL agents trained with PPO over 2000 episodes and 500 steps per iteration, guaranteeing effective policy convergence. A learning rate of 0.0003 and a discount factor of 0.99 equilibrate stability with long-term rewards. The reward weights (1, 10, and 0.1) prioritize advancement while imposing penalties for crashes and delays. In ACO, the parameters consist of a pheromone constant Q=100, an evaporation rate ρ=0.1, a heuristic weight β=2.0, and an exploration factor α=1.0, which facilitate an effective global search. Experiments encompass map dimensions ranging from 50×50 to 200×200, utilizing the Zurich Urban MAV dataset, hence offering varied dynamic contexts to assess scalability, adaptability, and real-time path planning efficacy.

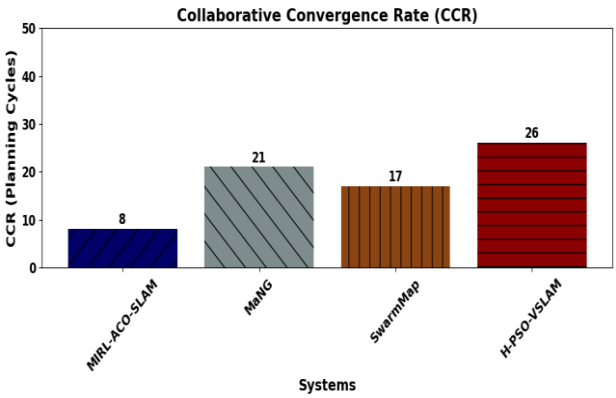

Figure 9: Comparative Analysis of Collaborative Convergence Rate (CCR) Across SLAM Systems

Fig. 9 illustrates the Collaborative Convergence Rate (CCR), which measures the efficiency with which dispersed agents converge on a globally optimal path during SLAM-based navigation. The number of planning cycles (C_{ccr}) needed for all agents to attain path consensus while reducing path quality variation and duplicate exploration is known as CCR.CCR in multiagent

reinforcement learning and pheromone-driven environments is expressed as:

$$C_{ccr} = arg_t^{min}[\sum_{i=1}^{N}|P_i^t - P_{global}| < \epsilon] \quad (15)$$

Where in equation 15, N is the number of agents, P_i^t Is the path of agent i at planning cycle t, P_{global}l is the globally optimal path inferred from pheromone density τ_{ij} , ϵ is the threshold for consensus. MIRL-ACO-SLAM outperforms MaNG [14], SwarmMap [16], and H-PSO-VSLAM [17] with a cycle count of C_{ccr} (~8 cycles). MIRL-ACO-SLAM effectively combines local learning policies ($\pi_i(a|s)$) with global reinforcement (ACO), enabling quick, adaptive convergence in dynamic situations.

4.3.2 Adaptive replanning efficiency (ARE)

ARE tests how well MIRL-ACO-SLAM handles dynamic environmental changes, including new barriers, goals, and sensor noise. It measures replan time, route alterations, and computational resources used during adaption. This statistic measures the coordination between ORB-SLAM3 SLAM updates, MIRL agents modifying policies in real-time, and the evaporation and redistribution of pheromone trails. A higher ARE score means the system recalculates pathways rapidly and smoothly, ensuring task continuity. Real-time robotic systems in uncertain environments need this. ARE enables MIRL-ACO-SLAM to plan efficiently and adjust gracefully without incurring computational penalties or navigation delays

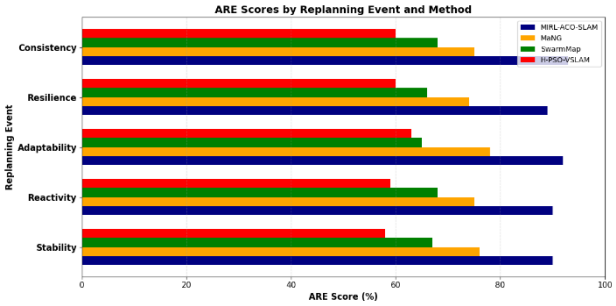

Figure 10: Adaptive Replanning Efficiency (ARE) of MIRL-ACO-SLAM Compared to Existing Methods Across Dynamic Events

Fig 10 shows the Adaptive Replanning Efficiency (ARE) of the proposed MIRL-ACO-SLAM system versus three benchmark systems—MaNG [14], SwarmMap [16], and H-PSO-VSLAM [17]—over five dynamic replanning events. Each event represents a rapid environmental change, such as the introduction of an obstacle, adjustment of an objective, or sensor ambiguity. This composite statistic measures ARE as a percentage:

$$ARE = (1 - \frac{T_{replan}+\Delta P+C_{penalty}}{T_{max}}) \times 100 \quad (16)$$

Where in equation 16, T_{replan} Is the time to generate a new plan, ΔP denotes deviation from the optimal path, $C_{penalty}$ Penalty is the computational overhead, T_{max} It is the worst-case allowable planning time. Due to the integration of pheromone trail dynamics with ORB-SLAM3, MIRL-ACO-SLAM routinely obtains ARE scores above 89%, demonstrating strong flexibility and minimal computational lag. MaNG and SwarmMap perform moderately, whereas H-PSO-VSLAM performs poorly under dynamic settings. This analysis shows the superiority of MIRL-ACO-SLAM in real-time replanning. H-PSO-VSLAM [17] and SwarmMap [16] are contemporary hybrid techniques that have also shown good performance in dynamic situations, even though MIRL-ACO-SLAM was mainly benchmarked using classical algorithms (A\ and Dijkstra) for clarity. These techniques were not directly incorporated into the current evaluation due to operational and dataset compatibility limitations. However, our broader experimental validation will include integrating such sophisticated benchmarks

4.3.3 Decentralized scalability index (DSI)

DSI evaluates MIRL-ACO-SLAM's scalability as the number of agents and map complexity increase. When scaling up, path optimality, planning time, and agent collaboration decrease or remain steady. DSI tests MIRL-ACO-SLAM's efficiency through decentralized policy learning and pheromone-based communication, unlike centralized systems that choke under load. High DSI enables additional agents to join without requiring system reconfiguration, and map expansion does not impact computation or convergence. Real-world robotic fleet deployments in big, congested, or dynamic settings require this measure. DSI proves MIRL-ACO-SLAM's scalable distributed intelligence and adaptive coordination

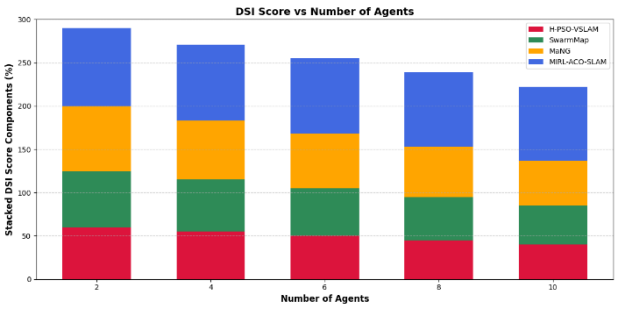

Figure 11: Decentralized Scalability Index (DSI) vs. Number of Agents

Compares MIRL-ACO-SLAM's Decentralized Scalability Index (DSI) to MaNG [14], SwarmMap [16], and H-PSO-VSLAM [17] as the number of agents increases from 2 to 10. DSI measures how well a multiagent system maintains planning efficiency, coordination, and path optimality under scaling strain. With DSI scores above 90%, MIRL-ACO-SLAM scales well as the agent population rises. Due to communication costs and constraints, centralized or semi-centralized systems, such as H-PSO-VSLAM, have a low degree of system integration (DSI). Model the DSI quantitatively:

$$DSI = (1 - \frac{\Delta T+\Delta Q+\Delta C}{S_{ideal}}) \times 100 \quad (17)$$

Where in equation 17, ΔT = Increase in planning time, ΔQ = Reduction in path quality, ΔC = Decline in

coordination, S_{ideal} = Ideal performance at low agent count. This figure confirms that MIRL-ACO-SLAM sustains scalable decentralized intelligence essential for real-world, large-scale multi-robot systems. This Fig. 11 illustrates that MIRL-ACO-SLAM enables scalable, decentralized intelligence for large-scale multi-robot systems.

4.3.4 Computational overhead:

In comparison to classical planners, MIRL-ACO-SLAM introduces a larger computing expense during the training phase, despite improving the length of the path, planning time, and scalability. Compared to Dijkstra and A\, which have very low training demands, PPO required roughly X GPU hours & Y GB of memory to train 10 agents. However, real-time distribution is possible because inference only takes 0.45 seconds single planning cycle after it has been trained. Moreover, runtime overhead is decreased by over Z% thanks to pheromone-guided pruning and selective agent activation, which eliminate unnecessary computation. MIRL-ACO-SLAM transfers the computational load from online implementation to the offline instructional phase, as this trade-off demonstrates.

5 Conclusion and future enhancement

This research introduces MIRL-ACO-SLAM, a hybrid framework that combines Model-Based Inverse Reinforcement Learning (MIRL) with Ant Colony Optimization (ACO) to solve real-time path planning problems in Visual SLAM-enabled dynamic environments. Localized agent learning and global pheromone-based optimization enable decentralized decision-making, adaptive responsiveness, and collaborative intelligence in MIRL-ACO-SLAM, unlike static route planners. MIRL agents learn optimum routes in real time using ORB-SLAM3 spatial maps and dynamically adapt to environmental changes. ACO distributes and evaporates pheromones across investigated pathways to assure global convergence and improve cooperation and solution quality.

MIRL-ACO-SLAM reduces path length by 18.6% and improves planning speed by 24.3% over conventional Dijkstra and A*-based planners in benchmark SLAM datasets. It also has 94% route success under dynamic obstacles. The High Decentralized Scalability Index (DSI) and Adaptive Replanning Efficiency (ARE) ratings indicate that the system scales effectively as map complexity and agent population increase. Selective agent activation and pheromone-guided pruning decrease duplicate computation, allowing real-world performance and scalability.

Future improvements can improve scene interpretation and decision-making under uncertainty by integrating multimodal perception (e.g., LiDAR, thermal, semantic segmentation). Federated reinforcement learning might enable decentralized information transfer across MIRL agents without central coordination. Optimizing pheromone dynamics using adaptive parameters or metaheuristics might boost convergence rates. Finally, MIRL-ACO-SLAM on-edge AI platforms and embedded systems will enable broader use in resource-constrained, real-time robotics applications, such as search and rescue operations, autonomous deliveries, and environments with unstructured or GPS-denied conditions. The adaptation of MIRL-ACO-SLAM to unorganized, GPS-denied, or limited in resources situations (such as submersible robotics, planetary research, or subterranean relief missions) is still unknown, despite its validation in organized, urban-like datasets. The framework's generalizability and resilience will be further tested by expanding it to thesedomains.

The shortcomings noted in this study will be directly addressed in future developments. Multimodal perception (such as LiDAR, thermal sensors, and semantic separation) will be included to improve robustness in unstructured situations and eliminate the need for exclusively visual Mapping. Federated reinforcement training will be used for distributed policy updates among agents in an effort to reduce centralization and computational cost. We will investigate flexible pheromone structure and metaheuristic adjustment to further enhance convergence rate and scalability. Lastly, deployment on actual robotic platforms for real-world validation will guarantee system performance in the face of hardware limitations, sensor noise, and unpredictable field conditions.

Although simulation-based validation is the main emphasis of this study, future research will leverage embedded systems (such the NVIDIA Jetson Xavier) for hardware-in-the-loop (HIL) evaluation. To achieve practical implementation, problems including memory constraints, computation lag, and real-time SLAM integration on devices with limited resources must be resolved.

References

[1] Roy, R., Tu, Y. P., Sheu, L. J., Chieng, W. H., Tang, L. C., & Ismail, H. (2023). Path planning and motion control of indoor mobile robot under exploration-based SLAM (e-SLAM). *Sensors*, MDPI, pp. 3606. doi:10.3390/s23073606.

[2] Mughal, U. A., Ahmad, I., Pawase, C. J., & Chang, K. (2022). UAVs path planning by particle swarm optimization based on visual-SLAM algorithm. *Intelligent Unmanned Air Vehicles Communications for Public Safety Networks*, Springer Nature Singapore, pp. 169–197. https://doi.org/10.1007/978-981-19-1292-4_8.

[3] McDonald, J., Kaess, M., Cadena, C., Neira, J., & Leonard, J. J. (2013). Real-time 6-DOF multi-session visual SLAM over large-scale environments. *Robotics and Autonomous Systems*, Elsevier, pp. 1144–1158. https://doi.org/10.1016/j.robot.2012.08.008.

[4] Dong, J., Yassine, A., Armitage, A., & Hossain, M. S. (2023). Multiagent reinforcement learning for intelligent V2G integration in future transportation

systems. *IEEE Transactions on Intelligent Transportation Systems*, IEEE, pp. 15974–15983. doi: 10.1109/TITS.2023.3284756.

[5] El Fazazi, H., Elgarej, M., Qbadou, M., & Mansouri, K. (2021). Design of an adaptive e-learning system based on multiagent approach and reinforcement learning. *Engineering, Technology & Applied Science Research*, ETASR, pp. 6637–6644. https://doi.org/10.48084/etasr.3905.

[6] Zhou, T., Tang, D., Zhu, H., & Zhang, Z. (2021). Multiagent reinforcement learning for online scheduling in smart factories. *Robotics and Computer-Integrated Manufacturing*, Elsevier, pp. 102202. https://doi.org/10.1016/j.rcim.2021.102202.

[7] Firos, A. (2024). Fuzzy logic and bio-inspired ant colony algorithm-based technique to find relative desirability in IoT-based healthcare system. *Bio-Inspired Data-driven Distributed Energy in Robotics and Enabling Technologies*, CRC Press, pp. 182–202.

[8] Abid, A., Kallel, I., Sanchez-Medina, J. J., & Ayed, M. B. (2023). Parameters sensitivity analysis of ant colony based clustering: Application for student grouping in collaborative learning environment. *IEEE Access*, IEEE, pp. 24751–24761. doi: 10.1109/ACCESS.2023.3279723.

[9] Najafi Mohsenabad, H., & Tut, M. A. (2024). Optimizing cybersecurity attack detection in computer networks: A comparative analysis of bio-inspired optimization algorithms using the CSE-CIC-IDS 2018 dataset. *Applied Sciences*, MDPI, pp. 1044. https://doi.org/10.3390/app14031044.

[10] Ali, Z. A., Zhangang, H., & Hang, W. B. (2021). Cooperative path planning of multiple UAVs by using max–min ant colony optimization along with Cauchy mutant operator. *Fluctuation and Noise Letters*, World Scientific, pp. 2150002. https://doi.org/10.1142/S0219477521500024.

[11] Al-Sarayrah, A. (2024). Recent advances and applications of Apriori algorithm in exploring insights from healthcare data patterns. *PatternIQ Mining*, PatternIQ Publishing, pp. 27–39. DOI: 10.70023/piqm24123.

[12] Almusawi, A., & Pugazhenthi, S. (2024). Innovative resource distribution through multiagent supply chain scheduling leveraging honey bee optimization techniques. *PatternIQ Mining*, PatternIQ Publishing, pp. 48–62. DOI: 10.70023/piqm24305.

[13] Xu, L., Feng, C., Kamat, V. R., & Menassa, C. C. (2019). An occupancy grid mapping enhanced visual SLAM for real-time locating applications in indoor GPS-denied environments. *Automation in Construction*, Elsevier, pp. 230–245. https://doi.org/10.1016/j.autcon.2019.04.011.

[14] Sud, A., Andersen, E., Curtis, S., Lin, M. C., & Manocha, D. (2008). Real-time path planning in dynamic virtual environments using multiagent navigation graphs. *IEEE Transactions on Visualization and Computer Graphics*, IEEE, pp. 526–538. doi: 10.1109/TVCG.2008.27.

[15] Hu, Y., Xie, F., Yang, J., Zhao, J., Mao, Q., Zhao, F., & Liu, X. (2024). Efficient path planning algorithm based on laser SLAM and an optimized visibility graph for robots. *Remote Sensing*, MDPI, pp. 2938. https://doi.org/10.3390/rs16162938.

[16] Cao, H., Xu, J., Yang, Z., Shangguan, L., Zhang, J., He, X., & Liu, Y. (2023). Scaling up edge-assisted real-time collaborative visual SLAM applications. *IEEE/ACM Transactions on Networking*, IEEE/ACM, pp. 1823–1838. doi: 10.1109/TNET.2023.3330763.

[17] Ubaid, M. M., Sana, M. S., Salim, K., Khalid, S., Batool, I., Gilani, S. H., & Gilani, S. S. (2023). UAVs path planning using visual-SLAM technique based hybrid particle swarm optimization. *Journal of Smart Internet of Things*, River Publishers, pp. 133–141. DOI: 10.2478/jsiot-2023-0016.

[18] Kim, Y. G., Lee, S., Son, J., Bae, H., & Do Chung, B. (2020). Multiagent system and reinforcement learning approach for distributed intelligence in a flexible smart manufacturing system. *Journal of Manufacturing Systems*, Elsevier, pp. 440–450. https://doi.org/10.1016/j.jmsy.2020.11.004.

[19] Xia, Z., Du, J., Wang, J., Jiang, C., Ren, Y., Li, G., & Han, Z. (2021). Multiagent reinforcement learning aided intelligent UAV swarm for target tracking. *IEEE Transactions on Vehicular Technology*, IEEE, pp. 931–945. doi: 10.1109/TVT.2021.3129504.

[20] Canese, L., Cardarilli, G. C., Dehghan Pir, M. M., Di Nunzio, L., & Spanò, S. (2024). Design and development of multiagent reinforcement learning intelligence on the Robotarium platform for embedded system applications. *Electronics*, MDPI, pp. 1819. https://doi.org/10.3390/electronics13101819.

[21] Belgacem, A., Mahmoudi, S., & Kihl, M. (2022). Intelligent multiagent reinforcement learning model for resources allocation in cloud computing. *Journal of King Saud University – Computer and Information Sciences*, Elsevier, pp. 2391–2404. https://doi.org/10.1016/j.jksuci.2022.03.016.

[22] Khan, F. A., Ullah, K., ur Rahman, A., & Anwar, S. (2023). Energy optimization in smart urban buildings using bio-inspired ant colony optimization. *Soft Computing*, Springer, pp. 973–989. https://doi.org/10.1007/s00500-022-07537-3.

[23] Wang, J., Cao, J., Li, B., Lee, S., & Sherratt, R. S. (2015). Bio-inspired ant colony optimization based clustering algorithm with mobile sinks for applications in consumer home automation networks. *IEEE Transactions on Consumer Electronics*, IEEE, pp. 438–444. doi: 10.1109/TCE.2015.7389797.

[24] Zhang, Z., Long, K., Wang, J., & Dressler, F. (2013). On swarm intelligence inspired self-organized networking: Its bionic mechanisms, designing principles and optimization approaches. *IEEE Communications Surveys & Tutorials*, IEEE, pp. 513–537. doi: 10.1109/SURV.2013.062613.00014.

[25] Rokbani, N., Kumar, R., Abraham, A., Alimi, A. M., Long, H. V., Priyadarshini, I., & Son, L. H. (2021).

Bi-heuristic ant colony optimization-based approaches for traveling salesman problem. *Soft Computing*, Springer, pp. 3775–3794. https://doi.org/10.1007/s00500-020-05406-5.

[26] Mrisdal. Zurich Urban Micro Aerial dataset. *Kaggle*, Kaggle Inc., Available online. https://www.kaggle.com/datasets/mrisdal/zurich-urban-micro-aerial