Journal of Intelligent Systems and Internet of Things
Vol. 18, No. 02, PP. 36-59, 2026
DOI: https://doi.org/10.54216/JISIoT.180203

A New Strategy for Exploration and Area Coverage Using Swarm Robots by Enhancing the Pelican Optimization Algorithm

Dena Kadhim Muhsen[1,*], Ahmed T. Sadiq[1], Firas Abdulrazzaq Raheem[2]
[1]Computer Science College, University of Technology - Iraq, 10066 Baghdad, Iraq
[2]Control and Systems Engineering College, University of Technology - Iraq, 10066 Baghdad, Iraq
Emails: dena.k.muhsen@uotechnology.edu.iq; ahmed.t.sadiq@uotechnology.edu.iq; Firas.A.Raheem@uotechnology.edu.iq

Abstract

Area coverage and exploration of unknown environments by swarm robots autonomously is one of the challenges in the robotics domain. This paper proposes a new strategy for area coverage in two parts; firstly, enhancing a Pelican Optimization Algorithm (POA) using swarm robots to explore an unknown area. Secondly, merges many algorithms with the proposed POA, such as Timed Elastic Band (TEB) as a local planner for obstacle avoidance, SLAM (Simultaneous Localization and Mapping), and a training model which is called You Only Look Once version 8 nano (YOLOv8n) for person detection. The proposed POA algorithm successfully monitored a large area and achieved a high exploration ratio with minimal time. In this work, the new strategy is applied to a robot warehouse environment, utilizing a swarm of robots to explore the area and find targets, which are employees suffocated by the effects of chemical pollution. The simulation and real-world tests for a new strategy were done in the Robot Operating System (ROS) using the TurtleBot3 robot. The total time-consuming for exploration and detection time is less by POA, while the coverage ratio is the largest when compared with the original RRT exploration algorithm for empty, small, and large environments, respectively.

Received: March 19, 2025 Revised: June 10, 2025 Accepted: July 18, 2025

Keywords: Area coverage; POA; Autonomous robot; ROS; Turtlebot3 Burger robot; YOLOv8n

1. Introduction

In the mobile robotics systems world, the term exploration indicates the process of exploring an unknown area by one or multiple robots and building a map for the environment either indoor or outdoor. Many applications use multiple robots or swarm robots that are moving without human assistance for the purpose of exploration [1, 2]. These applications comprise autonomous transportation [3], industry [4,5], healthcare [6], rescue [7,8], and agriculture [9,10]. The basic concept of mobile robots is the ability to detect the shortest path between the start point and the target point. In addition, avoiding obstacles and reaching a destination is the main aim of mobile robots in many different environments either known or unknown [11]. Swarm robots are decentralized robustness systems in which many robots cooperatively do a particular task successfully [12]. To improve coverage in an area the swarm robots are distributed to accelerate the exploring area and find targets in minimum time and maximum coverage. Autonomous robot swarms can construct a highly robust system to localize the robot within its environment and navigate while avoiding boundaries and dynamic obstacles [13]. Many algorithms and methods are used to assist fully decentralized autonomous mobile robots and achieve area coverage, such as multiple modality sensor data fusion, Simultaneous Localization and Mapping (SLAM) algorithms, and optimized Artificial Intelligence (AI) techniques [14]. One of the important artificial intelligence algorithms is the metaheuristic algorithms, which are nature-inspired optimization methods that guide robots to improve the solution but not produce the best solution. In most situations, the solution is good and near to optimal within a reasonable amount of time [15]. The Pelican optimization algorithm (POA) is one of the metaheuristic algorithms used in this work.

The POA algorithm uses in its operation, the random function to achieve the distribution of the population in the initialization stage. The algorithm is largely based on the solution produced from the initialization stage. In the development stage, which leads to the situation, the algorithm is easy to fall into a local optimum [16]. The robot operating system (ROS) is the main framework for building robot software used to implement techniques in this work. ROS is similar to an operating system on a personal computer, in that it includes a set of programs that provide control to a user. In the situation of a ROS however, these programs allow a user to control the mobile operations of a robot instead of applications on a computer [17]. Turtlebot3 robot burger- type based on ROS is the hardware used. It is a small and useful platform used in education, research, and product prototyping. It comprises a Raspberry Pi 3 SBC suitable for robust embedded systems and has a 360-degree distance sensor that gives it the ability to achieve SLAM and navigation in different indoor environments [18]. Gazebo and RViz are general-purpose robotics simulator platforms used by ROS. Various components are required to operate the Gazebo simulation. All elements necessary for simulation such as objects, robots, sensors, and light sources are provided in a file with .world extension. This file uses XML format for simulation elements description [19]. Many general applications can use this strategy, but in this work, it applies to a robot warehouse environment and uses a swarm of robots to explore the area and find targets, which are employees suffocated from the effects of chemical pollution. It uses this case study because the urgent cases need to be discovered quickly to emergency it and save people's lives, which is the highest priority considered in most systems of area coverage for search and rescue. The new strategy is used for exploring unknown area and finding suffocated persons in the minimum time. The contributions of this work are:

- Enhancing the Pelican optimization algorithm for exploration and area coverage by swarm mobile robots.
- Propose a new strategy that includes the proposed Pelican optimization algorithm, SLAM, TEB algorithm, YOLOv8n algorithm, and move base in ROS.
- Prove that the new POA algorithm and new strategy are more effective for exploration in unknown area with static and dynamic obstacles.

This work is systematized as follows: Section 2 illustrates the studies for mobile swarm robots and algorithms used for exploration and area coverage, section 3 submits the pelican algorithm as the main algorithm for exploration and global path planning, section 4 presents the Proposed Strategy for Exploring and Area Coverage and all necessary algorithms and tools to implement enhances POA and to quickly detect suffocated persons to easy reach and emergency them by ambulance. Section 5 explains robot simulation and real-world experiments and models used, and section 6 conclusions.

2. Related works

According to Franchi A. and others in 2007 submitted a new method for autonomous decentralized mobile robots frontier exploration and mapping of an unknown area. The robots cooperate with each other and share the information of their positions and target points. The basic idea is to reduce the time of exploration by robots to detect target frontier points. The simulation results showed that the proposed strategy is reducing exploration time compared to a state-of-the-art strategy [20]. While Yi Zhou and others in 2013 proposed, a new algorithm depends on frontiers and Particle Swarm Optimization (PSO). Multiple mobile robots cooperate with each other to explore unknown environments by the proposed distributed algorithm. The robots are navigated toward the remote frontier after exploring the local area. The exploration is completed when there is no frontier cell in the environment. Simulation and real robots are tested, and the results show that the time of the proposed algorithm is less than about 60% compared with other methods of the random frontier and is 21 that are more efficient. Umari, H., S.in 2017, presents a new exploration strategy based on the use of multiple rapidly exploring Random Trees (RRTs). The proposed strategy is implemented and tested using the Robot Operating System (ROS) framework. Additionally, this work uses local and global trees for detecting frontier points, which enables efficient robotic exploration [22]. Zeng T. B. in 2019 proposed an algorithm for area exploration by mobile robots in an unknown environment depending on a small point cloud local map. Frontier detection is done by implementing the RRT algorithm on unordered point cloud local maps, and Motion planning is achieved by the dynamic window approach (DWA) algorithm, with a guarantee the resources of computation did not increase when the map expands. the simulation results showed a high-quality exploration, assisting in obstacle avoidance in dynamic environments and reducing computing cost and complexity [23].

In addition, Indraneel P. and others in 2023 proposed a new exploration strategy called PGART planner that connects mapping and searching by growing sets of random trees rooted upon a pose graph produced through mapping to create points of interest. The important objective for robots is to detect victim's positions in the minimum time. The strategy results are illustrated in the ROS and give more area coverage at 20% more than other state-of-the-art algorithms, and more efficiently in an unknown environment [24]. Khaleel, R. Z and others proposed two methods in 2024, represented by Pelican Optimization Algorithm (POA) and Particle Optimization Algorithm (PSO) to have an optimal trajectory for the mobile-wheeled robot without collision. The mathematical

representation of a mobile wheeled robot has been developed and experimental results have been conducted to verify the numerical results, which have been simulated using MATLAB software environment. The results showed that less trajectory error can be obtained with POA as compared to PSO [25]. Li, C. Q and others proposed a multi-strategy improved pelican optimization algorithm (MPOA) in 2024. In the initialization stage, chaotic mapping is used to increase the diversity of the pelican population individuals. In the exploration stage, an adaptive feedback adjustment factor is proposed to adjust the local optima of pelican individuals' positions and balance the algorithm's local development capability. In the development stage, the Lévy flight strategy is introduced to adjust the domain radius of the pelican population individuals, and the Gaussian mutation mechanism is used Simulation experimental results showed that the improved algorithm has significantly improved and effectively shortened the length of the planned path [26].

In our work use the suggested POA for exploration and area coverage to provide a solution by addressing the shortcomings of earlier exploration algorithms. The proposed new strategy uses this enhanced POA led to consume the least amount of time while producing very precise detection. In this work, this new strategy is an efficient manner and faster exploration for an unknown area than previous methods. It applied in this work to a robot warehouse environment and uses a swarm of robots to explore the area with find targets, which are employees suffocated from the effects of chemical pollution.

3. Pelican optimization algorithm (POA)

One of the significant artificial intelligence algorithms is the metaheuristic algorithm, which is a nature-inspired optimization algorithm called the Pelican optimization algorithm. The pelican is big and has a long beak with a large bag in its throat, which is used to catch and swallow the prey. It lives in groups of more than a hundred pelicans. Pelicans search together for food that includes fish and sometimes frogs, turtles, and crustaceans. The basic idea of the algorithm and all its steps is illustrated in Algorithm 1 [27-29]. The operation of the hunt begins after detecting the position of the prey, and then, diving to it. Then, they spread their wings on the surface of the water to force the fish to go to shallow water so that they can catch them. A large amount of water enters to the pelican's beak when hunting fish, which moves the head forward before swallowing the fish to remove excess water. Due to their lifestyle and strategy of hunting followed by pelicans, they are considered to do intelligent operations, and these birds are highly skilled. The POA is a population-based algorithm in which pelicans are members of this population; each population member means a candidate solution. Initially, population members are randomly initialized due to the lower bound and upper bound of the problem using (1) [30]:

$$(1)\quad X_{a,b} = l_b + rand.(u_b - l_b), a = 1.2, \ldots . N. b = \ 1.2 \ldots \ldots, m$$

where $X_{a,b}$ is the value of the *b*th variable specified by the ath candidate solution, N is the number of population members, m is the number of problem variables, rand is a random number in the interval [0, 1], l_b is the *b*th lower bound, and u_b is the *b*th upper bound of problem variables. The population members of pelicans in the POA are identified using the population matrix in (2). Each row of this matrix represents a candidate solution, while the columns of this matrix represent the proposed values for the problem variables.

$$(2)\qquad X = \begin{bmatrix} X_1 \\ \vdots \\ X_a \\ \vdots \\ X_N \end{bmatrix}_{N\times m} = \begin{bmatrix} X_{1,1} & \cdots & X_{1,b} & \cdots & X_{1,m} \\ \vdots & \ddots & \vdots & \iddots & \vdots \\ X_{a,1} & \cdots & X_{a,b} & \cdots & X_{a,m} \\ \vdots & \iddots & \vdots & \ddots & \vdots \\ X_{N,1} & \cdots & X_{N,b} & \cdots & X_{N,m} \end{bmatrix}_{N\times m}$$

where X is the population matrix of pelicans and X_a is the *a*th pelican. The objective function of the given problem was evaluated based on each of the candidate solutions. The values obtained for the objective function are determined using a vector called the objective function vector in (3):

$$(3)\qquad \begin{bmatrix} F_1 \\ \vdots \\ F_a \\ \vdots \\ F_N \end{bmatrix} = \begin{bmatrix} F(x_1) \\ \vdots \\ F(X_a) \\ \vdots \\ F(X_N) \end{bmatrix}$$

$N \times 1 \qquad N \times 1$

where F is the objective function vector and F_a is the objective function value of the ith candidate solution. The POA strategy has two phases: Exploration when pelicans move towards prey and exploitation when winging on the water surface [31]. The pelican's movement to the location of prey is simulated using (4):

(4) $$X^{P1}{}_{a,b} = \begin{cases} X_{a,b} + rand\ .\ (P_b - I.X_{a,b}) & F_p < F_a \\ X_{a,b} + rand\ .\ (X_{a,b} - P_b) & otherwise \end{cases}$$

where $X^{P1}{}_{a,b}$ is the new status of the *a*th pelican in the *b*th dimension, I is the random number either 1 or 2, p_b is the location of prey, and F_p is the value of the objective function. The new location of Pelican is updated by (5) where $X^{P1}{}_a$ is the new status of the *a*th pelican and $F^{P1}{}_a$ is the value of the objective function:

(5) $$Xa = \begin{cases} X^{P1}{}_a & F^{P1}\ a < F_a \\ X_a & otherwise \end{cases}$$

Calculate the new status of the jth dimension by (6) and update the ith population member by (7):

(6) $$X^{P2}{}_{a,b} = X_{a,b} + R \cdot \left(1 - \frac{t}{T}\right) \cdot (2 \cdot rand - 1).X_{a,b}$$

(7) $$Xa = \begin{cases} X^{P2}{}_a & F^{P2}\ a < F_a \\ X_a & otherwise \end{cases}$$

Algorithm 1: Pelican Optimization Algorithm (POA) [29]

Input:

N population size

T number of iterations

I Random number

Output: best candidate solution obtained by POA.

Initialization:

position of pelicans

FP ◄— objective function

Begin

1. For t= 1 to T
2. Generate the position of the prey at random.
   For a = 1 to N
   Phase 1: Exploration the search space
5. For b = 1 to m
6. Calculate new status of the *b*th dimension, Eq. 4.
7. End.
8. Update the *a*th population member by Eq. 5.

ase 2: Exploiting the search space

10. For b = 1 to m
11. Calculate new status of the *b*th dimension, Eq. 6.
12. End.
13. Update the *a*th population member by Eq. 7.
14. End.
15. Update best candidate solution.
16. End.

End.

4. System design to implement the new strategy for exploration

This work proposes a new strategy for area coverage in two parts; firstly, enhancing a pelican optimization algorithm (POA) using swarm robots to explore an unknown area. Secondly, merges many algorithms with the proposed POA, such as Timed Elastic Band (TEB) as a local planner for obstacle avoidance, SLAM (Simultaneous Localization and Mapping), and a training model which is called You Only Look Once version 8 nano (YOLOv8n) [32] for detecting all targets (suffocated persons) in this area. Five TurtleBot3 robots of Burger-type are used to accomplish this task. Many simulation testing and real-world experiments are tested in the Robot Operating System (ROS), Gazebo, and RViz to produce the results. All the outputs of the proposed algorithm are evaluated to compute the total time and ratio for area coverage ensure that all targets are found, make comparisons, and prove to achieve maximum area cover in minimum time. There are many stages to implement the system:

4.1 Initialization stage

The indoor environment of the system application model is built in Gazebo as a warehouse model comprising many targets, which are employees on the ground because they are suffocated from the effects of chemical pollution and should be rescued. Many static obstacles, such as boxes, were put in the warehouse, and the movement of robots that transfer the products represents dynamic obstacles. Three TB3B robots were placed inside the warehouse for exploration as shown in Fig. 1. The simulation warehouse dimensions are set to 7.15 m * 7.5 m. The fine-tuned Gmapping SLAM was used to map the environment [33]. In addition, the environment in the real world is a room with dimensions of 3.5 m * 3.5 m, including some boxes as static obstacles, toys on the ground representing suffocated persons, as in Fig. 2. Table 1 presents the specifications of the computational resources used in this study.

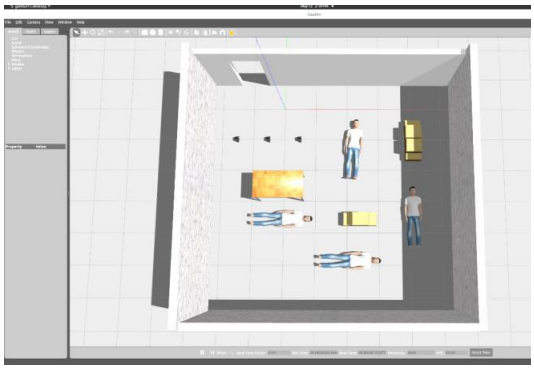


Figure 1. The work environment in Gazebo

Figure 2. The work environment in a real-world

Table 1: Platform and configurations of the environment.

Platform	Configuration
Linux	Ubuntu 20.04.6 LTS
CPU	12th Gen Intel Core I9-12900H* 20
GPU	NVIDIA GeForce GTX 1060 6G
Memory	15.3 GB
Programming language	Python - C++
ROS	Noetic
Gazebo	Version 11
RViz	Version 1.14.20

TB3B is used which is a 2D LiDAR-based differential drive mobile robot platform that has open-source compatibility with ROS. As shown in Fig. 3, TB3B consists of four layers [34].

- **Layer 1:** Base layer, two Dynamixel XL 430-W250 motors are connected to a low-level controller: the Open-source Control Module for ROS (OpenCR). A lithium–polymer battery is placed between them.
- **Layer 2:** Contains the OpenCR board. It comprises an inertial measurement unit and connects the central controller to the motors and LiDAR.
- **Layer 3:** Holds the central controller, which is a Raspberry Pi board. It interacts with all other components of the mobile robot platform, and it communicates with RViz and Gazebo on the PC via Wi-Fi and SSH protocols.
- **Layer 4:** A 360° LDS-08 LiDAR sensor is placed on top of the robot that can gather distance data around the robot in the range of 120–3500 mm.

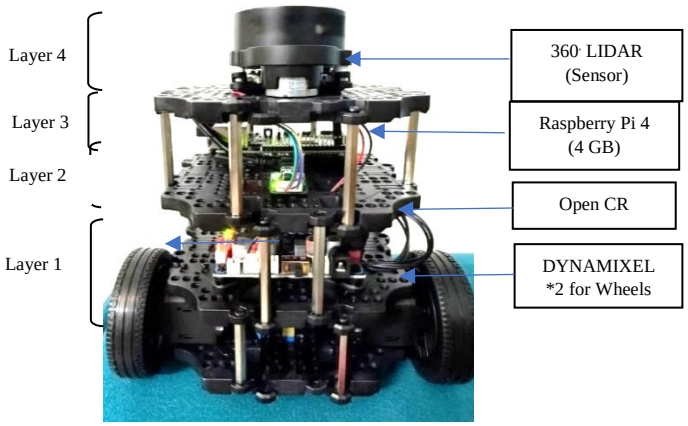


Figure 3. Layers of Turtlebot3 Burger–Type

4.2 YOLOv8n detection stage

The process of training the YOLOv8n model involves several key steps, including image acquisition, preprocessing, training, integration with ROS, and performance evaluation, as depicted in Fig.4. These steps are detailed as follows:

1. **Images acquisition:** Images of people on the ground were created as combine's data from a personal dataset from the simulations and Roboflow data website [35]. This dataset enhanced the model's detection ability in both real and simulation environments.

2. **Preprocessing:** The dataset is divided into three split sets for training, validation, and testing. The training dataset enabled the model to learn the key features that exist in the images. The validation set was employed to validate the model's performance during training. While, the testing set, which includes unseen data, was used to evaluate the trained model's performance.
3. **Training the YOLOv8n model:** This step detects an object by training model. It is an algorithm that divides an image into the grid system and each grid detects objects within itself.

4. **Integration with ROS:** A new package was created for the inference of the YOLOv8n model, including a launch file that configures the necessary parameters such as the model file, the input image topic, and the result topic. This package is integrated with ROS to enable real-time object detection and ensure smooth interaction between the YOLOv8n model and the ROS framework.

5. **Performance Evaluation:** To evaluate the results of the YOLOv8n trained model, many metrics are used, such as Mean Average Precision (mAP), Recall, and Precision. Some of the detection results are shown in Fig.5 and 6 for simulation and the real world, respectively.

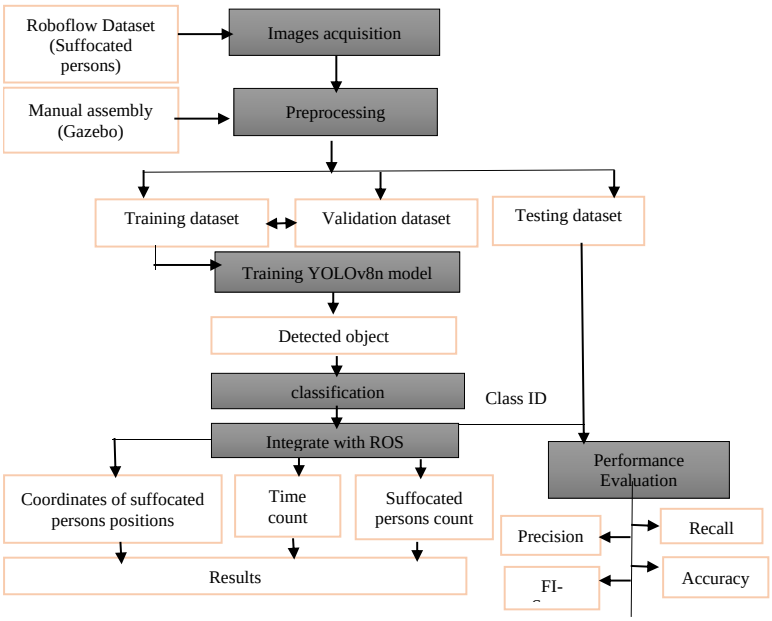


Figure 4. Block diagram of the object detection work

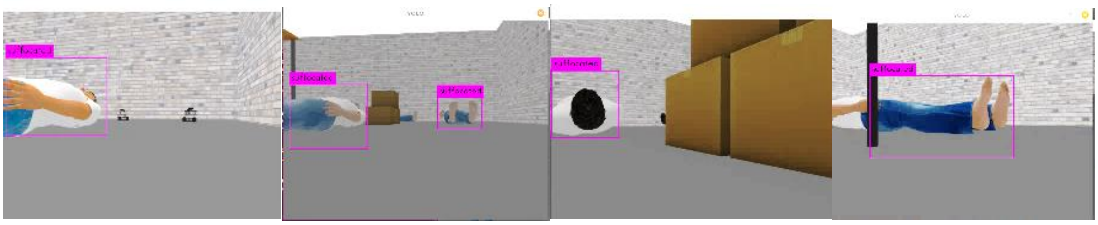


Figure 5. YOLOV8n predicted images in simulation

Figure 6. YOLOV8n predicted images in real world

4.3 Proposes a new enhancement on the pelican algorithm for exploring

Algorithm 2 provides an outline of the proposed POA exploration algorithm for swarm robots. Fig.7 illustrates the structure of the proposed exploration system in general. Each robot runs the POA algorithm to generate exploration goals when utilizing POA for swarm robot exploration. However, multiple robots may generate similar or overlapping goals, leading to redundant computations and inefficient resource utilization. To address this issue, filtering and goal assignment strategies have been implemented [22]:

A. Filtering Exploration Goals: The initial list of exploration goals is filtered to eliminate redundant and invalid points. This process involves:

1. Density Analysis and Grouping:

- Points are analyzed to identify dense regions.
- Points are iteratively shifted towards denser areas to form groups.
- Representative points (centers) from each group are selected to summarize the original data, reducing the number of points while preserving essential information.

2. Computational Efficiency:

- This reduction in points ensures that subsequent steps are more efficient.
- The representative points maintain accuracy while decreasing the computational load.

B. Assigning Exploration Goals to Robots: Exploration goals are assigned to robots based on navigation cost, information gain, and overlap reduction:

1. Navigation Cost: Measures the distance a robot must travel to reach a point.

2. Information Gain: Estimates the area of unknown territory that can be explored from a point.

3. Overlap Reduction:

- Prevents redundant assignments by biasing robots to explore nearby points.
- A reduction function decreases the value of overlapping points.

4. Bidding Approach:

- Robots bid on points based on expected revenue (information gain minus navigation cost).
- The highest bid wins the assignment.
- If all robots are busy, the reduction is temporarily removed to reassess points, ensuring optimal task allocation.

Algorithm 2: The Proposed Pelican Optimization Algorithm (POA) for Robot Exploration

Input:

Optimization problem information, map data, POA population size (N), and number of iterations (T).

Output:

Final exploration goal.

Initialization:

Initialize the robot's position randomly within bounds.

Calculate the initial exploration value based on the robot's position.

Wait for initial map data and points to be received.

Compute initial map dimensions and starting points from received points.

Begin:

For t = 1 to T do
Generate a random point (w_{rand}) in the search space.
Phase 1: Exploring the search space
Find the closest point ($w_{closest}$) to the random point (w_{rand}).
Move towards the random point (w_{rand}) from the closest point ($w_{closest}$) using Eq. 4.
6. Check if the new point (w_{new}) is obstacle-free.
7. if the new point is in a frontier region then
8. Update exploration goal with w_{new} and publish.
9. Else if the new point is in free space (explored area) then
10. Update the robot's position using Eq. 5.

End if

the search space

position based on POA using Eq.6.

djust each dimension of the robot's position.

15. Update the robot's position using Eq. 7.
16. End for

End.

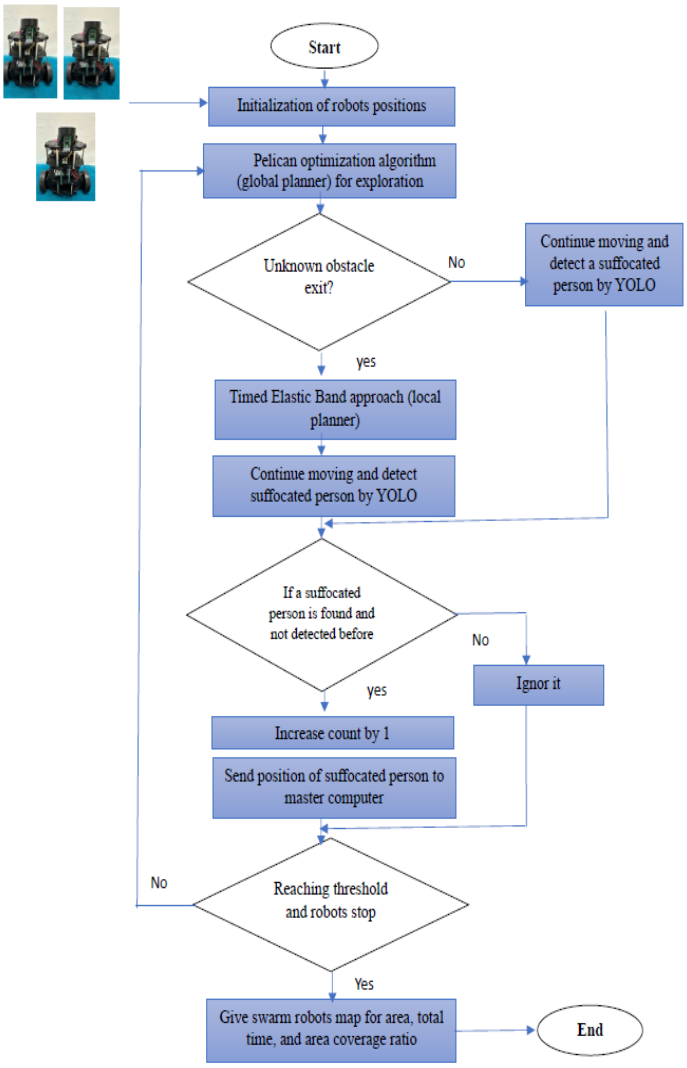


Figure 7. The general structure of the proposed exploration system

5. Results and discussion of comparing exploration and area coverage by pelican exploration algorithm versus RRT exploration algorithm

5.1 Results of YOLOv8n detection

The trained YOLOV8 model on suffocated people identification achieved precision of 88.55%, recall of 83.17%, mean average precision at 50% intersection over union (IoU) known as mAP50% of 90.90%, and mean average precision between 50% and 95% intersection over union (IoU) known as mAP50-95 of 55.76%. These metrics are represented in Table 2.

Table 2: Metric values of YOLOv8n

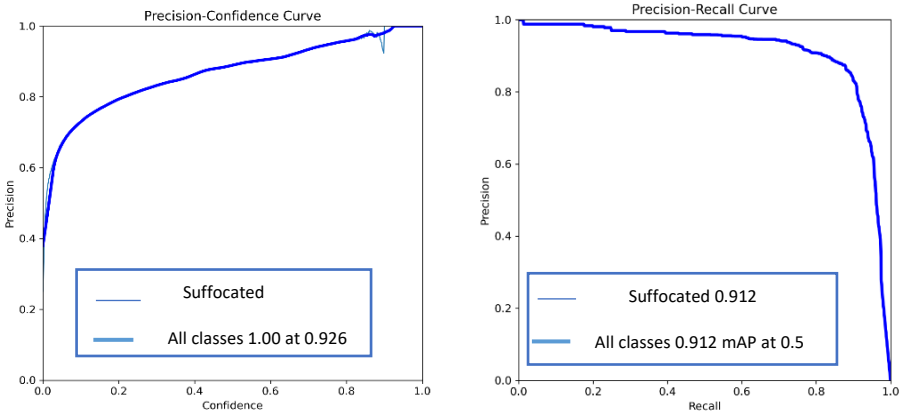
Metrics	Score
Precision	88.55%
Recall	83.17%
mAP50	90.90%
mAP50-95	55.76%

The results of the real-time inference speed of the trained model are summarized in Table 3. The preprocessing step took an average of 4.0 milliseconds (ms), while the inference step required 27.9 ms, and the post processing step took 4.8 ms. This results in a total processing time of 36.7 ms per image, which is approximately 0.0367 seconds per image. As a result, the model achieves a frame rate of about 27.25 frames per second (FPS), proving its capability for real-time detection of suffocated individuals.

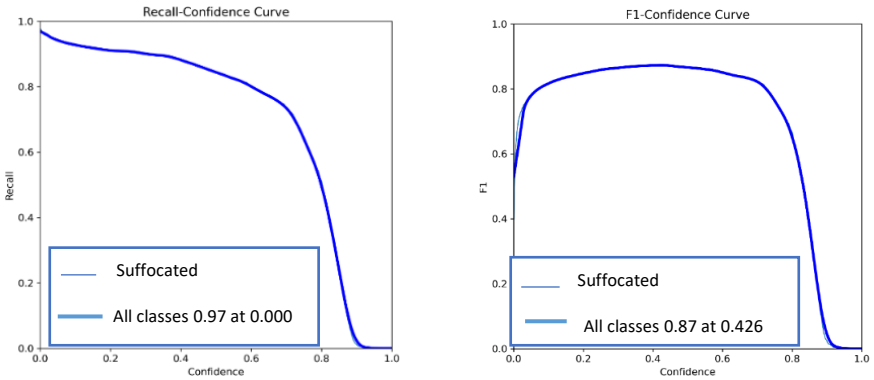
Table 3: The results of the real-time inference speed

Metric	Score
Preprocess	4.0 ms
Inference	27.9 ms
Postprocess	4.8 ms
Total time per image	4.0+27.9+4.8=36.7
Total time per image in seconds	36.7 ms = 36.7×10e-3 seconds = 0.0367 second
FPS	1/0.0367≈27.25

The intersection over union (IoU) evaluation metric was used to evaluate the accuracy of the model detector in the dataset used. Fig. 8 illustrates the training results based on the error value of the model in training. There are four illustrated images: Precision-Confidence, Recall, Recall Confidence, and F1 Confidence. The mAP50 value for the detection of the suffocated class is 90.90%, which serves as an approximation of the accuracy. Furthermore, Fig. 9 illustrates the model-training matrix.



(a) (b)



(c) (d)

Figure 8. (a) Precision Confidence, (b) Recall, (c) Recall Confidence (d) F1-Confidence

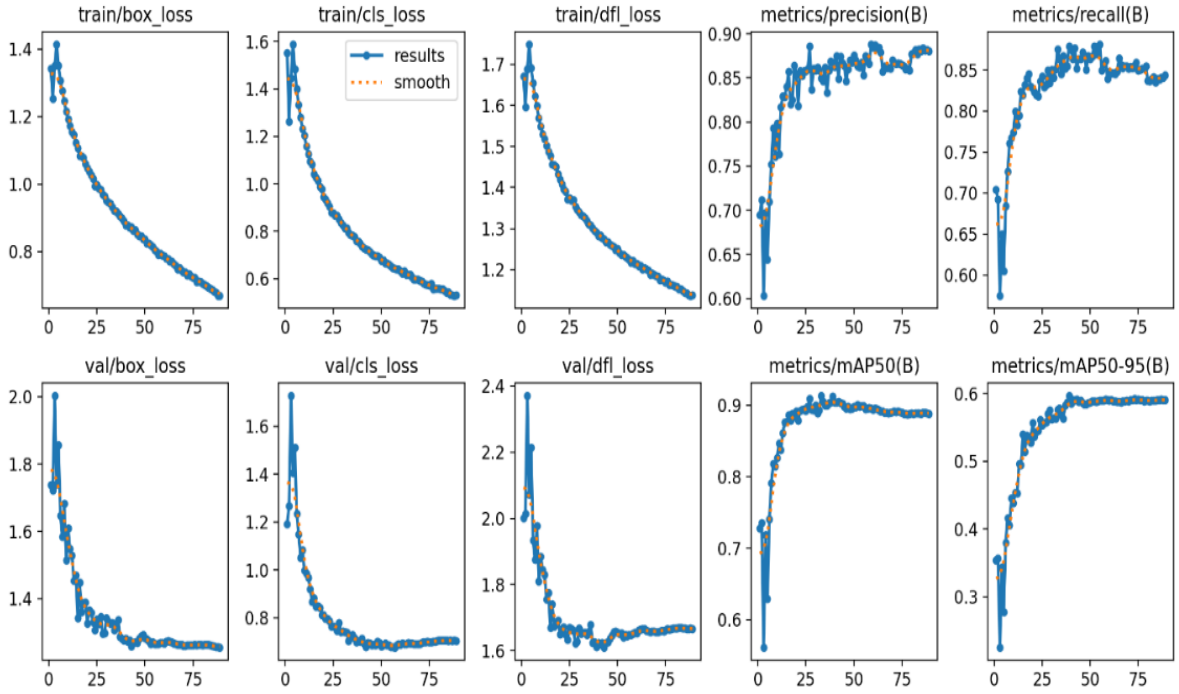


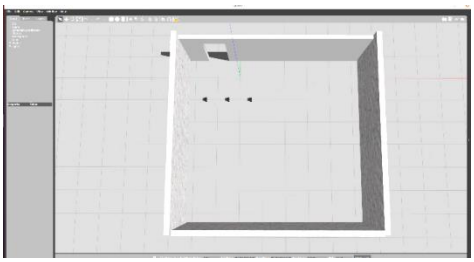
Figure 9. Training matrix

5.2 Simulation and real-world results

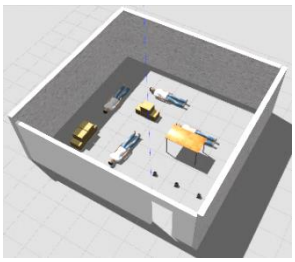
To evaluate the performance of the proposed pelican algorithm for exploration, it was compared with a known RRT exploration algorithm [22]. Testing a hundred simulations for each environment and taking the average of five runs. Three TB3B with cameras are used with YOLOv8n detection in empty, small, large, extra-large, and real-world environments as in Fig. 10 and Table 4. Testing is done in these simulation environments with different numbers of suffocated persons and obstacles, while a real room, including three boxes as obstacles and two toys on the ground as suffocated persons. In all implementations, the robot's paths are indicated by orange, yellow, and red for robot1, robot2, and robot3, respectively in RViz.

Table 4: The specification of environments

Environment type	Environment area
Empty	7.15 * 7.5 m^2
Small	7.15 * 7.5 m^2
Large	10.88 *13.7 m^2
Extra large – model_1	16.2 * 24*87 m^2
Extra large – model_2	21.54 * 34.42 m^2
Small real-world	3.5 * 3.5 m^2



a. The empty warehouse model



b. The small warehouse model

c. The large warehouse model

d. The extra-large warehouse model _1

e. The extra-large warehouse model _2

f. Small real-world model

Figure 10. Gazebo shows the various warehouse models

5.2.1 Empty warehouse environment

Firstly, the exploration is implemented in an empty warehouse with an area of 7.15 * 7.5 m^2, as shown in Fig.11. More than 100 runs were done for both the RRT exploration algorithm and the proposed pelican algorithm and Table. 5 shows the five runs that approximate the average of these runs. Simulation 1 in Table. 5 is illustrated in Fig. 12 in a and b, respectively, with details of map merging. When three TB3B robots explore the warehouse area, the results demonstrate that despite the proposed pelican outperforming the RRT exploration in time consumption enhancing about 40% and area coverage, both these algorithms take more time in an empty environment than the

time taken in an environment that contain objects. The analysis of reasons for increased exploration time is as follows:

- **Lack of Landmarks:**

Both RRT and Pelican exploration algorithms rely on environment features to guide the exploration process. In a featureless environment, the algorithms may have difficulty determining progress and making efficient decisions about where to explore next.

- **Path Planning Challenges:**

In an empty space, path planning might not have clear obstacles to navigate around, leading to inefficient exploration patterns. The algorithms might generate many redundant paths or fail to cover the space efficiently.

- **Sensor and Localization Uncertainty:**

Sensor readings can become less reliable without distinct features, and localization might suffer. This can cause the robots to move more cautiously, re-scan areas, and take a longer time to confirm their positions.

- **Map Merging and Coordination:**

When multiple robots are exploring, merging maps without features can be challenging, leading to inefficiencies and delays in coordinating the exploration efforts.

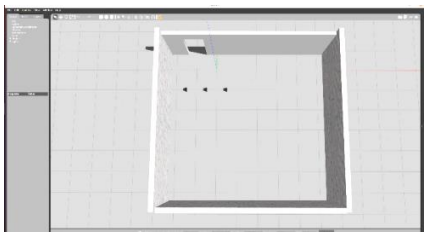


Figure 11. Gazebo shows the empty warehouse model

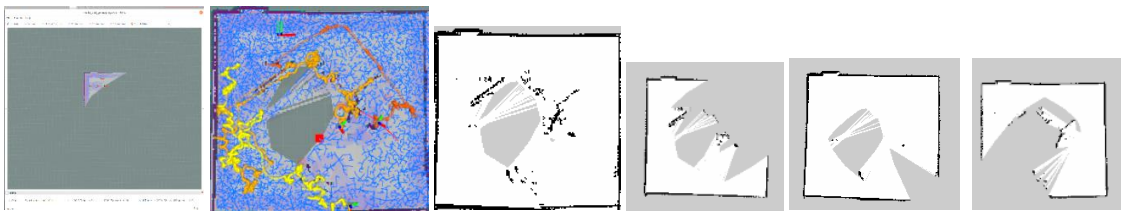


Figure 12- a. Rviz shows the RRT exploration for area coverage by swarm robots in an empty environment, the total merged map for the swarm robots, and the maps for each robot.

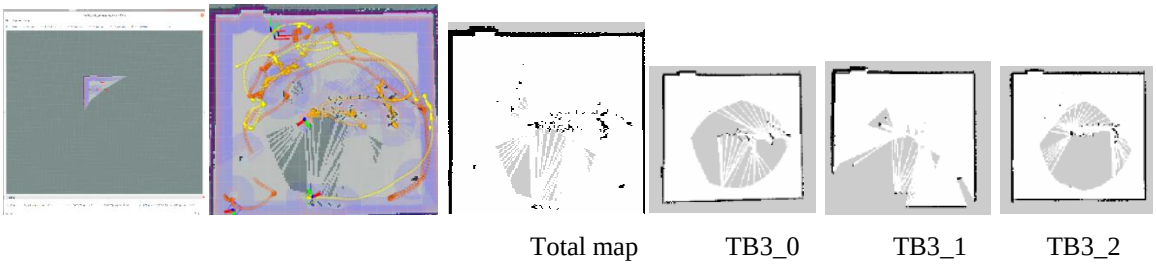
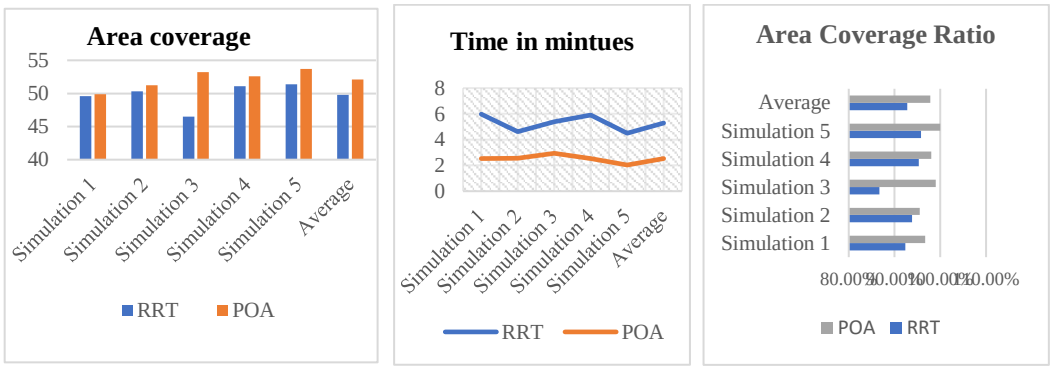


Figure 12-b. Rviz shows the proposed POA exploration for area coverage by swarm robots in an empty environment, the total merged map for swarm robots, and the maps for each robot

Table 5 and Fig. 13 Compare RRT and POA algorithms in five simulations to see how well they covered space and explored efficiently. RRT has an average area coverage of 49.77 units (92.8%) and POA has 52.12 units (97.84%). RRT exploration takes 5.29 minutes, but POA takes 2.52 minutes, a 52% efficiency improvement. Different amounts of times are spent investigating for various causes. Both algorithms rely on environmental features for decision-making and path planning. In featureless simulation worlds, mapping and exploration are difficult without landmarks or barriers .

Table 5: Comparison of RRT exploration versus pelican exploration in an empty environment

Simulation number	Robot number	Robot position	Area Coverage by RRT Exploration				Area Coverage by POA Exploration			
			Area coverage (53.62)	Area coverage ratio	Time in minutes	failer robots (stop for a period)	Area coverage (53.62)	Area ratio	Time in minutes	failer robots (stop for a period)
1	0	(-1.5,-1.0)	49.58	92.4	5.98	No	49.9	96.7%	2.54	No
	1	(0.5,-1.0)				No				No
	2	(-0.5,-1.0)				Yes				No
2	0	(-1.5,-1.0)	50.3	93.8	4.63	No	51.22	95.5%	2.55	No
	1	(0.5,-1.0)				Yes				No
	2	(-0.5,-1.0)				No				No
3	0	(-1.5,-1.0)	46.5	86.7	5.40	Yes	53.2	99 %	2.94	No
	1	(0.5,-1.0)				Yes				No
	2	(-0.5,-1.0)				No				No
4	0	(-1.5,-1.0)	51.1	95.3%	5.92	Yes	52.6	98%	2.54	No
	1	(0.5,-1.0)				No				No
	2	(-0.5,-1.0)				Yes				No
5	0	(-1.5,-1.0)	51.4	95.8	4.55	No	53.7	100%	2..03	No
	1	(0.5,-1.0)				Yes				No
	2	(-0.5,-1.0)				Yes				No
Average			49.77	92.8	5.29		52.12	97.84		



(a) (b) (c)

Figure 13. Comparison of RRT versus POA, a: Area coverage, b: Exploration time, c: Area coverage ratio

5.2.2 Small warehouse environment

Three TB3B robots implemented the exploration with YOLOv8n detection in a small-complicated warehouse with an area of 7.15 * 7.5 m^2. The warehouse included many boxes and one table as obstacles and four persons on the ground suffocated from warehouse pollution. The small warehouse environment is shown in Fig. 14. More than 100 runs were done for both the RRT exploration algorithm and the proposed pelican algorithm, and Table 6 shows the five runs that approximate the average of these runs. Simulation 1 in Table 6 is illustrated in Fig. 15 in a and b, respectively, with map merging details. The results illustrated that the exploration by the proposed pelican algorithm produces smooth navigation and reduces randomness through robots moving in this environment. The main important points noticed compared to the RRT exploration algorithm in this environment are:

- **Maximum area coverage:** The coverage by the proposed pelican algorithm is better than that of the RRT algorithm because it reduces randomness and follows the expected paths depending on behavior rather than random paths generated by the tree of the RRT algorithm.
- **Time consumption:** The proposed pelican algorithm is faster than the RRT algorithm due to the fast decisions and guiding the path toward the next goal depending on algorithm behavior and environmental features.
- **Coverage ratio:** the proposed pelican algorithm covers the area by approximately 99.9%, while the RRT exploration algorithm covers less ratio because some of the robots stop.
- **Robot frailer:** In the proposed pelican algorithm, the robots did not fail because the algorithm decided quickly to assign goals in navigation. Many times, in the RRT algorithm, the robot does not move and stop for a period.

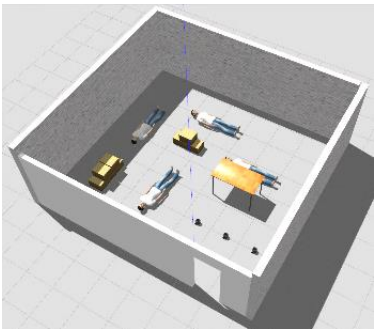


Figure 14. Gazebo shows the small warehouse model

Figure 15- a. Rviz shows the RRT exploration for area coverage by swarm robots for simulation 1 in a small environment, the total merged map for swarm robots, and the maps for each robot.

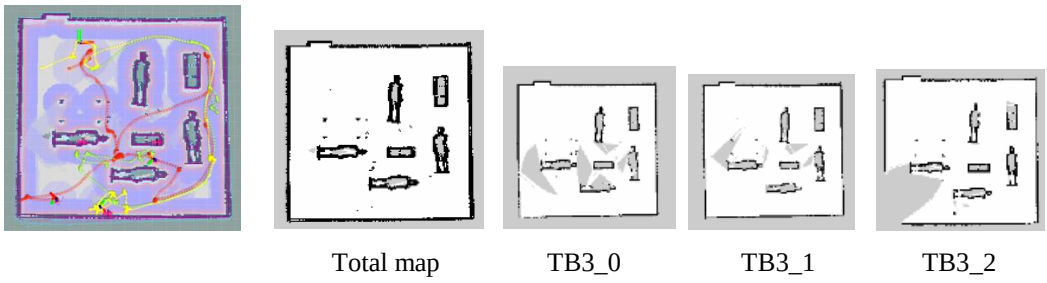


Figure 15-b. Rviz shows the POA exploration for area coverage by swarm robots for simulation 1 in a small environment, the total merged map for swarm robots, and the maps for each robot.

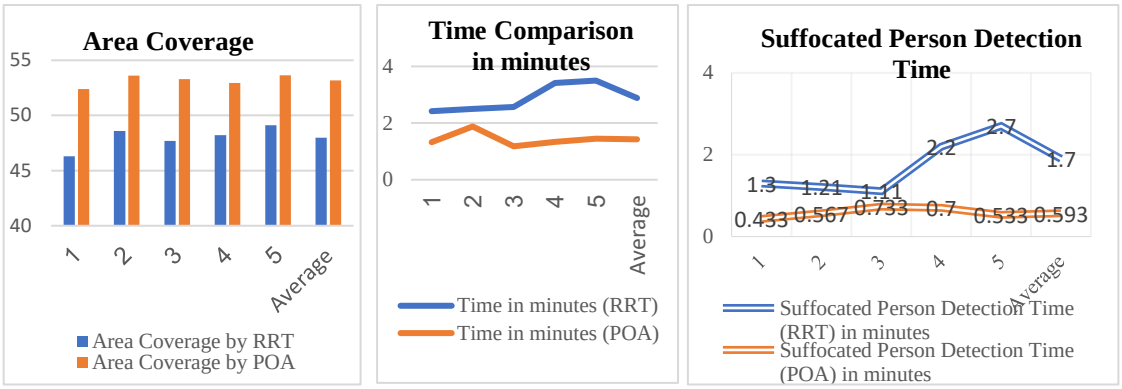
Table 6 and Fig. 16 compare RRT and POA exploration efficiency and suffocated person recognition over five simulations. POA continually exceeds the competition with 53.17 m^2 of coverage despite its 99.14% coverage ratio and RRT's 47.98 units. RRT explores for 2.88 minutes per simulation, compared to POA's 1.43 minutes, demonstrating a significant efficiency gain. Both systems accurately detected four suffocation incidents in each simulation, which were tracked. Algorithms took different times to detect suffocation. POA found suffocated persons in an average of 0.59 minutes, but RRT took 1.9 minutes.

Table 6: Comparison of RRT exploration versus pelican exploration in a small environment

Simulation number	Robot number	Robot position	Area Coverage by RRT Exploration						Area Coverage by POA Exploration					
			Area coverage	Area ratio	Time in minutes	Suffocated person detection count	Time of detection suffocated person in minutes	failer robots (stop for a period).	Area coverage (53.62)	Area ratio	Time in minutes	Suffocated person detection count	Time of detection suffocated person in minutes	failer robots (stop for a period) .
1	0	(-1.5,-1.0)	46.3	86%	2.42	4	1.3	No	52.4	97.7 %	1.33	4	0.433	No
	1	(0.5,-1.0)						Yes						No
	2	(-0.5,-1.0)						No						No
2	0	(-1.5,-1.0)	48.6	91%	2.5	4	1.21	Yes	53.61	99.9 %	1.88	4	0.567	No
	1	(0.5,-1.0)						No						No
	2	(-0.5,-1.0)						No						No
3	0	(-1.5,-1.0)	47.7	89%	2.57	4	1.11	Yes	53.3	99.4 %	1.18	4	0.733	No
	1	(0.5,-1.0)						No						No
	2	(-0.5,-1.0)						Yes						No
4	0	(-1.5,-1.0)	48.2	89.8 %	3.41	4	2.2	No	52.95	98.7 %	1.34	4	0.7	No
	1	(0.5,-1.0)						No						No
	2	(-0.5,-1.0)						Yes						No
5	0	(-1.5,-1.0)	49.1	91.5 %	3.5	4	2.7	No	53.63	100 %	1.45	4	0.533	No
	1	(0.5,-1.0)						Yes						No
	2	(-0.5,-1.0)						No						No
Average			47.98	89.46	2.88	4	1.9		53.17	99.14	1.43	4	0.59	

Due to its rapid detection time, POA may be useful in urgent situations. Multiple factors drive RRT and POA to function differently. POA's optimization methods can speed up area coverage, identify important events like suffocating persons, and improve path planning and decision-making. RRT, on the other hand, might have trouble finding the best paths, which causes the search to take longer and causes it to miss some important events. POA can plan and explore more during normal and emergency tasks. There are many features of POA in small environments like:

- POA's wider scope shows more precise navigation and more environmental coverage.
- POA finds and reacts to critical events (for example, finding people who have been suffocated faster than other methods). POA algorithms may better absorb environmental data and recognize critical qualities, making it more dependable than RRT, which had some failures. This dependability is needed for continuous operation, wide exploration, and incident detection. These findings underline the importance of real-world algorithm selection. POA optimizations can speed up crucial event responses and improve exploration efficiency. Its significance lies in its wider coverage, faster exploration, and early suffocation detection. It is important to select algorithms based on operational needs, where POA can improve exploration efficiency and response time to critical events.



(a) (b) (c)

Figure 16. Comparison of RRT versus POA, a: Area coverage, b: Exploration time, c: Suffocated person detection time

5.2.3 Large warehouse environment

Three TB3B robots implemented the exploration with YOLOv8n detection in a large complicated warehouse with an area of 10.88 *13.7 m^2. The warehouse included many boxes, shelving cupboards, and tables as obstacles, and eight persons on the ground suffocated from warehouse pollution. Fig. 17 shows the model of the large warehouse in Gazebo. More than 100 runs were done for both the RRT exploration algorithm and the proposed pelican algorithm and Table 7. shows the five runs that approximate the average of these runs. Simulation 1 in Table 7. is illustrated in Figures 18 in a and b, respectively, with details of map merging. The fast in exploration is very important, especially for large environments that represent a challenge because of the needed time and the intensive resources. This large environment contains many obstacles. The robots using the proposed pelican algorithm achieved good results and outperformed RRT exploration in the same criteria for a small area.

Figure 17. Gazebo shows the large warehouse model

Figure 18 – a. Rviz shows the RRT exploration for area coverage by swarm robots in a large environment, b: Total merged map for swarm robots and the maps for each robot.

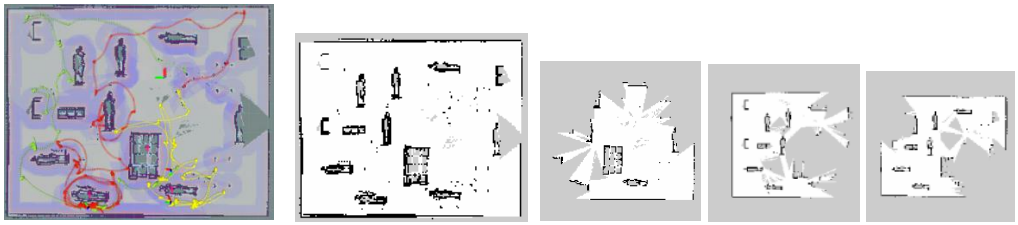


Figure 18 – b. Rviz shows the POA exploration for area coverage by swarm robots in a large environment, the total merged map for swarm robots, and the maps for each robot

Table 7. and Fig. 19 compare RRT and POA in five simulations for area coverage, exploration length, and suffocated human detection in a scenario. POA has a coverage ratio of 98.2% and an average area coverage of 146.46 m^2, whereas RRT has 136.42 m^2 and 91.48 %. POA explores faster than RRT, taking 6.04 minutes for each simulation against 10.74 minutes. Both algorithms identify eight suffocation victims in different simulations at various times. The average RRT detection time is 5.24 minutes, while POA took 2.36 minutes. Performance discrepancies are caused by many causes. The proposed POA's algorithm speeds up route planning and decision-making, improving area coverage and suffocation response times. Complex environments may impede the RRT algorithm, resulting in slower replies and less exploration. The performance of the proposed POA algorithm differs from that of the RRT algorithm in a large environment depending on its numerous features.

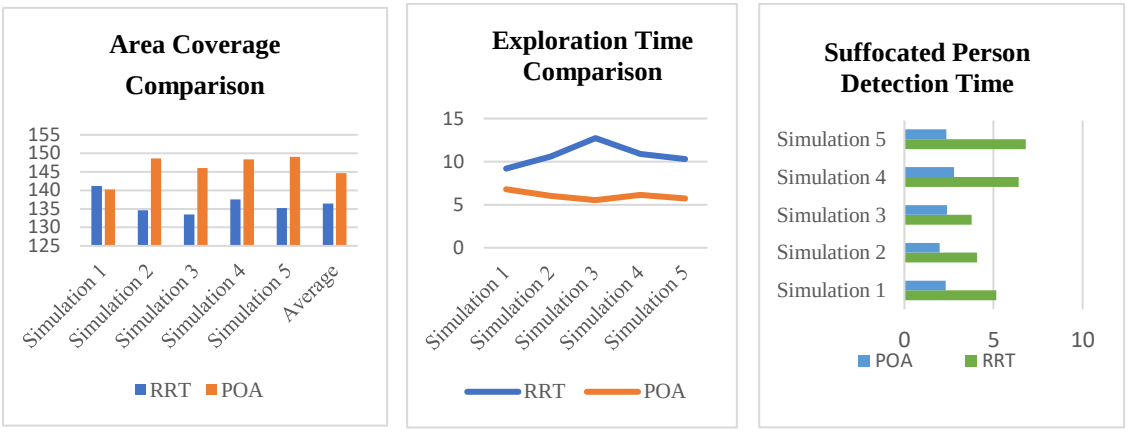
- It enhances path planning, decision-making, area coverage, and suffocation detection. RRT may fail to optimize in large, complicated locations, leading to more exploration and detection.
- It has a high coverage percentage, enabling effective exploration of broad areas. For crisis exploration, POA can examine and chart the area more thoroughly.
- It is more dependable than RRT, for continuous operation, wide exploration, and incident discovery.

Lastly, the proposed POA algorithm can move through large environments and find important events faster than RRT. These results show how important it is to choose algorithms based on practical needs. It can make exploration more efficient and reduce the time it takes to respond to important events.

Table 7: Comparison of RRT exploration versus pelican exploration in a large environment

Simulation number	Robot number	Robot position	Area Coverage by RRT Exploration						Area Coverage by POA Exploration					
			Area coverage	Area ratio	Time in minutes	Suffocated person detection count	Time of detection suffocated person in minutes	failer robots (stop for a period) .	Area coverage (149.05)	Area ratio	Time in minutes	Suffocated person detection count	Time of detection of a suffocated person in minutes	failer robots (stop for a period).
1	0	(-1.5,-1.0)	141.2	94.7 %	9.19	8	5.16	No	140.2	94%	6.79	8	2.32	No
	1	(0.5,-1.0)						Yes						No
	2	(-0.5,-1.0)						No						No
2	0	(-1.5,-1.0)	134.6	90.3 %	10.61	8	4.08	Yes	148.6	99.6 %	6.02	8	1.97	No
	1	(0.5,-1.0)						No						No
	2	(-0.5,-1.0)						No						No
3	0	(-1.5,-1.0)	133.5	89.5 %	12.73	8	3.78	Yes	146	97.9 %	5.53	8	2.39	No
	1	(0.5,-1.0)						No						No

	2	(-0.5,-1.0)						Yes					No
4	0	(-1.5,-1.0)	137.6	92.2 %	10.9	8	6.41	No	148.4	99.5 %	6.14	8	2.79	No
	1	(0.5,-1.0)						Yes						No
	2	(-0.5,-1.0)						Yes						No
5	0	(-1.5,-1.0)	135.2	90.7 %	10.31	8	6.8	No	149.1	100 %	5.72	8	2.35	No
	1	(0.5,-1.0)						Yes						No
	2	(-0.5,-1.0)						No						No
Average			136.42	91.48 %	10.74	8	5.24	8	146.46	98.2 %	6.04	8	2.36	



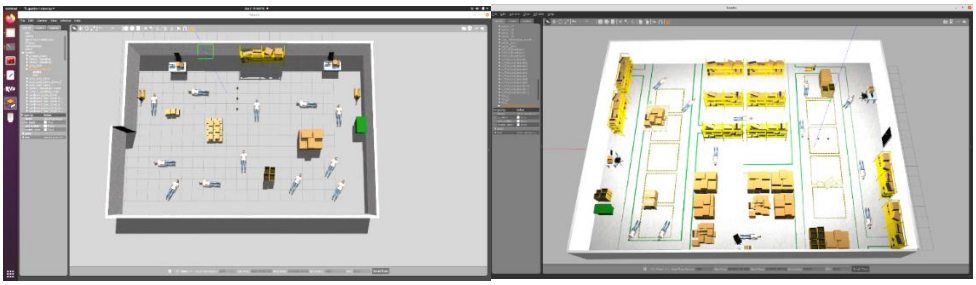
(a) (b) (c)

Figure 19. Comparison of RRT versus POA, a. Area coverage. b. Exploration time. c. Suffocated person detection time

5.2.4 Extra-large warehouse environment

The exploration with YOLOv8n detection was tested for complicated extra-large environments by three TB3B robots. The warehouse included many boxes, shelving cupboards, tables as obstacles, and twelve persons on the ground suffocated from warehouse pollution. The results illustrate that the exploration by the proposed pelican algorithm maximizes area coverage, reducing time consumption. Fig. 20 shows the models of the large warehouse in Gazebo, in (a) model_1 with an area of 16.2 * 24*87 m^2, and in (b) model_2 with an area of 21.54 * 34.42 m^2. After testing this extra-large environment and many other very large and complicated environments, the points of interest are noticed as requirements of the extra-large environment to avoid taking more time as follows:

- It requires significant computational power and memory
- It requires more number of robots
- Testing different types of robots due to specific conditions.



(a) (b)

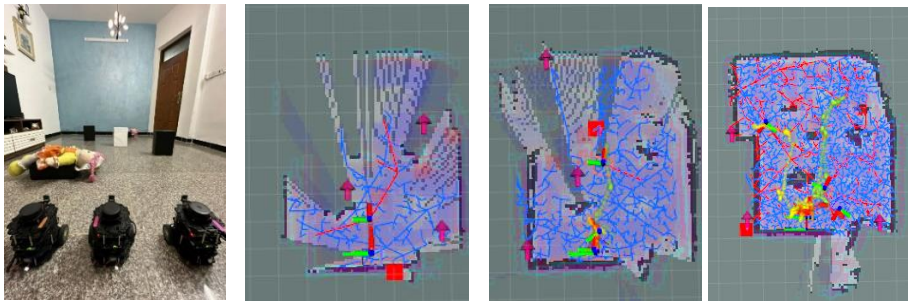
Figure 20. Gazebo shows the extra-large warehouse model

In general, the proposed algorithm outperforms the known RRT exploration algorithm in the following points:

- Less consumption time.
- Maximum area coverage.
- No delay in deciding the next plan and no stop for a period.
- The robot does not fail.
- The assignment of the goal in exploration is close to suboptimal.

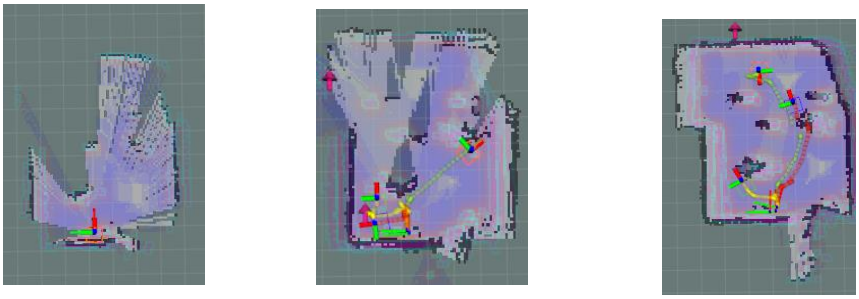
5.2.5 Real-world experiments results

Three TB3B robots implemented the exploration with YOLOv8n detection in a real room environment with an area of 3.5 m * 3.5 m. The room included three boxes as obstacles and two toys on the ground as suffocated persons, Fig. 21. More than 100 runs were done for both the proposed pelican algorithm and the RRT exploration algorithm. Table 8 shows the five runs that approximate the average of these runs. Simulation 1 in Table 8 is illustrated in Fig. 21 and Fig. 22 for the RRT algorithm and proposed POA, respectively. The results illustrated that the exploration by the proposed pelican algorithm produces smooth navigation and reduces randomness through robots moving in this environment.



(1) ⟶ (2) ⟶ (3)

Figure 21. RViz shows the RRT exploration for area coverage by swarm robots for experiment 1 in a real-world environment



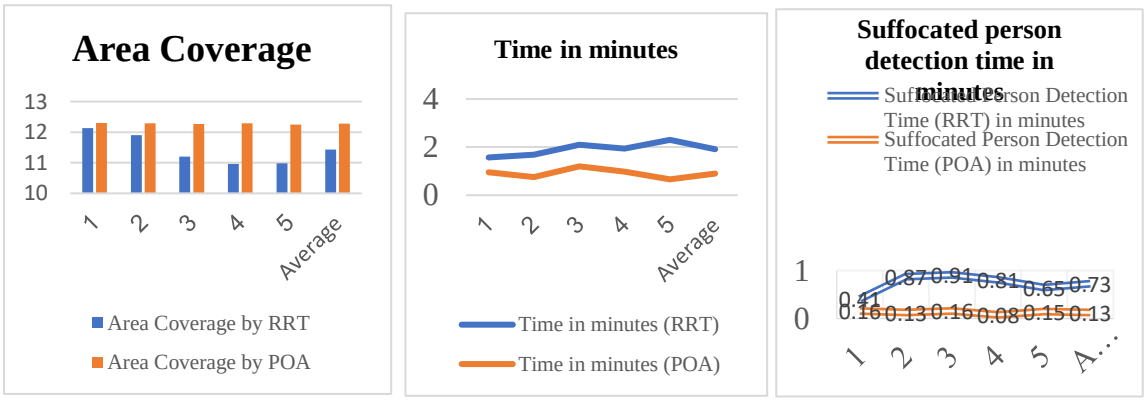
(1) ⟶ (2) ⟶ (3)

Figure 22. RViz shows the proposed POA exploration for area coverage by swarm robots for experiment 1 in a real-world environment

Table 8 and Fig. 23 compare RRT and the proposed POA in five real-world experiments for area coverage, exploration time, and accurate human detection. The proposed POA has a coverage ratio of 100% and an average area coverage of 12.28 m^2, whereas RRT has 93% and 11.43 m^2. The Proposed POA explores faster than RRT, taking 0.9 minutes on average against 1.91 minutes. Both algorithms identify two suffocated persons (assumed toys) in different experiments at various times. The average RRT detection time is 0.73 minutes, while POA is 0.13 minutes. The performance discrepancies are caused by some of the robots were failure to decide where the next position they must move to it by the RRT exploration algorithm and as a result, they still stopped. The proposed POA algorithm speeds up route planning and decision-making, guiding the path toward the next goal depending on algorithm behaviour and environmental features. It improves area coverage because it reduces randomness and has a fast response time for suffocated person's detection.

Table 8: Comparison of RRT exploration versus proposed pelican exploration in a real-world environment

Simulation number	Robot number	Area Coverage by RRT Exploration						Area Coverage by POA Exploration					
		Area coverage (12.25 m^2)	Area ratio	Time in minutes	Suffocated person detection count	Time of detection suffocated person in minutes	failer robots (stop for a period)	Area coverage (12.25 m^2)	Area ratio	Time in minutes	Suffocated person detection count	Time of detection of a suffocated person in minutes	failer robots (stop for a period).
1	1	12.13	99%	1.57	2	0.41	No	12.3	100%	0.95	2	0.16	No
	2						No						No
	3						Yes						No
2	1	11.9	97.1 %	1.68	2	0.87	No	12.29	100 %	0.75	2	0.13	No
	2						No						No
	3						No						No
3	1	11.2	91.4 %	2.1	2	0.91	Yes	12.27	100 %	1.2	2	0.16	No
	2						No						No
	3						No						No
4	1	10.96	89.4 %	1.94	2	0.81	No	12.29	100 %	0.98	2	0.08	No
	2						No						No
	3						Yes						No
5	1	10.98	89.6 %	2.3	2	0.65	No	12.25	100 %	0.66	2	0.15	No
	2						Yes						No
	3						No						No
Average		11.43 m^2	93.3 %	1.91	2	0.73		12.28 m^2	100%	0.9	2	0.13	



(a) (b) (c)

Figure 23. Comparison of RRT versus proposed POA in the real world, a. Area coverage. b: Exploration time. c. Suffocated person detection time

6. Conclusion

The proposed Pelican Optimization Algorithm (POA) for exploration by swarm robots achieved a successful monitoring area and a high ratio for area coverage with minimum time consumption. By focusing on high-revenue points and minimizing unnecessary travel, the proposed POA ensures that the swarm operates in an optimized and coordinated manner. The proposed POA provided a solution by addressing the shortcomings of earlier exploration algorithms. Specifically, POA reduces exploration time by approximately 50% compared to RRT and achieves area coverage ratios of 98%, 99%, and 98% in empty, small, and large environments, respectively. In contrast, RRT achieves 93%, 89%, and 92% in the same scenarios.

POA consistently outperforms RRT in featureless scenarios without clear landmarks and barriers. This highlights the importance of selecting algorithms based on environmental characteristics and the advantages of optimization-driven strategies in robotic exploration. Furthermore, integrating the lightweight YOLOv8n model with ROS and camera assistance enables real-time, accurate object detection without overloading system resources. This enables precise and efficient goal localization by the robots. Using ROS's modular architecture and visualization tools gives further support for the efficient deployment and monitoring of the system in physical environments. The TurtleBot3 Burger platform, equipped with a high-precision LiDAR sensor and Raspberry Pi, serves as an effective testbed for real-world implementation, offering flexibility through support for various programming languages. The overall performance of the proposed strategy demonstrates its suitability for broader applications, including exploration of unknown environments and real-time object detection.

Conflicts of Interest: The authors declare no conflict of interest.
Author Contributions: Author 1: Data collection, concept, analysis, methodology, writing—original draft preparation, software, and writing—review, and editing. Author 2: The supervision, validation, review, and investigation. Author 3: Supervision, validation, investigation.

References

[1] M. McNeill and D. Lyons, "An approach to fast multi-robot exploration in buildings with inaccessible spaces," in *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Dali, China, 2019, pp. 660-667, doi: 10.1109/ROBIO49542.2019.8961681.

[2] D. K. A. T. Sadiq, D. Muhsen, and F. A. Raheem, "A systematic review of rapidly exploring random tree RRT algorithm for single and multiple robots," *Cybern. Inf. Technol.*, vol. 24, no. 1, 2024.

[3] J. Bae and M. Park, "A heuristic for efficient coordination of multiple heterogeneous mobile robots considering workload balance," *IEEE Robot. Autom. Lett*, vol. 6, pp. 4064–4070, 2021.

[4] M. Manoharan, A. N. Shridhar, V. Y. Vinod, and S. Kumaraguru, "A novel volume decomposition methodology for multi-robots collaborative additive manufacturing," in *Proc. 2020 IEEE 4th Conf. Inf. Commun. Technol. (CICT)*, Chennai, India, 2020, pp. 1–6.

[5] M. Saboia *et al.*, "ACHORD: Communication-aware multi-robot coordination with intermittent connectivity," *IEEE Robot. Autom. Lett*, vol. 7, pp. 10184–10191, 2022.

[6] M. Patchou, B. Sliwa, and C. Wietfeld, "Flying robots for safe and efficient parcel delivery within the COVID-19 pandemic," in *Proc. 2021 IEEE Int. Syst. Conf. (SysCon)*, Vancouver, BC, Canada, 2021, pp. 1–7.

[7] Y. Chang *et al.*, "LAMP 2.0: A robust multi-robot SLAM system for operation in challenging large-scale underground environments," *IEEE Robot. Autom. Lett*, vol. 7, pp. 9175–9182, 2022.

[8] F. Zitouni, S. Harous, and R. Maamri, "A distributed approach to the multi-robot task allocation problem using the consensus-based bundle algorithm and ant colony system," *IEEE Access*, vol. 8, pp. 27479–27494, 2020.

[9] A. Ranjha, G. Kaddoum, and K. Dev, "Facilitating URLLC in UAV-assisted relay systems with multiple-mobile robots for 6G networks: A prospective of Agriculture 4.0," *IEEE Trans. Ind. Informat.*, vol. 18, pp. 4954–4965, 2022.

[10] A. Dutta, S. Roy, O. P. Kreidl, and L. Bölöni, "Multi-robot information gathering for precision agriculture: Current state, scope, and challenges," *IEEE Access*, vol. 9, pp. 161416–161430, 2021.

[11] S. A. Alsaidi, D. K. Muhsen, and S. M. Ali, "Improved scatter search algorithm based on meerkat clan algorithm to solve NP-hard problems," *Period. Eng. Nat. Sci.*, vol. 8, no. 3, pp. 1555–1565, 2020, doi: 10.21533/pen.v8i3.1563.

[12] T. K. Kaiser *et al.*, "ROS2SWARM - A ROS 2 package for swarm robot behaviors," in *2022 Int. Conf. Robot. Autom. (ICRA)*, 2022, doi: 10.1109/ICRA46639.2022.9812417.

[13] B. A. Pappas, "Multi-robot frontier based map coverage using the ROS environment," M.S. thesis, Auburn Univ., Auburn, AL, USA, 2014. [Online]. Available: https://etd.auburn.edu/handle/10415/4058

[14] T. Horelican, "Utilizability of Navigation2/ROS2 in highly automated and distributed multi-robotic systems for industrial facilities," *IFAC-PapersOnLine*, vol. 55, no. 4, pp. 109–114, 2022, doi: 10.1016/j.ifacol.2022.06.018.

[15] S. A. Alsaidi, D. K. Muhsen, and S. M. Ali, "Improved scatter search algorithm based on meerkat clan algorithm to solve NP-hard problems," *Period. Eng. Nat. Sci.*, vol. 8, no. 3, pp. 1555–1565, 2020, doi: 10.21533/pen.v8i3.1563.

[16] Y. Gao, L. Zhang, J. Zhou, W. Yuan, and Y. Qiu, "Improved extreme learning machine based on adaptive dual-strategy optimization algorithm and its application," *Res. Sq.*, 2022, Preprint, doi: 10.21203/rs.3.rs-2293384/v1.

[17] K. Janavi and A. R. Teja, "Robot operating systems (ROS): The fundamentals of ROS and its remarkable performances in the world of drones," *Int. J. Res. Appl. Sci. Eng. Technol.*, vol. 10, no. 9, pp. 1844–1849, 2022, doi: 10.22214/ijraset.2022.46938.

[18] E. Salinas-Avila *et al.*, "Assistant delivery robot for nursing home using ROS: Robotic prototype for medicine delivery and vital signs registration," in *Proc. 5th Int. Conf. Electron., Commun. Control Eng.*, 2022, pp. 141-148.

[19] S. T. Pramod Thale, M. Mangesh Prabhu, P. Vinod Thakur, and P. Kadam, "ROS based SLAM implementation for autonomous navigation using Turtlebot," *ITM Web Conf.*, vol. 32, p. 01011, 2020, doi: 10.1051/itmconf/20203201011.

[20] A. Franchi, L. Freda, G. Oriolo, and M. Vendittelli, "A decentralized strategy for cooperative robot exploration," in *Proc. 1st Int. Conf. Robot Commun. Coord.*, Athens, Greece, 2007, pp. 1–8.

[21] Y. Zhou *et al.*, "A PSO-inspired multi-robot map exploration algorithm using frontier based strategy," *Int. J. Syst. Dyn. Appl.*, vol. 2, no. 2, pp. 1-13, Apr. 2013, doi: 10.4018/ijsda.2013040101.

[22] H. Umari and S. Mukhopadhyay, "Autonomous robotic exploration based on multiple rapidly-exploring randomized trees," in *2017 IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2017, pp. 1396-1402.

[23] T. B. Zeng and B. Si, "Mobile robot exploration based on rapidly-exploring random trees and dynamic window approach," in *2019 5th Int. Conf. Control, Autom. Robot. (ICCAR)*, Beijing, China, 2019, pp. 51-57, doi: 10.1109/ICCAR.2019.8813489.

[24] P. Indraneel and R. Zheng, "Graph-based simultaneous coverage and exploration planning for fast multi-robot search," *arXiv: 2303.02259v1 [cs.RO]*, Mar. 2023, doi: 10.48550/arXiv.2303.02259.

[25] R. Z. Khaleel *et al.*, "Improved trajectory planning of mobile robot based on pelican optimization algorithm," *J. Eur. Syst. Autom.*, vol. 57, no. 4, 2024.

[26] C. Q. Li, Z. F. Jiang, and Y. P. Huang, "Multi-strategy improved pelican optimization algorithm for mobile robot path planning," *Inf. Technol. Control*, vol. 53, no. 2, pp. 372-389, 2024.

[27] F. Gul *et al.*, "Novel implementation of multi-robot space exploration utilizing coordinated multi-robot exploration and frequency modified whale optimization algorithm," *IEEE Access*, vol. 9, pp. 22774–22787, 2021, doi: 10.1109/ACCESS.2021.3055852.

[28] S. D. S. G. Seyed *et al.*, "Improved pelican optimization algorithm for solving load dispatch problems," *Energy*, vol. 289, p. 129811, 2024, doi: 10.1016/j.energy.2023.129811.

[29] P. Trojovský and M. Dehghani, "Pelican optimization algorithm: A novel nature-inspired algorithm for engineering applications," *Sensors*, vol. 22, no. 3, p. 855, 2022, doi: 10.3390/s22030855.

[30] Q. Xiong, J. She, and J. Xiong, "A new pelican optimization algorithm for the parameter identification of memristive chaotic system," *Symmetry*, vol. 15, no. 6, p. 1279, 2023, doi: 10.3390/sym15061279.

[31] J. He *et al.*, "Study on reservoir optimal operation based on coupled adaptive ε constraint and multi-strategy improved pelican algorithm," *Sci. Rep.*, vol. 13, no. 14093, 2023, doi: 10.1038/s41598-023-41447-0.

[32] S. B. Neamah and A. Karim, "A real-time traffic monitoring system based on deep learning and YOLOV8," *ARO-The Sci. J. Koya Univ.*, vol. 11, no. 2, pp. 137–150, 2023, doi: 10.14500/aro.11327.

[33] Z. A. Ahmed and S. M. Raafat, "An extensive analysis and fine-tuning of Gmapping's initialization parameters," *Int. J. Intell. Eng. Syst.*, vol. 16, no. 3, pp. 126–138, 2023.

[34] F. Martínez, "TurtleBot3 robot operation for navigation applications using ROS," *Tekhnê*, vol. 18, p. 19, 2021.

[35] D. Abhayankar and D. K. Sanjay Tanwani, "Exploring object detection algorithms and implementation of YOLOv7 and YOLOv8 based model for weapon detection," *Int. J. Intell. Syst. Appl. Eng.*, vol. 12, no. 3, pp. 877–886, 2024.