Robotica (2024), **42**, pp. 977–1000
doi:10.1017/S0263574723001819

CAMBRIDGE UNIVERSITY PRESS

RESEARCH ARTICLE

Self-adaptive learning particle swarm optimization-based path planning of mobile robot using 2D Lidar environment

Julius Fusic S.[1] and Sitharthan R.[2]

[1]Department of Mechatronics Engineering, Thiagarajar College of Engineering, Madurai, Tamil Nadu, India and [2]School of Electrical Engineering, Vellore Institute of Technology, Chennai, Tamil Nadu, India
Corresponding author: Julius Fusic S.; Email: erjuliusfusic@gmail.com

Received: 30 June 2023; **Revised:** 13 October 2023; **Accepted:** 10 December 2023; **First published online:** 26 January 2024

Keywords: Self-adaptive learning particle swarm optimization; 2D Lidar; Hector SLAM; logistic application; mobile robot

Abstract

The loading and unloading operations of smart logistic application robots depend largely on their perception system. However, there is a paucity of study on the evaluation of Lidar maps and their SLAM algorithms in complex environment navigation system. In the proposed work, the Lidar information is finetuned using binary occupancy grid approach and implemented Improved Self-Adaptive Learning Particle Swarm Optimization (ISALPSO) algorithm for path prediction. The approach makes use of 2D Lidar mapping to determine the most efficient route for a mobile robot in logistical applications. The Hector SLAM method is used in the Robot Operating System (ROS) platform to implement mobile robot real-time location and map building, which is subsequently transformed into a binary occupancy grid. To show the path navigation findings of the proposed methodologies, a navigational model has been created in the MATLAB 2D virtual environment using 2D Lidar mapping point data. The ISALPSO algorithm adapts its parameters inertia weight, acceleration coefficients, learning coefficients, mutation factor, and swarm size, based on the performance of the generated path. In comparison to the other five PSO variants, the ISALPSO algorithm has a considerably shorter path, a quick convergence rate, and requires less time to compute the distance between the locations of transporting and unloading environments, based on the simulation results that was generated and its validation using a 2D Lidar environment. The efficiency and effectiveness of path planning for mobile robots in logistic applications are validated using Quanser hardware interfaced with 2D Lidar and operated in environment 3 using proposed algorithm for production of optimal path.

1. Introduction

Mobility robots have a wide range of realistic applications, including logistic application, due to their competence to maneuver in their environment and do intelligent tasks autonomously. Simultaneous localization and mapping (SLAM) are a crucial enabling technology that enables the robot to concurrently map its surroundings and estimate its own position using onboard sensors. Real-time path planning can be done with SLAM technology to complete challenging maneuvering tasks in smart logistic applications. Path planning is a fundamental task for autonomous mobile robots to navigate in dynamic and unknown environments. The objective of path planning is to generate a feasible and optimal path from the start to the goal location while avoiding obstacles and adhering to the robot's dynamics constraints. Various optimization algorithms have been proposed to solve the path planning problem, including the Particle Swarm Optimization (PSO) algorithm. However, the conventional PSO algorithm suffers from premature convergence and poor global search capability. To overcome these limitations, we propose a SALPSO-based path planning algorithm that incorporates self-adaptation and obstacle avoidance. The aim of this research is to find the most effective path in a Lidar-mapped obstacle environment. The Improved Self-adaptive learning particle swarm optimization (ISALPSO) algorithm is used to reduce

the complex environment with multiple obstacles for finding viable path. The key highlights of this research are listed below.

- The suggested work is proposed to build a Hector SLAM algorithm utilizing the RVIZ platform to generate a 2D Lidar map and gathers 2D LiDAR global data for autonomous vehicles localized in complex environments, and it integrates the occupancy grid process with the fine-tuning of information obtained from the Hector SLAM technique to successfully improve mobile robot localization. On the basis of this, hardware is created to gather multipoint global data from a complicated environment, correct it using the binary occupancy grid method, and then transmit the corrected data back to the ROS platform for real-time navigation.
- The obstacle-free path in a 2D environment is found using the suggested ISALPSO method and its other PSO variant algorithms to predict the path that will get to the target quickly and without violating the obstacle.
- The optimal route is found using the proposed SALPSO technique, which dynamically modifies variables such as population size, mutation rate, acceleration coefficients, and inertia weight.
- The effectiveness, computational cost, path length, and smoothness of the route are confirmed for various scenarios using statistical analysis and compared to other alternatives.
- The Quanser Kobuki (Qbot2) robot platform, which relies on Lidar, is used for experimental validation of the proposed model and simulation. This is done in order to build the hardware required to confirm in real time that the path predicted by the proposed algorithm is better than other variations, as shown by the simulation.

The sections of the paper are organized as follows: Section 2 presents the related work on the path planning algorithm and SLAM methods. Using an occupancy grid, Section 3 builds a two-dimensional ecosystem for generated Lidar map. Section 4 provides details on the SALPSO heuristic path planning implementation are discussed. Results and analysis of the simulation are presented in Section 5. Experimentation results are validated in Section 6. Section 7 addresses the conclusion in its future scope.

2. Related works

Route planning techniques can be categorized as either global or local. Global path planning takes into consideration the provided map and searches for the overall transportation route, which is crucial for determining the best path rather than the amount of time needed for path calculation [1]. On the other hand, a local path establishes a route by identifying obstacles in real time using a variety of sensors (such as cameras, LiDAR sensors, laser sensors, ultrasonic sensors, and sensors for sound and heat). According to Li et al. [2], a robot going through an unfamiliar ecosystem can estimate its own pose using position and map data, and it can even create a progressive map as it moves. This allows the robot to execute autonomous obstacle avoidance and navigation. The motion trajectory of the vehicle is composed of the moment coordinates x_1, x_2, and x_r that correspond to the discrete moments $t = 1, 2$, and r that make up the SLAM issue [3]. For the nonlinear system in the actual motion process, the motion model and observation model are as follows:

$$\begin{cases} x_r = f(x_{r-1}, a_r, c_r) \\ J_r = k(x_r, b_r) \end{cases} \tag{1}$$

where x_r stands for robot posture, j_r for the system's observed value, $f(x)$ for the robot motion equation and $k(x)$ for the robot observation equation, a_r for the robot control value, c_r for the system's process noise, and b_r for the system's observation noise. Elnabarawy et al. provide an overview of the SLAM algorithms for autonomous mobile robots and its various algorithms such as EKF-SLAM [4], Hector SLAM, Fast SLAM, and Graph SLAM [5]. Genetic algorithms were suggested as a path planning strategy by Pande et al. for mobile robots [1, 6]. The algorithm generates a set of feasible paths and selects the

https://doi.org/10.1017/S0263574723001819 Published online by Cambridge University Press

best one based on their fitness function. Parker presented a survey of various path planning algorithms for mobile robotics, including grid-based, potential field, heuristic search, and machine learning-based methods [3]. The paper provides a comprehensive review of the state-of-the-art in the field. Peng et al. [7] proposed a path planning approach based on the artificial potential field method, which simulates attractive and repulsive forces acting on the robot to guide it toward the goal while avoiding obstacles. A B Wahab et al. [8] presented a path planning approach for mobile robots in unknown environments using PSO. The algorithm optimizes a cost function based on the distance to the goal and the obstacle avoidance criteria. Arifin et al. [9] proposed a path planning approach using ant colony optimization and fuzzy logic. The algorithm generates a set of paths based on the pheromone trail left by the ants and selects the best one using fuzzy logic.

A comparative analysis of evolutionary algorithms, such as genetic algorithm, PSO, and ant colony optimization, for path planning of mobile robots and analyze the performance of each algorithm based on various metrics [10]. Sharma and Singh [11] proposed a heuristic-based navigation method for mobile robots using a hybrid fuzzy logic and artificial potential field approach. The proposed method combines the advantages of both methods to improve the efficiency and robustness of mobile robot navigation [11]. Hao, He, and Sun [12] presented a new hybrid heuristic method for mobile robot path planning based on an improved artificial potential field algorithm. The proposed method integrates a virtual attractive force and a repulsive force based on an improved potential field algorithm to achieve efficient path planning [12]. Wang, Zhao, and Zeng [13] proposed an adaptive artificial potential field algorithm for mobile robot path planning. The proposed method adapts to the environment and the robot's motion characteristics to achieve efficient path planning in complex environments [13]. Gao, Xu, Zhang, and Huang [14] presented a path planning method for mobile robots based on an improved ant colony optimization algorithm. The proposed method enhances the ant colony optimization algorithm by introducing a local search strategy and pheromone evaporation mechanism to achieve efficient path planning [14]. Wang and Wu [15] proposed a novel path planning method for mobile robots based on the A* algorithm and artificial potential field. Fusic et al. proposed the classical algorithms like RRT* [16] are widely used in 3D environment navigation approach [15].

Liu and Zhang [17] propose an improved potential field algorithm for mobile robot path planning. The algorithm includes a repulsion field generated by obstacles and an attraction field generated by the goal, and the weights of these two fields are adjusted using a variable gain factor to improve the path planning performance [17]. Hussain et al. [18] present a hybrid path planning algorithm based on bat-inspired algorithm and ant colony optimization for mobile robots. The algorithm generates a set of potential paths using the bat-inspired algorithm and then optimizes them using ant colony optimization to obtain the optimal path. Wu et al. [19] propose an improved artificial potential field algorithm for mobile robot path planning. The algorithm uses a Gaussian function to model the repulsion field generated by obstacles and a sigmoid function to model the attraction field generated by the goal, which results in smoother and more efficient path planning [19].

Ren et al. [20] introduce a novel artificial potential field method for mobile robot path planning. The method includes a repulsion field generated by obstacles and an attraction field generated by the goal and introduces a curvature correction factor to adjust the direction of the robot to avoid local minima [20]. Al-Atabany et al. [21] propose a hybrid heuristic algorithm for mobile robot path planning in dynamic environments. The algorithm combines the artificial potential field method with the genetic algorithm to generate an optimal path and uses a dynamic obstacle avoidance strategy to handle changes in the environment [21]. This section thoroughly explains the concept of multiple route planning strategies for mobile robots. These approaches are classified into numerous algorithm ways in order to enhance the optimization time, path length, path smoothness, and adaptation of feasibility across varied contexts. These methods include classical, sampling-based, map-based, and heuristic strategies for identifying optimal path. The PSO and its variants have a substantial advantage over other algorithms in terms of adapting to complex situations because the majority of impediments in the real world are tangible [22]. It is evident that the proposed SALPSO method did not need to take a longer computing time or path to complete the task. Figure 1 depicts the pipeline flow diagram for the suggested navigation system.

https://doi.org/10.1017/S0263574723001819 Published online by Cambridge University Press

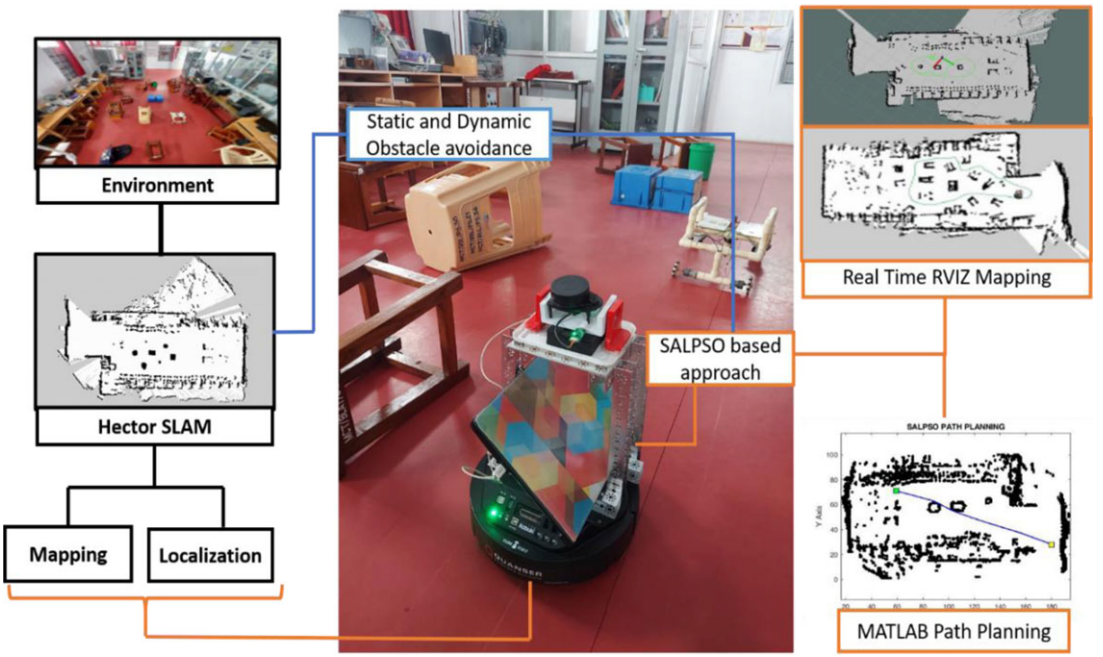

Figure 1. *Pipeline diagram for Lidar-based navigation system.*

3. Environment modeling algorithm

The construction of an environment model is the initial stage in determining a mobile robot's likely path. The proposed SALPSO and other PSO variants are used in logistic mobile applications to determine the best path for loading and unloading operations. For the proposed work, three scenarios with four routes are taken into account. The Lidar-mapped image is then generated into 2D occupancy grids using MATLAB function Binary occupancy grid. The modified grids allow for the interpretation of objects (obstacle) on the map as 1 and spaces as 0. The flow diagram in Fig. 2 shows in detail how the input Lidar map image is converted step-by-step into an occupancy grid. The grid generation technique and its pseudo code are displayed in Table I for the environment algorithm.

The provided pseudocode details the steps required to transform a LiDAR image into an occupancy grid. Using binary Occupancy Map [38], the occupancy grid is prepared, an RGB image is converted to a grayscale image for pixel identification, and thresholds are defined for the range of LiDAR data. The final product is a set of occupancy grid statistics for four distinct environmental configurations. Here, a range threshold is set to classify LiDAR points. Points with a range between 90 and 100 are likely to be treated differently in the occupancy grid generation. To make identifying individual pixels easier, you've apparently taken the next step of transforming a color (RGB) image into a grayscale image. This nesting loop traverses the Y axis of the occupancy matrix, where j is the row index. This line determines whether the given grid cell $[i, j]$ is occupied or not. It assigns the values i and j to x_{obs} and y_{obs} and 0.8 to r_{obs}.

4. Improved self-adaptive learning (SALPSO) methodology

The population-based stochastic search method known as PSO is used to solve optimization problems in multidimensional space and has been widely used in scientific and technical fields [23]. The existing PSO variant universality, or their capacity to perform well across a range of various fitness environments, remains an issue [24, 39]. The state-of-the-art PSO variants, including Chaotic Cooperative Particle

https://doi.org/10.1017/S0263574723001819 Published online by Cambridge University Press

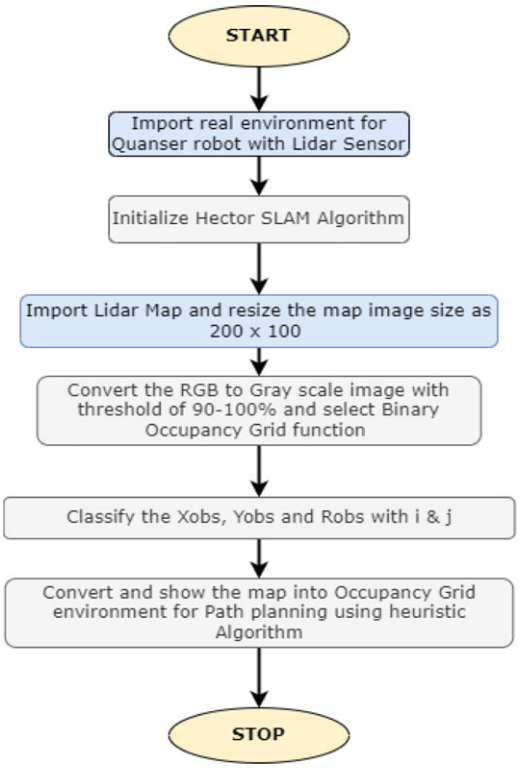

Figure 2. *Flow diagram for Lidar map conversion.*

Swarm Optimization (CCPSO) proposed by Liu et al. [25], are an advanced variant of the PSO algorithm that incorporates chaotic dynamics to enhance its performance, Comprehensive Learning Particle Swarm Optimization (CLPSO) proposed by Zhao et al. [26] is developed to enhance the search and optimization capabilities of PSO, particularly in complex and high-dimensional problem spaces, Local Search Strategy Particle Swarm Optimization (LPSO) proposed by Jiang et al. [27, 28] enhances the exploration and exploitation capabilities of PSO by introducing a local search mechanism. LPSO retains the fundamental components of PSO, including the concept of particles, their positions, velocities, and the social learning process. Each particle in the swarm maintains its position and velocity, updates them based on its own experience, and communicates with neighboring particles to share information about the best solutions found, Velocity Particle Swarm Optimization (VPSO) implemented by Fusic, S. Julius [29, 30] stated the primary distinction of VPSO is its central focus on controlling and optimizing particle velocities. The adjustment of velocities helps particles navigate through the search space effectively, allowing them to escape local optima and converge toward global optima, and Binary Particle Swarm Optimization (BPSO) tailored for discrete optimization problems where variables can take binary values, typically 0 or 1 [31, 32]. BPSO is widely used in various domains, including machine learning, feature selection, and combinatorial optimization to name a few. Even though numerous PSO variants have been put forth, one computation algorithm with resilience and universality, that is, its ability to handle a variety of problems with various features, remains unsatisfactory. In general, the selection of appropriate strategies, especially the choice of the velocity update mechanism, plays a critical role in the success of PSO in solving a particular problem [33–35].

The objective of employing a self-adaptive learning framework is to autonomously select appropriate strategies on various optimization problems and at multiple objective process phases depending on feedback on the fineness of solutions generated via SALPSO proposed by Tao et al. In the proposed

https://doi.org/10.1017/S0263574723001819 Published online by Cambridge University Press

Table I. *Pseudocode Lidar map into occupancy grid.*

Source: ***Import Lidar-mapped point environment***
Source Input: Resize the Lidar map [200, 100]
Output: *Depending on the binary occupancy grid, the environment 1 to 4 data are generated*

```
1   Prepare occupancy grid using binary occupancy map
    Grayscale image synthesis from RGB for
2   pixel identification
3   Set the threshold of the range = 90 –100
4   The Lidar environment image converts to occupancy grid X and Y axis (200 × 100)
5   For (i = 1: 200)
6     For (j = 1: 100)
        c = get occupancy(map, [i,j])
7       set x_obs, y_obs, r_obs as i, j, 0.8
        Increment a= a + 1
8     If a == 1 ; Show (map)
9     End
10  End
```

improved SALPSO derived from the standard PSO algorithm where PSO has two models: G_{best} and P_{best}. The G_{best} model is more oriented toward exploration, while the P_{best} model is more oriented toward utilization. The P_{best} and G_{best} depend on the X_K^i and X_K^j for the learning process but majority of PSO algorithms fall short of efficiently obtaining the optimal solutions, particularly when attempting to solve difficult multi-objective or complex environment optimization issues that exhibit multimodality. The single search approach that obtains learning information only from X_K^i and X_K^j reduces the performance of path planning prediction in real-time data from sensors like Lidar and Kinect sensors. The proposed Improved SALPSO algorithm has a fast convergence rate and avoids being caught by local optima locations. To solve this challenge, the performance of the Pbest and Gbest prediction must be balanced. Instead of immediately updating the current learning information with the monotonic learning strategy in SLPSO, each particle adaptively selects the best appropriate search behavior from a collection of various learning strategies based on its selection ratio of the related operator. In the proposed work, the introduction of improved SALPSO has four operator which is the common difference from the standard PSO algorithm. The proposed SALPSO stochastically decides which approach is used to update the current particle using execution probability. Also, during the entire optimization procedure, avoid employing execution probabilities that are comparatively fixed. In order to adaptively update the execution probabilities depending on the feedback of prior optimization methods, SALPSO employs a gradual updating technique based on a given learning rate (α). Improved SALPSO (ISALPSO) uses a learning rate to regulate the execution probability learning speed during the optimization process. By using a gradual learning mechanism, SALPSO can minimize the influence of fitness attribution of various learning landscapes. The proposed SALPSO-based path planning algorithm consists of the following steps: (1) defining the problem, including the start and goal locations, obstacles in the environment, and robot's dynamics constraints; (2) designing the SALPSO algorithm that generates a feasible path while optimizing the performance criteria and avoiding obstacles; (3) implementing the SALPSO algorithm on the robot's hardware or software and collecting sensor data in real time; (4) generating a path using the SALPSO algorithm based on the real-time sensor data; (5) evaluating the generated path based on the performance criteria and obstacle avoidance; (6) adjusting the SALPSO algorithm's parameters based on the performance of the generated path; and (7) converting the generated path into commands that can be sent to the robot's actuators to move the robot along the path. The psedocode for the proposed ISALPSO algorithm is shown in the Table II.

https://doi.org/10.1017/S0263574723001819 Published online by Cambridge University Press

Table II. *Pseudocode for proposed ISALPSO algorithm.*

Code Input: Create_LIDAR Model ( ) → Occupancy Grid Model (200 × 100)
Input start **and** end **points:** (x_a, y_a, x_b, y_b)**, itr = 100 → Maximum iterations**
Output: ***No. of Points between*** (x_l, y_l, x_u, y_u) **=18 random points plots**

Initialization ISALPSO algorithm parameters
Randomly initialize the position of all n particles k as $n = n_1, n_2, n_3 \ldots .. n_{1000}$
Initialize particles $(n, X_k^i, V_k^i, w_k, \alpha \& Gs)$ where $Gs - Learning\ period$
Set weight of the ISALPSO algorithm as $w_k = \frac{\log(ps-j+1)}{\log(1)+\ldots+\log(ps)}$
Calculate *Violation of Particle* $\varnothing_k =$ and validation of all generated particles
Set local best (P_{best}) for generation of each iterated particle → kth particle local best.
Choose the ith approach using a roulette wheel based on the kth particle's $\boldsymbol{Tp'_k}$
Analyze the new particle n_k fitness value,
Calculate the selection ratio and the reward point for each iteration with the previous one.
If $(\boldsymbol{P_{best} > G_{best}})$ **then replace the** $(\boldsymbol{G_{best} = P_{best}})$ → set next iteration
Update the optimal particle based on violation $\varnothing_k == \mathbf{0}$**.** $\boldsymbol{G_{best}}$
$\boldsymbol{k_{iteration} = 1}$
while $\boldsymbol{k_{current} < Max_{itr}}$ ***or Violation*** $(\varnothing_k)$ ***is zero*** **do**
  Update velocity function $\boldsymbol{V_k^{x,y}}$ as
  $\boldsymbol{V_k^{i+1} = w_k * V_k^i + c_c ran_1(P_{best\_nearest} - X_k^i) + c_s ran_1(P_{best} - X_k^i)}$
  $\boldsymbol{V_k^{i+1} = w_k * V_k^i + c_c ran_1(P_{best\_nearest} - X_k^i) + c_s ran_1(P_{best} - X_k^i)}$
  $\boldsymbol{X_k^{i+1} = X_k^i + V_{average}^i * \mathbf{N}(0, 1)}$
  $\boldsymbol{V_k^{i+1} = W_k * V_k^i + c_c ran_1(G_{best} - X_k^i) + c_c ran_1(G_{best} - X_k^i)}$
  $\boldsymbol{V_{k+1}^i = X_k^{i+1} + V_k^{i+1}}$ **Each** ***k*****th particle update**
  If $\boldsymbol{X_k^i}$ ***is optimal than*** $\boldsymbol{P_{best}}$
    select $\boldsymbol{P_{best}}$ *to be* $\boldsymbol{X_k^i}$
  End if
  If $\boldsymbol{X_k^i}$ ***is superior than*** $\boldsymbol{G_{best}}$
    select $\boldsymbol{G_{best}}$ ***to be*** $\boldsymbol{X_k^i}$
  End if
  Update $\boldsymbol{V_k^i, X_k^i}$ **based on** $\varnothing_k$ ***value***
  $\boldsymbol{i_{iteration} = i_{iteration} + 1}$
end while
return

4.1. Step 1: Initialization

The population or particle count is initially produced at random with n ($n = 1000$). Each particle's initial position (X_k^0) is chosen at random for the environment's coordinate range of (X_a, Y_a) to (X_b, Y_b). The (X_a, Y_a) to (X_b, Y_b) is vary with the hector SLAM generated map images which are bound and converted as (0,0) to (100,200), respectively. Every particle in the population possesses $\langle Position\ of\ particle\ X_k^i | Velocity\ of\ particle\ V_k^i \rangle$ the listed properties are possessed by every particle in the population. In the proposed constrain, the position $\boldsymbol{X_k^i}$ and velocity $\boldsymbol{V_k^i}$ are similar interpretation as in the other PSO algorithm proposed by Tao et al. [34]. The learning rate α can be selected as less than 1 with the particle weight assumed to be W_k for the kth best particle assumed to be $k = 1, 2 \ldots ps$.

$$W_k = \frac{\log(ps - j + 1)}{\log(1) + \ldots + \log(ps)} \tag{2}$$

https://doi.org/10.1017/S0263574723001819 Published online by Cambridge University Press

4.2. Step 2: Fitness function initialization

In order to find the shortest route, the fitness value is essential. Once the particle begins to accelerate and travel at a certain velocity $\boldsymbol{V}_{\boldsymbol{k}}^{\boldsymbol{i}}$, the probability of violation is presumed to be zero [33]. Because it provides the best fitness value, the proposed condition is considered to be the overall optimal option. The best local solution produced by each particle is compared to the best global solution while taking the violation value into account. The execution probability of the kth updating strategy is updated according to the following rule after a predetermined number of generations Gs.

$$Tp'_k = (1-\alpha)\, Tp_k + \alpha \frac{S_k^i}{Gs} \tag{3}$$

$$Tp_k = Tp'_k/(Tp'_1 + I.... + Tp'_n) \tag{4}$$

$$S_k^i = \frac{R_k^i}{\sum_j^n R_k^i} * ((1-n) * S_k^i \max) + S_k^i \min \tag{5}$$

where Tpk is the probability of temporal execution and α represents the learning rate, which regulates the repeating proportion. $\frac{S_k^i}{Gs}$ is assumed to be less than 1 based on the normalized execution probabilities. It is to be assumed that techniques that have produced higher-quality solutions will gradually raise their implementation probabilities. Further the selection ratio (S_k^i) for each learning operator is selected to determine the probability of selecting the updated particle. The (S_k^i) value incremented or decremented for successive iteration k of previous performance results in higher fitness value. The selection ratios of all operators are all set to 0.25 for each particle. The following modified equation will be used to update the selection ratio of the operator of particle k at iteration $i+1$ within the framework of the self-adaptive learning mechanism.

4.3. Step 3: Cost function and iteration

In the continuous loop, sequences used to illustrate the trajectory across the start and finish points, the particle position (x_a, y_a, x_b, y_b), and velocity parameters have been modified for each iteration. Each particle's fitness values ranged for each iteration, and the best position is ranked as the local best (P_{best}). Similar to other PSO variations, the P_{best} and G_{best} are related at the conclusion of each iteration.

4.4. Step 4: Completion of iteration

Between the starting point and the finishing point, the global best position is determined and navigated (x_a, y_a, x_b, y_b) for iteration $i = 1$. The solution is excluded while determining the optimum path if the violation $\varnothing_i$ happens during particle movement. Once the optimal approach to getting there is identified, steps 2–4 are repeated, and pseudocode is shown in Tables II and as the result. Table III displays the ISALPSO method and its variation parameters. The entire simulation framework utilized in the suggested study is shown as a flowchart in Fig. 2. The particle, that represents a significant response to the optimization process, may begin to move around in order to find the best answer in a given search space. The velocity and position of each and every swarm particle at the ith iteration will be modified as follows during the next iteration based on the four learning operators:

$$\text{Operator 1: } V_k^{i+1} = W_k * V_k^i + c_c \text{ran}_1(P_{\text{best}} - X_k^i) + c_s \text{ran}_1(P_{\text{best}} - X_k^i) \tag{6}$$

$$\text{Operator 2: } V_k^{i+1} = W_k * V_k^i + c_c \text{ran}_1(P_{\text{best\_nearest}} - X_k^i) + c_s \text{ran}_1\left(P_{\text{best}} - X_k^i\right) \tag{7}$$

$$\text{Operator 3: } X_k^{i+1} = X_k^i + V_{\text{average}}^i * \text{N}(0, 1) \tag{8}$$

$$\text{Operator 4: } V_k^{i+1} = W_k * V_k^i + c_c \text{ran}_1\left(G_{\text{best}} - X_k^i\right) + c_c \text{ran}_1\left(G_{\text{best}} - X_k^i\right) \tag{9}$$

https://doi.org/10.1017/S0263574723001819 Published online by Cambridge University Press

Table III. *Constriction factors of proposed SALPSO.*

n	Generated particle count	1000
t	Number of iterations	100
K_V	Current velocity factor	0.313
w_k	Inertial weight	$w_k = \frac{\log(ps-j+1)}{\log(1)+\ldots+\log(ps)}$
C_cC_s	Learning coefficient	1.2 & 1.6
α	Learning rate	0.16
S_k^i	Selection Ratio	0.25
Gs	Learning period	10
(x_k, y_k) $K = 1, 2, .18$	The generated trajectory's point	Developed with cubic spline interpolation

where X_k^{i+1}, $P_{\text{best\_nearest}}$ and V_{average}^i are the personal best position of the particle closest to particle k at iteration i and the average velocity of all particles at iteration i, respectively. N (0,1) is a random number taken from a uniform distribution ranging from 0 to 1. δ and C_c are weight and acceleration coefficients that can be used to balance the effect of exploration and exploitation during the search process. In general, w and 3 are set depending on the designer's practical expertise. Based on the working environment, the search behaviors are adjusted and learning strategy locate the nearer local minima point. The X_k^i and V_k^i signify the kth position and velocity at the ith iteration, respectively; P_{best} and G_{best} symbolize the kth particle's optimal positioning (Local best) and the ideal position of the global swarm (Global best) up to iteration i, where P_{best} signifies the best individual particle position, G_{best} denotes the best global particle position, C_C^i and C_S^i are ISALPSO's cognitive and social constant factors, alpha represents the learning rate 0–1 and ran1, ran2 are random numbers ranging from 0 to 1 are created. The C_c, C_s, and number of particles (nP) values for the proposed work are chosen based on variable values ranging from 20 to 100 for particles and 0 to 2 for learning factor, and the maze environment is run for 20 iterations. Based on the optimization, the $nP = 100$, $C_c = 1.1$, and $C_s = 1.7$ the best violation-free path is found.

Considering the violation $\varnothing_{k+1}$ represents the plotted point of particle is out of the using the Eq. (11) to forecast the path,

$$X_k^i = \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2} \tag{10}$$

$$\varnothing_{k+1} = \varnothing_k + \text{mean}\left(1 - \sqrt{(x_k - x_{\text{obs}})^2 + (y_k - y_{\text{obs}})^2}\right)/r_{\text{obs}} \tag{11}$$

where x_{obs} represents the environment obstacle in x coordinate, y_{obs} represents the environment obstacle in y coordinate. The radius of the various structural grid obstacles is represented by r_{obs}. All of the proposed obstacles are shaped using cubic spline interpolation, and the path prediction additionally employs cubic spline interpolation in proposed algorithm.

The proposed SALPSO algorithm constriction factor parameters for proposed environment simulations are listed in Table III. The Learning coefficient and learning rate parameters detailed in the Eq. (3) are used to tune the proposed SALPSO algorithm. In addition to this, the population size of the particles (n) is selected based on 20 experimental simulation results implemented using environment E3 and considered the constriction factor value for proposed SALPSO as shown in Table III for proposed application. These above-mentioned parameters are tuned in the simulated robot to achieve violation-free optimal path to reach the destination. In this model, the violation as ($\varnothing_i$) and the violation-free distance (D) need to achieve this shortest path as shown in environment 3. The values of the input and their response that are used for the experimental design as mentioned in Table III. From the optimal solution in Table III, the proposed SALPSO parameters are fixed as $C_c = 1.1$; $C_s = 1.7$; $\alpha = 0.16$; $n = 1000$, and iteration as 100. The inertial weight was selected based on the equation w_k.

https://doi.org/10.1017/S0263574723001819 Published online by Cambridge University Press

Table IV. *PSO variants comparison for four environments with different routes.*

					Heuristic Approach of Using PSO Variants											
					VCPSO		**BPSO**		**CMOPSO**		**CCPSO**		**LPSO**		**Proposed SALPSO**	
Lidar environments	**Route**	**Starting location ($x1, y1$)**	**Target location ($x2, y2$)**	**Optimal path (m)**	***D*** **(m)**	***T*** **(min)**	***D*** **(m)**	***T*** **(min)**	***D*** **(m)**	***T*** **(min)**	***D*** **(m)**	***T*** **(min)**	***D*** **(m)**	***T*** **(min)**	***D*** **(m)**	***T*** **(min)**
E1	**R1**	(77,77)	(98,20)	**60.7**	73.8	7.2	62.9	6.5	89.2	8	61.5	6.6	63.2	6.8	**61**	6.3
	R2	(125,70)	(66,25)	**74.2**	83.4	7.7	76.8	7	98.5	8.7	76	7.3	78.4	7.4	**74.7**	7.2
	R3	(135,40)	(67,80)	**78.9**	87	8.2	81.2	8	101.4	9	**79.1***	7.8	80.4	8	79.4	7.8
	R4	(80,10)	(120,80)	**80.5**	84.8	8.2	87	8.2	97	8.8	81.2	8	83	8.2	**80.8**	8
E2	**R1**	(45,85)	(155,25)	**128**	134	14	138	15	216	19	130	13.8	132.2	14	**129.5**	13.7
	R2	(155,45)	(70,27)	**86.8**	93.4	9	98	9.5	134	13	89.2	8.8	91	8.9	**87**	8.4
	R3	(140,30)	(42,50)	**100**	109.5	11.9	121	13.8	241	21	108.2	11.8	110.5	12.6	**106.4**	11.7
	R4	(41,50)	(130,25)	**92.4**	97	9.4	99.1	9.5	148.4	11	95.6	9.2	98.1	9.5	**93.6**	9
E3	**R1**	(180,28)	(59,71)	**128.4**	129.8	14.3	130.4	14.6	216.7	18.2	130	14.5	134.3	14.9	**128.8**	14
	R2	(180,69)	(47,35)	**137.2**	145	18.6	147	19	142.5	18.3	140.4	18	141.7	18	**139.4**	17.2
	R3	(140,30)	(65,70)	**85**	99.2	9.7	93.4	9.1	127.5	11.2	97.8	9.4	102.3	9.7	**91.8**	9
	R4	(40,40)	(135,74)	**100.9**	104.4	12.8	109.1	13.7	106.2	13.4	103.4	12.4	107.2	13.2	**102.5**	12
E4	**R1**	(128,20)	(87,80)	**72.6**	79.05	9.5	77.1	9.3	128.6	12.2	74.6	9.6	80.02	10.7	**73.8**	8.4
	R2	(125,83)	(90,40)	**55.5**	63.2	6.8	69	7	74.2	7.8	62.1	6.8	71.5	7.4	**59.3**	6.5
	R3	(85,14)	(123,55)	**55.9**	59.4	6.2	64.2	6.5	76.3	7.2	61.2	6.3	62.7	6.5	**56.5**	5.9
	R4	(120,87)	(50,60)	**75**	79	8.4	86	8.7	98.2	10.2	77.1	8.1	79.5	8.4	**76.2**	7.8

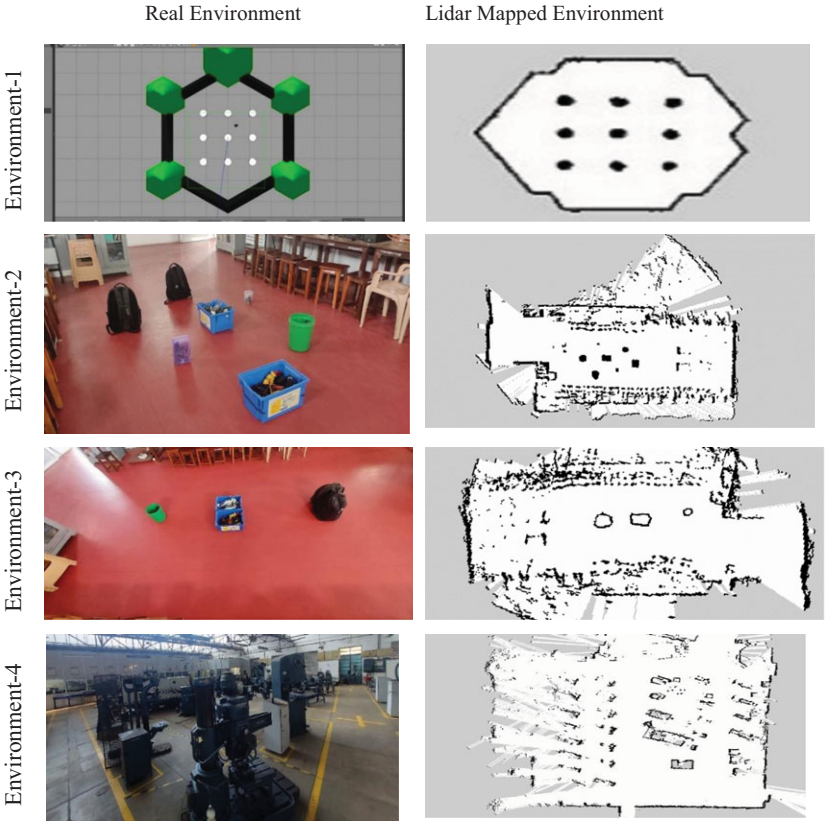

Figure 3. *Environment selected for proposed work and its Lidar mapping.*

5. Results and discussion

The mobile robot Lidar-mapped environment with proposed improved SALPSO, BPSO, LPSO, CCPSO, CLPSO, and VCPSO algorithm were simulated in MATLAB 2022a simulating tool. An Intel(R) Core (TM) i5-6500 CPU, 8 GB of RAM, and 3.20 GHz are used in a PC to run the entire script and simulations [16]. The suggested research examined four distinct environments employing different situations to evaluate the proposed SALPSO optimization approach and its modifications for a Mobile robot navigation strategy. All the four scenarios are shown clearly in Figs. 3 and 4. The Turtle robot environment is shown in detail in Fig. 5 as a binary occupancy grid with a dimension of 200 × 100. According to Fusic et al., Fig. 5 depicts the transformation of a Lidar-mapped environment utilizing Hector SLAM from a navigation system into an occupancy grid for classifying the free-moving path and obstacle, with an accuracy range of 90–100%. The conversion indicated in Fig. 5 is used in scenario 4 to treat each obstacle as 1 and the vehicle's travel path as 0. In the provided topographical surroundings, the shortest route between the logistic points of loading and unloading destination coordinates is (98,20) and (77,77). The optimal shortest straight-line path not considering obstacle is 60.7 m. Out of six PSO variants, the proposed SALPSO travels 61 m to the desired location without any violations. The CCPSO algorithm reaches the destination at a distance of 61.49 m, the LPSO algorithm at a distance of 63.2 m, the VCPSO algorithm at a distance of 73.8 m, the CMOPSO algorithm at a distance of 89.2 m, and the BPSO algorithm at a d value of 62.9 m. As a result, the algorithm's convergence is illustrated in Fig. 5 convergence graph, which shows an obstacle-avoiding trajectory with a smooth curve at each iteration for the proposed algorithm over other variants

https://doi.org/10.1017/S0263574723001819 Published online by Cambridge University Press

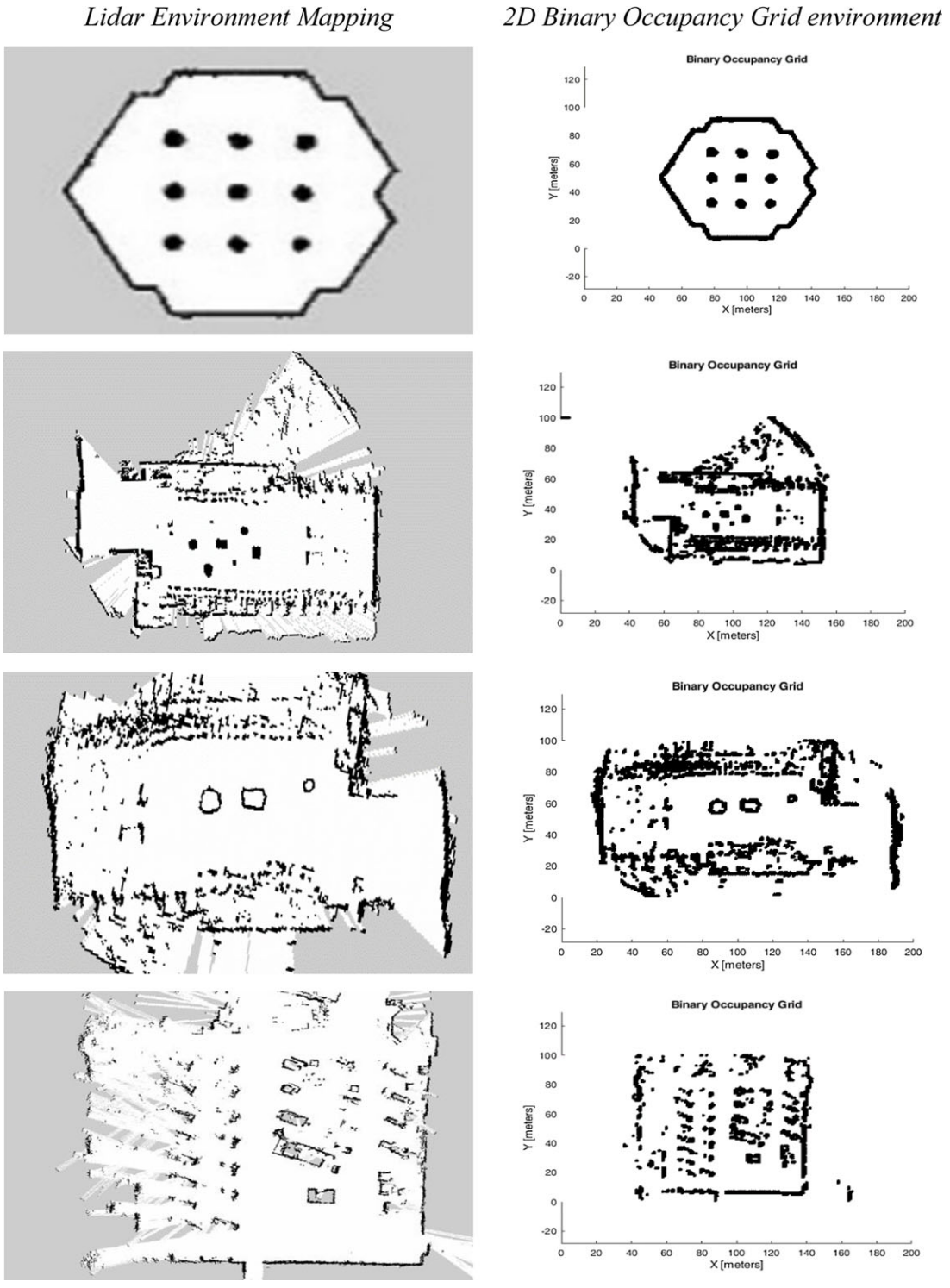

Figure 4. *Proposed binary occupancy grid conversion from Lidar point map.*

https://doi.org/10.1017/S0263574723001819 Published online by Cambridge University Press

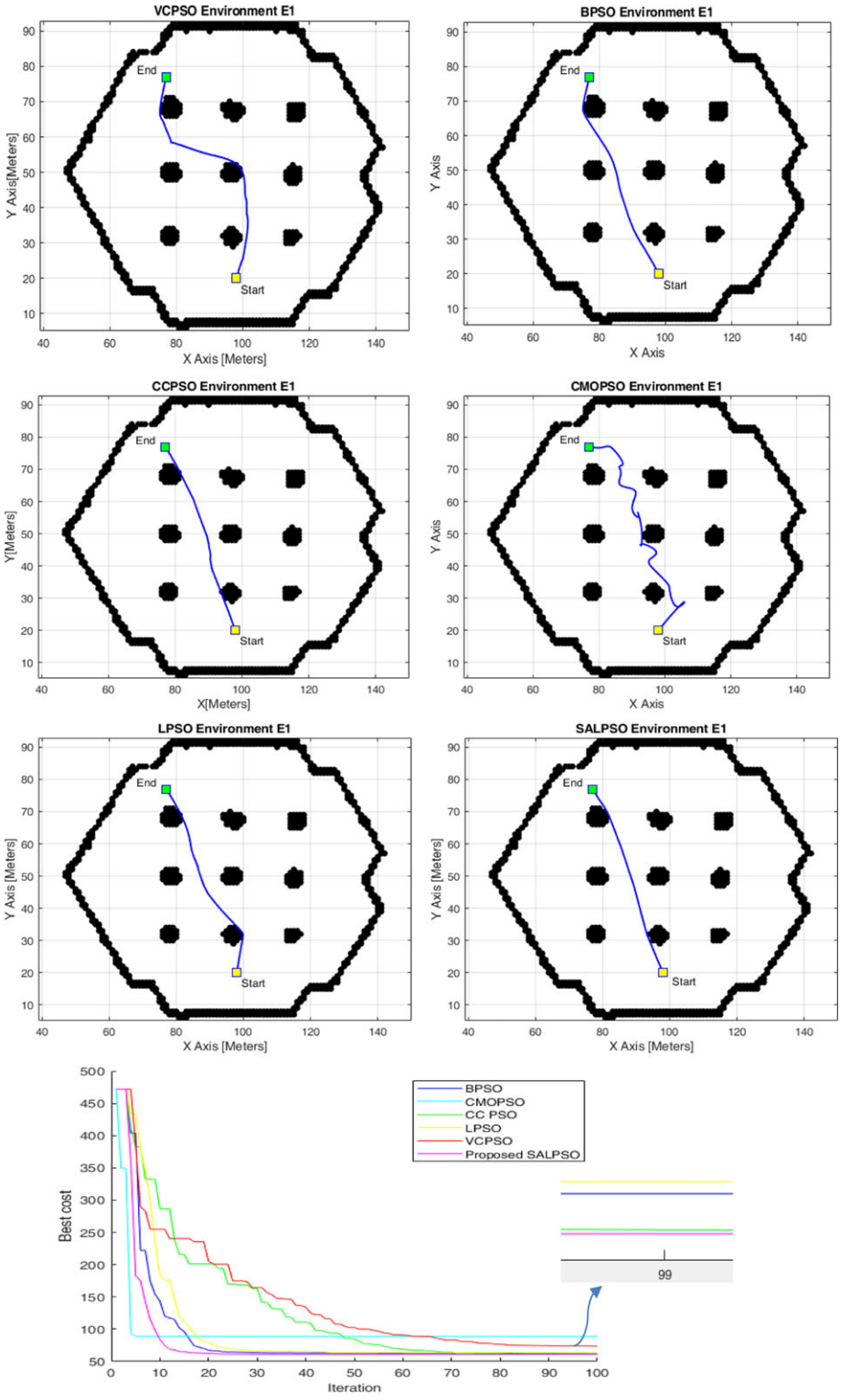

Figure 5. *Optimal heuristic algorithm route for proposed Lidar environment 1 path prediction and its convergence graph.*

https://doi.org/10.1017/S0263574723001819 Published online by Cambridge University Press

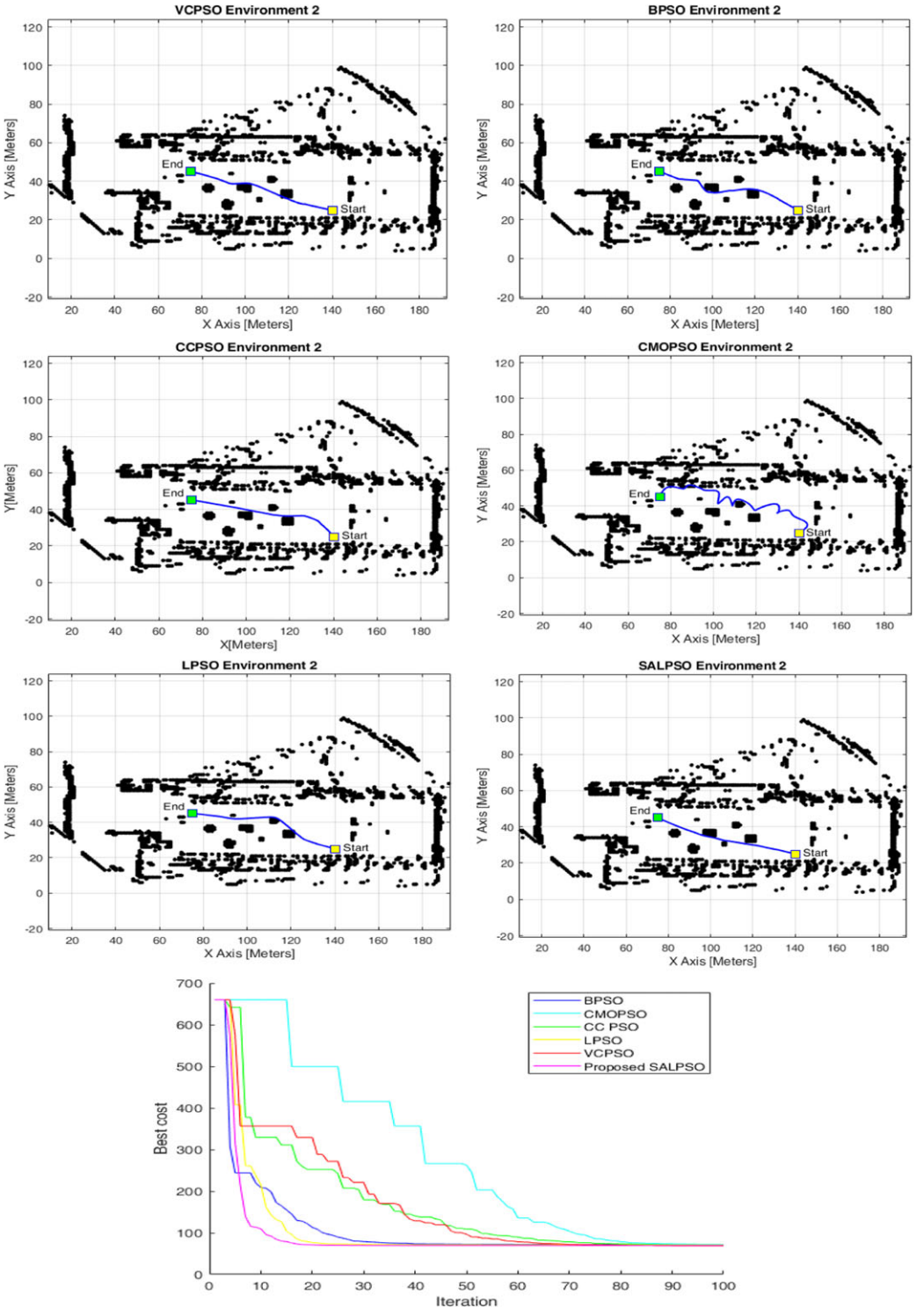

Figure 6. *Convergence graph and proposed heuristic algorithm path for suggested Lidar environment 2 path prediction.*

https://doi.org/10.1017/S0263574723001819 Published online by Cambridge University Press

Figure 5 depicts the heuristic algorithm technique for environment 1 at an occupancy grid rate of 85%. In the given topographical environment, the shortest distance between the logistic loading and unloading point coordinates is (75,45) and (140,25). Without considering the barriers, 68 m is the shortest distance between the two points. It is vital to safeguard the safety of the vehicle even if the goal is to get at the destination quickly and with the fewest number of miles traveled. The SALPSO is substantially more likely to be successful than the other five PSO variations in this area. The prediction of the optimal path in the simulation context of the Lidar SLAM environment is quite challenging for the given obstacles encountered. The suggested heuristic approach thus made it obvious with the proper constriction factor and narrow down the path in fewer iterations making the search for the best path in a smoother manner. The introduction of the cubic spline curve simplified the process of predicting the shortest path in the given environment.

Real-time barriers are more frequent in environment 3 than they are in environment 2. Figure 6 details the optimal path predicted in the environment 2 occupancy grid. The shortest distance is about 128 m from the end user to the logistic mobility terminal. By applying the tried-and-true heuristic technique, the SALPSO outperformed the VCPSO and CCPSO in determining the shortest path, accomplishing the task at 129.5 m with no barrier violations. A pre-processed, binary occupancy grid view of the Lidar map environment is shown in Fig. 7. In the provided environment, the shortest route between the start and end locations, neglecting barriers, is 128 m. In the given environment, the five heuristic approaches are employed to discover the optimum path to exclude violations. The proposed SALPSO optimization path, out of the six PSO variant algorithms, achieves the destination through the stipulated path given by Lidar map generated using Hector SLAM and arrives at 129.5 m without violation. The other approaches get there at distances of 130, 132.2, 134, 138, and 216 m, respectively. The coordinates for the starting point are (180,28) while the point of completion is (59,71). However, the presented scenario contains more breaches than the other situations do. The feasible path was identified using SALPSO at a distance of 73.8 m. The path and barriers for the vehicle in environment 4 were chosen to resembled the industrial environment with several machinery as shown in the Fig. 8. The industrial shop is SLAM mapped in environment 4 to create an industrial environment sample that predicts the machines as obstacles and finds the best route between the machines as a logistical component carrying between two destinations between the machines.

The constriction factors, obstacle collision factor as a violation, and time necessary to complete 100 iterations for each optimal route using the proposed algorithm to calculate the distance between the starting and ending destinations are shown in Table III. Consider the example of loading and unloading a warehouse, where the consignee is the customer who ordered the delivery of the goods. The shortest path in environment 4 is 72 m, independent of an obstacle. The proposed heuristic approach is put through four iterations while the given scenario case is simulated. In the hypothetical situation, the SALPSO, out of six algorithms (at a maximum iteration of 100), arrives at the desired location at a distance of 73.8 m without running short of the impediment. The proposed method combines the advantages of both methods to achieve efficient and robust path planning for mobile robots. In SALPSO, the acceleration coefficients ($c1$ and $c2$) and mutation rate (μ) are updated during the optimization process based on the performance of the swarm. This allows the algorithm to adapt to the problem being solved and improve its search ability. The fitness function used in the algorithm is specific to the problem being solved and represents the objective to be optimized, such as minimizing the distance traveled by the robot or maximizing its coverage of a given area. In order to gather information for a statistical study of the best path, each case scenario from Fig. 9 was run five times. The results are summarized in Table III together with the travel distance and time for each version. Researchers typically employ a parable about the loading and unloading of a warehouse, where the consignee is the customer who placed the delivery order. The process is optimized five times using proposed SALPSO and other PSO variant techniques. There are 5 cost function values in each set of operations.

For environment 1 route 2, the minimum distance is 74.2 m. The six PSO versions of the best algorithms travel from 74.7 to 98.5 m to their final destination. Out of that, the projected SALPSO takes 7 min to go 74.7 miles to its destination. The optimum paths for routes 3 and 4 are 79.4 and 80.8,

https://doi.org/10.1017/S0263574723001819 Published online by Cambridge University Press

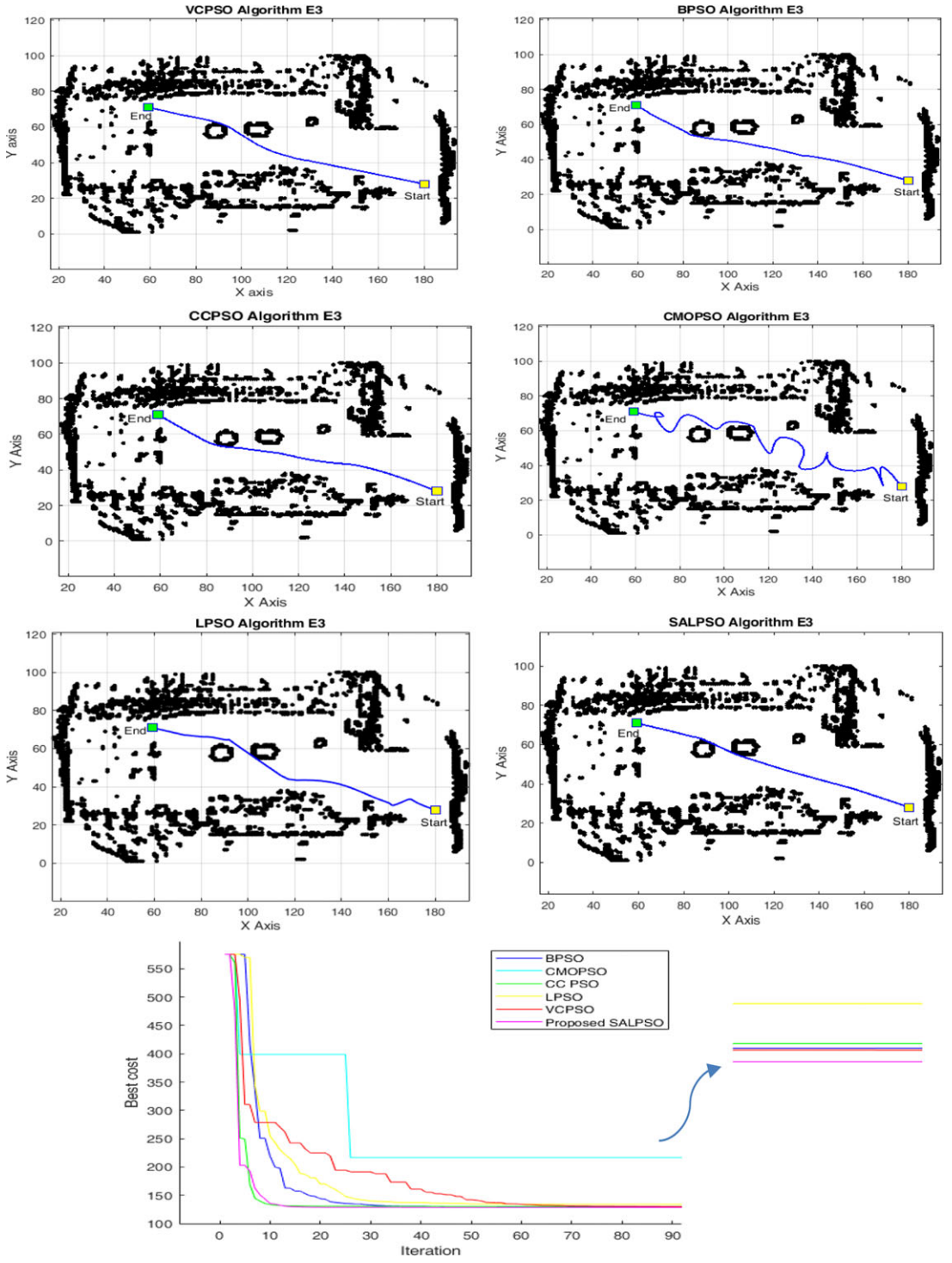

Figure 7. *Convergence graph and proposed heuristic algorithm path for suggested Lidar environment 3 path prediction.*

https://doi.org/10.1017/S0263574723001819 Published online by Cambridge University Press

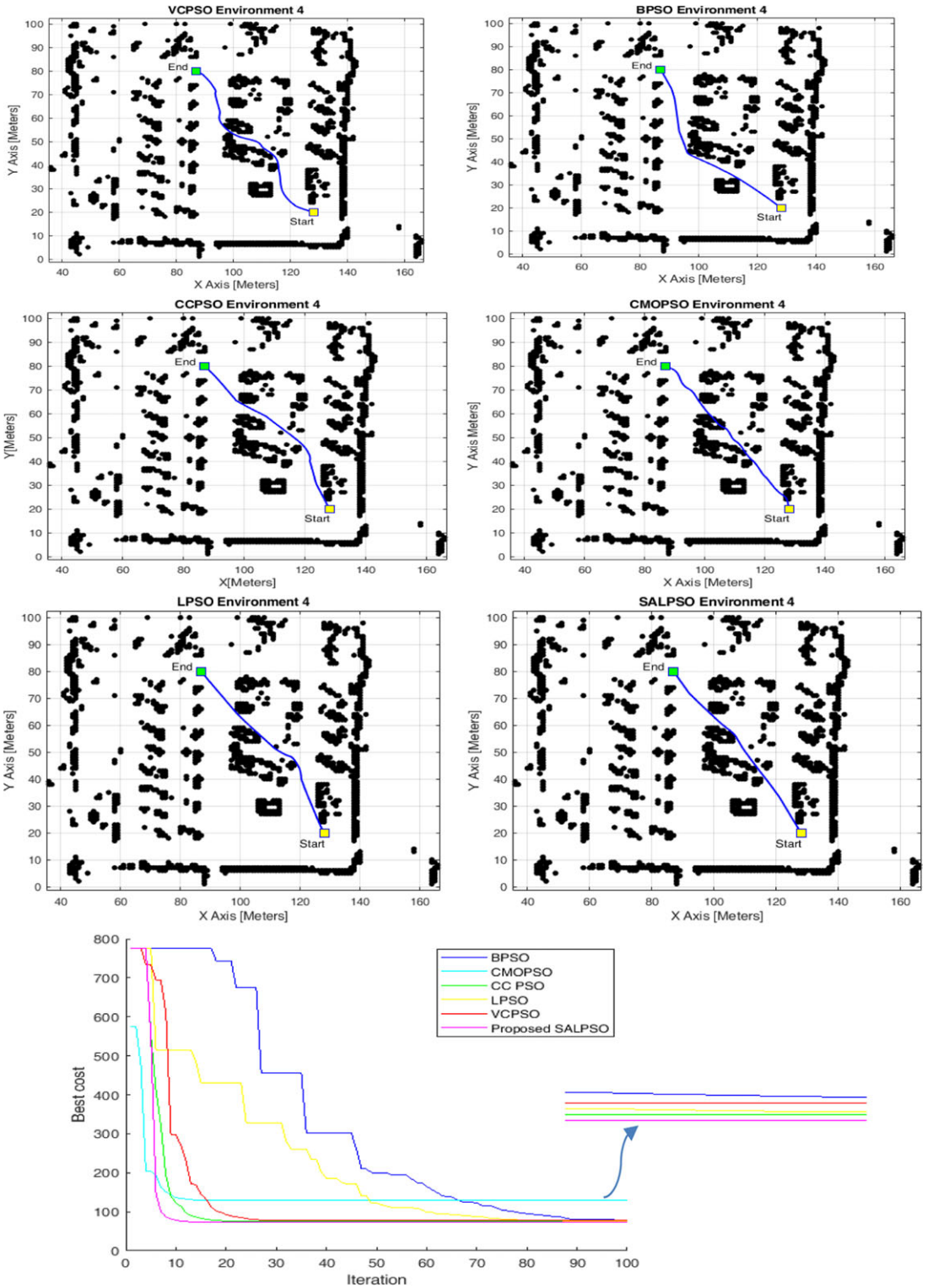

Figure 8. *Convergence graph and proposed heuristic algorithm path for suggested Lidar environment 3 path prediction.*

https://doi.org/10.1017/S0263574723001819 Published online by Cambridge University Press

https://doi.org/10.1017/S0263574723001819 Published online by Cambridge University Press

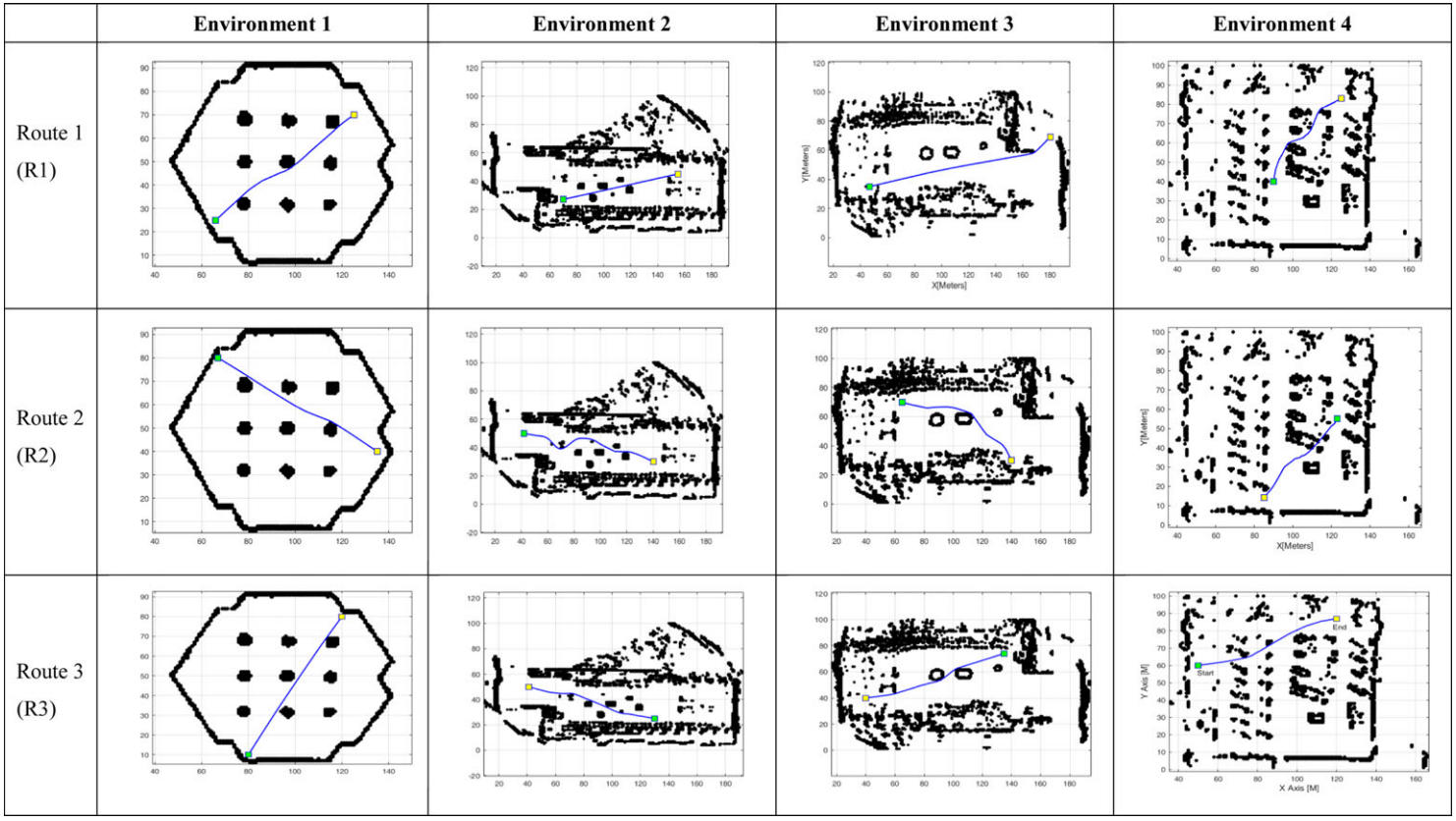

Figure 9. *Optimal path obtained from proposed SALPSO for all the environment with different routes – R1, R2, R3, and R4.*

respectively, when employing the suggested SALPSO. In route 3 the optimal path generated by the proposed algorithm is outperformed by the CCPSO algorithm with 0.3 m in difference. In environment 2 of the warehouse 2 (route 2) application, the shortest distance between loading items into the warehouse and unloading items at the destination is 80.5 m. Even though the shortest path without barriers only extends for a small distance, the ideal route is 80.7 m long when adopting the specified vehicle terrain route. Routes 3 and 4 arrive at their destinations at 106.4 and 93.6 m, respectively. Table IV optimization factor clearly shows that the SALPSO method generates more viable paths in all circumstances than the other five heuristic techniques.

In Table V, the proposed work is compared with CCPSO variant where the results are close to one another while compared to the other variants. Out of five different iteration results, the scenario 4 in environment 1 has a probability of less than 0.02 violation range in proposed algorithm than the CCPSO. The t-test values ranged from 4.57 to 7.84, suggesting statistical differences in the data between routes. Confidence intervals (CI) varied from 6.14 to 14.28, indicating the range within which the population parameter (e.g., mean path length) is likely to fall. The time required to traverse the routes varied between 6.3 and 8 min, with R1 being the quickest and R4 taking the longest. In environment 2, SD error values ranged from 0.68 to 3.26, representing the standard error of the standard deviation for each route. The time required to traverse the routes ranged from 9 to 17.2 min, with R3 being the quickest and R2 taking the most time. T-test values ranged from 5.81 to 9.47, demonstrating statistical differences in the data between routes. Similarly in environment 4, t-test values ranged from 6.47 to 13.07, suggesting statistical differences in the data between routes. The optimal path lengths ranged from 55.5 to 75 m, with R2 being the shortest and R1 the longest. This disparity is massive by any measure of statistical significance. The dissimilarity between the means of Groups Two and One. The calculations provided by this tool are rather elementary. The following constitutes the 95% confidence interval for this discrepancy: The calculated t-value was 43.274, with a 1.5 standard error of difference, using intermediate values between 78.01 and 83.75. Figure 10 R2 indicates a statistically significant result with a two-tailed p value of less than 0.0001. The statistical tests indicate that this difference is noteworthy. Around 38,014, including 41,466 as the lower bound of the 95% confidence interval for the gap between Groups 1 and 2. $t = 6.874$ and standard error difference $= 0.68$ are the determined intermediate values. In summary, the table provides a detailed summary of path planning and obstacle avoidance results across four different environments and multiple routes.

Xuexi Zhang et al.'s [36] benchmark instance is used to further validate ISALPSO's performance. The 2D lidar images acquired for the indoor rescue application are processed using the A* and DWA algorithm. The suggested approach uses an occupancy grid (200 × 100) to estimate indoor navigation feasibility. While the method under consideration was developed with logistics transit in mind, it makes use of cubic spline curves for smoothness in both cases. However, with a suitable grid configuration, the suggested SALPSO can be employed with both Lidar and satellite image-based localization methods. As shown in Fig. 10, the optimal performance is measured against a benchmark instance with the coordinates (20,40) and (130,40). Results show that the proposed ISALPSO algorithm is able to locate the optimal obstacle-free path in a short amount of time (the optimal distance for the ISALPSO route is 94.08 m, while the optimal distance for the A* and DWA algorithms is 107 m).

A realistic mapping, simulation, and feedback adjustment are all functions of the simulated model in the field of robotics. Following a series of simulation tests in the MATLAB simulation environment, the proper operational parameters are chosen and supplied to the empirical entity in order to obtain control of the virtual model over the practical robot [2]. Before the experiment starts, set consistent settings for the electric actuator's and the two-wheeled robot's movements. By defining the geographic coordinates of the target point, the physical robot can move in a directed manner because it uses an orthogonal encoder for global positioning [37]. This work presents the algorithm's verification and analysis on the intelligent Quanser Robot platform, which is built on ROS. The platform's drive system consisted of four in-wheel motors. Additionally, it featured a 2D-LiDAR (RPLIDAR A1M8) mounted horizontally at the front end of the robot along with a frame structure and a 3D-printed Lidar holder. This platform was outfitted with a laptop, inertial navigation, and other devices [36]. Using an Intel i5 10300H and

https://doi.org/10.1017/S0263574723001819 Published online by Cambridge University Press

Table V. *Statistical t-test comparison of proposed ISALPSO vs CCPSO for four environments with different routes.*

					Simulated Result Factors						
Environment	**Routes**	**Starting point**	**Destination point**	**Shortest path without obstacle**	**Violation (m)**	**Time (min)**	**Optimal path**	**Standard deviation**	**T-test value**	**CI**	**SD error**
Environment I	**R1**	(77,77)	(98,20)	**60.7**	nil	6.3	**61**	1.04	14.28	12	1.12
	R2	(125,70)	(66,25)	**74.2**	nil	7.2	**74.7**	1.85	6.14	18	1.44
	R3	(135,40)	(67,80)	**78.9**	0.02	7.8	**79.1***	2.01	7.84	13	1.97
	R4	(80,10)	(120,80)	**80.5**	nil	8	**80.8**	2.17	4.57	8	1.14
Environment II	**R1**	(45,85)	(155,25)	**128**	nil	13.7	**129.5**	1.57	5.22	26	1.51
	R2	(155,45)	(70,27)	**86.8**	nil	8.4	**87**	3.01	7.05	17	0.68
	R3	(140,30)	(42,50)	**100**	nil	11.7	**106.4**	2.22	12.44	35	3.26
	R4	(41,50)	(130,25)	**92.4**	0.01	9	**93.6**	2.82	14.51	39	1.84
Environment III	**R1**	(180,28)	(59,71)	**128.4**	nil	14	**128.8**	3.07	12.94	24	1.51
	R2	(180,69)	(47,35)	**137.2**	nil	17.2	**139.4**	1.24	5.81	15	0.68
	R3	(140,30)	(65,70)	**85**	nil	9	**91.8**	2.53	9.47	39	3.26
	R4	(40,40)	(135,74)	**100.9**	0.02	12	**102.5**	2.78	6.24	24	1.84
Environment IV	**R1**	(128,20)	(87,80)	**72.6**	nil	8.4	**73.8**	3.08	10.84	12	1.51
	R2	(125,83)	(90,40)	**55.5**	nil	6.5	**59.3**	1.75	6.47	29	0.68
	R3	(85,14)	(123,55)	**55.9**	nil	5.9	**56.5**	2.86	12.94	34	3.26
	R4	(120,87)	(50,60)	**75**	nil	7.8	**76.2**	2.06	13.07	41	1.84

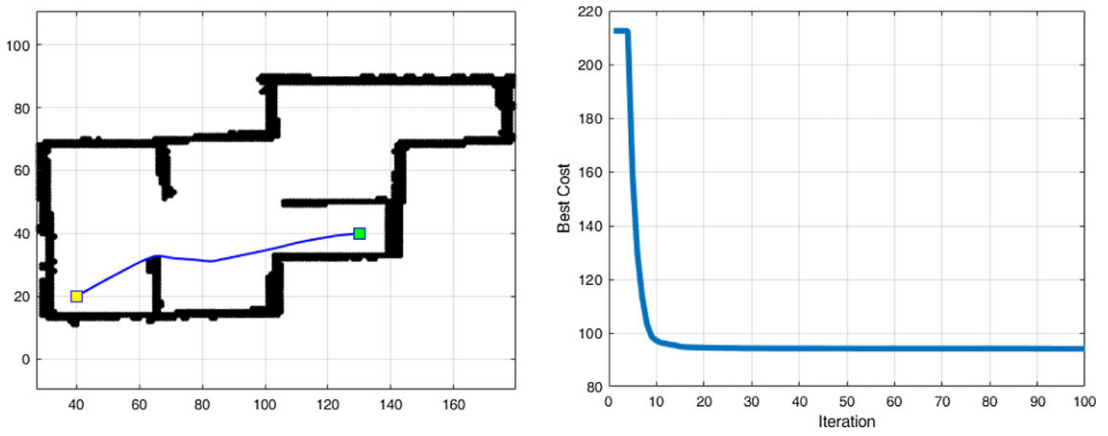

Figure 10. *Optimal path for Xuexi Zhang et al. (2020) benchmark instance using ISALPSO.*

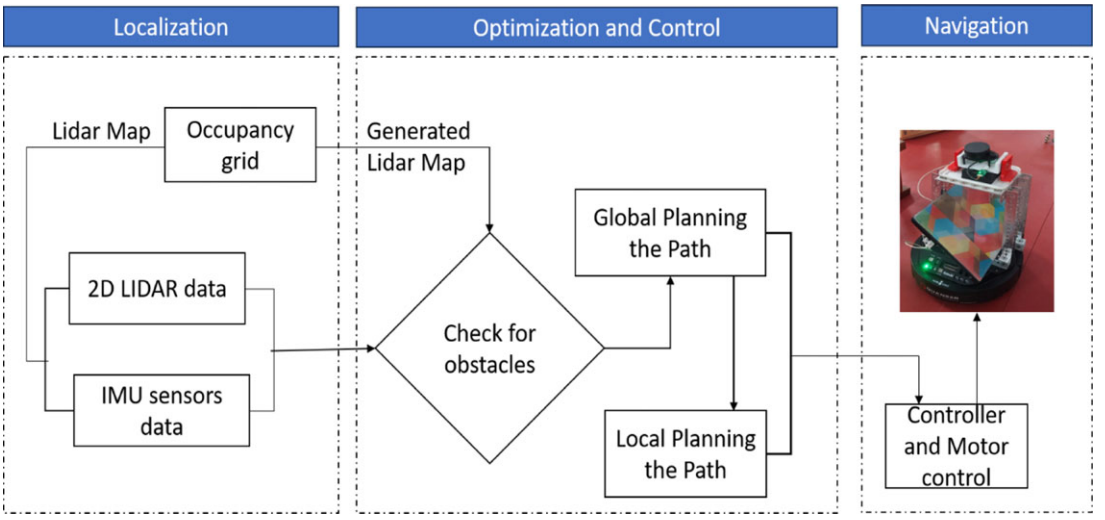

Figure 11. *Proposed SALPSO system framework diagram.*

16 GB of operating memory, the algorithm was performed in an Ubuntu 18.04 environment. Figure 11 displays the system framework diagram and the ROS intelligent vehicle platform.

Set up the physical robot control system with the same global coordinate system as environment 3, send the coordinates of the points in Fig. 9 from the starting node to the target point, and then let the robot move in accordance with the coordinates after receiving the path coordinates. Using the virtual outcome of the 18-point path presented in Fig. 11, Fig. 12 shows the robot's path prediction approach.

It is evident from the figure that the actual robot moves in accordance with the course learned via training in the simulated environment depicted in Fig. 9. We simulate a 2-wheeled mobile robot navigating in a crowded environment in order to test the proposed SALPSO-based path planning method. We evaluate the ISALPSO-based algorithm's performance in comparison to that of existing path planning algorithms, such as conventional PSO variants. The results show that the ISALPSO-based algorithm generates high-quality paths that optimize the travel time.

https://doi.org/10.1017/S0263574723001819 Published online by Cambridge University Press

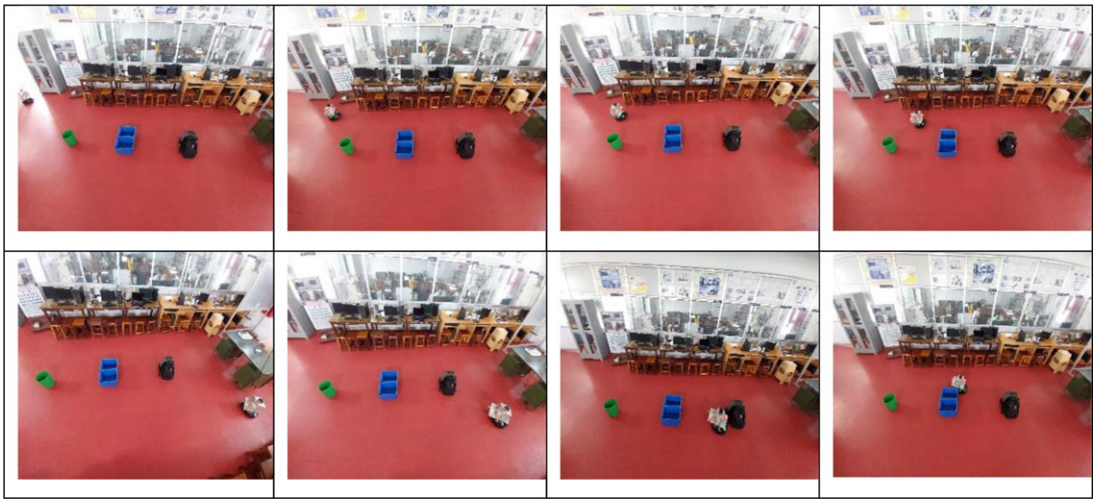

Figure 12. *Quanser robot validation of environment 3 using proposed SALPSO.*

6. Conclusion

In this paper, we proposed a SALPSO-based path planning algorithm for a 2-wheeled mobile robot that incorporates self-adaptation and obstacle avoidance. The proposed algorithm generates high-quality paths that optimize the performance criteria and avoid obstacles in the environment. The proposed SALPSO algorithm's self-adaptation mechanism improves the algorithm's performance and robustness to changing environments. The proposed algorithm can be applied to various mobile robots in unknown environments. This work proposes the path optimization of a mobile robot in four different lidar environments in an unknown environment. To find the correct path devoid of obstacles, the six heuristic PSO algorithms VCPSO, CCPSO, BPSO, CLPSO, LPSO, and proposed SALPSO were examined. Hector SLAM-based 2D lidar is initially used to create a 200 $\times$ 100 binary occupancy grid before mapping the selected locations. Using smooth cubic spline curve geometry, the localized starting and finishing locations are then used to create grid images. The findings also reveal a strong relationship between travel time, the intended distance trajectory, and the quantity of barriers ultimately encountered as a violation factor. The simulation results show that the CLPSO, BPSO, VCPSO, and LPSO algorithms require more time to estimate the best route and have longer paths. Even using the CCPSO method, the path length is optimally shortened, and the convergence rate is significantly reduced in few cases. However, the efficacy was confirmed by statistical analysis in terms of path length, violations, and computational time, the proposed SALPSO algorithm performs better than the CCPSO and other variant algorithms. The development of a digital representation of the mobile robot, the completion of communication between the simulation model and the physical entity, and the realization of the mobile robot's trajectory optimization are validated and the proposed SALPSO is viable for logistic autonomous mobile robot applications.

Data availability statement. Data could be given upon request, if needed.

Acknowledgments. The authors would like to express their gratitude to Centre for Robotics, Department of Mechatronics Engineering, Thiagarajar College of Engineering and Vellore Institute of Technology for their assistance in the creation of this publication.

Author contributions. Conceptualization, Methodologies, Draft Writing – Julius Fusic S and Sitharthan R. Simulation experimentation work and result discussion, editing, and project administration – Julius Fusic S.

Competing interests. The authors declare no competing interests.

https://doi.org/10.1017/S0263574723001819 Published online by Cambridge University Press

References

[1] P. V. Pande, D. K. Parhi and S. K. Pratihar, "Path planning for mobile robots using genetic algorithms," *J. Intell. Robot. Syst.* **54**(2), 223–245 (2009).

[2] A. Li, J. Cao, S. Li, Z. Huang, J. Wang and G. Liu, "Map construction and path planning method for a mobile robot based on multi-sensor information fusion," *Appl. Sci.* **12**(6), 2913 (2022).

[3] L. E. Parker, "A survey of path planning algorithms for mobile robotics," *IEEE Trans. Robot. Autom.* **10**(5), 505–526 (1994).

[4] S. Elnabarawy, M. Elhafsi and A. Elhawary, "Survey of SLAM algorithms for autonomous mobile robots," *Robotics* **9**(4), 96 (2020).

[5] M. H. Niaz, J. H. Lee and K. J. Kim, "A comprehensive review of SLAM algorithms and their applications in mobile robotics," *J. Intell. Robot. Syst.* **94**(1-2), 1–26 (2019).

[6] S. R. Goudar and M. M. Deshpande, "A comparative study of evolutionary algorithms for path planning of mobile robots," *Int. J. Adv. Res. Comput. Eng. Technol.* **1**(4), 147–153 (2012).

[7] H. Peng, Y. Li and X. Zhang, "Robot path planning using artificial potential field method," *J. Comput. Inf. Syst.* **6**(7), 2358–2365 (2010).

[8] A. Wahab, M. Nadhir, C. M. Lee, M. F. Akbar and F. H. Hassan. "Path planning for mobile robot navigation in unknown indoor environments using hybrid PSOFS algorithm." *IEEE Access* **8**, 161805–161815 (2020).

[9] A. Z. Arifin, H. Matsui and S. Sugimoto, "Path Planning for Mobile Robots Using Ant Colony Optimization and Fuzzy Logic," **In:** *Proceedings of the 6th International Conference on Intelligent Systems Design and Applications* (2006) pp. 433–438.

[10] V. Sathiya and M. Chinnadurai, "Evolutionary algorithms-based multi-objective optimal mobile robot trajectory planning," *Robotica* **37**(8), 1363–1382 (2019).

[11] A. Sharma and S. P. Singh, "Heuristic-based navigation of mobile robots using hybrid fuzzy logic and artificial potential field," *Robot. Auton. Syst.* **125**, 103409 (2020).

[12] J. Hao, X. He and Y. Sun, "A new hybrid heuristic method based on improved artificial potential field algorithm for mobile robot path planning," *IEEE Access* **9**, 23863–23873 (2021).

[13] J. Wang, J. Zhao and J. Zeng, "An adaptive artificial potential field algorithm for mobile robot path planning," *IEEE Access* **8**, 156601–156612 (2020).

[14] Y. Gao, Y. Xu, Q. Zhang and X. Huang, "A path planning method for mobile robots based on improved ant colony optimization algorithm," *Math. Probl. Eng.* (2020).

[15] X. Wang and Z. Wu, "A novel path planning method for mobile robots based on A* algorithm and artificial potential field," *J. Ambient Intell. Humaniz. Comput.* **12**(10), 11415–11425 (2021).

[16] S. Julius Fusic and R. Sitharthan, "Improved RRT* algorithm-based path planning for unmanned aerial vehicle in a 3D metropolitan environment," *Unmanned Syst.*, 1–17 (2023). https://www.worldscientific.com/doi/10.1142/S2301385024500225.

[17] Y. Liu and G. Zhang, "Mobile robot path planning based on an improved potential field algorithm," *Front. Robot. AI* **8**, 763408 (2021).

[18] R. Hussain, A. Ahmad and W. A. Khan, "Hybrid bat-inspired algorithm and ant colony optimization-based path planning for mobile robots," *SN Comput. Sci.* **2**(5), 1–15 (2021).

[19] Y. Wu, J. Wu and H. Wei, "Path planning for mobile robot based on improved artificial potential field algorithm," *J. Intell. Fuzzy Syst.* **39**(4), 6037–6049 (2020).

[20] J. Ren, C. Liu and Y. Zhang, "Path planning of mobile robot based on a novel artificial potential field method," *Int. J. Adv. Robot. Syst.* **17**(1), 1729881420907166 (2020).

[21] W. Al-Atabany, Y. Gao and H. Liu, "A hybrid heuristic algorithm for mobile robot path planning in dynamic environment," *J. Intell. Fuzzy Syst.* **40**(1), 1083–1093 (2021).

[22] E. Krell, A. Sheta, A. P. R. Balasubramanian and S. A. King, "Collision-free autonomous robot navigation in unknown environments utilizing PSO for path planning," *J. Artif. Intell. Soft Comput. Res.* **9**(4), 267–282 (2019).

[23] Y. Wang, B. Li, T. Weise, J. Wang, B. Yuan and Q. Tian, "Self-adaptive learning based particle swarm optimization," *Inf. Sci.* **181**(20), 4515–4538 (2011).

[24] J. Zhou and B. Duan, "A comprehensive review of simultaneous localization and mapping for mobile robots," *IEEE Access* **9**, 52970–52988 (2021).

[25] Y. Liu, X. Sun and M. Lu, "Chaotic cooperative particle swarm optimization for mobile robot path planning," *Robot. Auton. Syst.* **86**, 1–12 (2016).

[26] Y. Hu, Y. Wen and H. Zhang, "A comprehensive learning particle swarm optimization for path planning of mobile robot," *J. Intell. Robot. Syst.* **92**(3-4), 517–530 (2018).

[27] J. Jiang, Z. Wang and Y. Zhang, "An improved LSSPSO algorithm for mobile robot path planning," *J. Intell. Fuzzy Syst.* **41**(1), 551–562 (2021).

[28] Z. Liu and X. Zhang, "Mobile robot path planning based on improved PSO algorithm with local search strategy," *Int. J. Robot. Autom.* **34**(3), 240–247 (2019).

[29] W. Zhang, H. Zhang and Z. Xiong, "A modified velocity particle swarm optimization algorithm for mobile robot path planning," *Int. J. Adv. Robot. Syst.* **16**(3), 1–13 (2019).

[30] S. J. Fusic, R. Sitharthan, S. A. R. S. Masthan and K. Hariharan, "Autonomous vehicle path planning for smart logistics mobile applications based on modified heuristic algorithm," *Meas. Sci. Technol.* **34**(3), 034004 (2022).

[31] A. G. Gad, "Particle swarm optimization algorithm and its applications: a systematic review," *Arch. Comput. Methods Eng.* **29**(5), 2531–2561 (2022).

https://doi.org/10.1017/S0263574723001819 Published online by Cambridge University Press

[32] Z. Zhao and L. Liu, "An improved CLPSO algorithm for mobile robot path planning," *J. Intell. Fuzzy Syst.* **41**(1), 595–605 (2021).
[33] N. Srijaroon, K. Sethanan, T. Jamrus and C.-F. Chien, "Green vehicle routing problem with mixed and simultaneous pickup and delivery, time windows and road types using self-adaptive learning particle swarm optimization," *Eng. Appl. Sci. Res.* **48**(5), 657–669 (2021).
[34] X. Tao, X. Li, W. Chen, T. Liang, Y. Li, J. Guo and L. Qi, "Self-adaptive two roles hybrid learning strategies-based particle swarm optimization," *Inf. Sci.* **578**, 457–481 (2021).
[35] Q. Zhang, X. Ning, Y. Li, L. Pan, R. Gao and L. Zhang, "Path planning of patrol robot based on modified grey wolf optimizer," *Robotica* **41**(7), 1947–1975 (2023).
[36] X. Zhang, J. Lai, D. Xu, H. Li and M. Fu, "2D Lidar-based SLAM and path planning for indoor rescue using mobile robots," *J. Adv. Transport.* **2020**, Article ID 8867937, 14pp (2020).
[37] F. Alkhawaja, M. A. Jaradat and L. Romdhane, "Low-cost depth/IMU intelligent sensor fusion for indoor robot navigation," *Robotica* **41**(6), 1689–1717 (2023).
[38] S. J. Fusic, G. Kanagaraj, K. Hariharan and S. Karthikeyan, "Optimal path planning of autonomous navigation in outdoor environment via heuristic technique," *Transp. Res. Interdiscip. Perspect.* **12**, 100473 (2021).
[39] L. Li, L. Li, C. Li and Y. Li, "A self-learning PSO algorithm for mobile robot path planning," *IEEE Access* **9**, 17918–17929 (2021).

Cite this article: Julius Fusic S. and Sitharthan R. (2024). "Self-adaptive learning particle swarm optimization-based path planning of mobile robot using 2D Lidar environment", Robotica **42**, 977–1000. https://doi.org/10.1017/S0263574723001819