EAI Endorsed Transactions
on AI and Robotics
Research Article EAI.EU

An Integrated Beetle Antennae Search–Enabled Navigation Framework for Omnidirectional AGV Mobile Robots in Unknown Environments

Ata Jahangir Moshayedi[1], Atanu Shuvam Roy[2], Utsab Karan[3], Ming jun ZHANG[4,5], David Bassir[4,6,*]

[1]School of Information Engineering, Jiangxi University of Science and Technology, Ganzhou, Jiangxi, China
[2]Department of Computer Science and Engineering, University of Liberal Arts Bangladesh (ULAB), Dhaka, Bangladesh
[3]Department of EECE, Indian Institute of Technology Kharagpur, Kharagpur, India
[4]Smart Structural Health Monitoring and Control Laboratory, DGUT-CNAM, Dongguan University of Technology, D1, Daxue Rd., Songshan Lake, Dongguan, 523000, Guangdong, China
[5]CEDRIC/Laetitia, CNAM, Paris, France
[6]ENS -Paris-Saclay University, Centre Borelli, UMR CNRS 9010, 91190 Gif-sur-Yvette, France

Abstract

The Beetle Antennae Search algorithm is a relatively one of the recent optimization approach inspired by the foraging behavior of long-horn beetles, that copy how they use their antennae to explore the environment. This algorithm due to its simple structure and derivative-free nature, is well suited for robotic applications with limited computational resources. Despite many BAS-based navigation studies still handle path planning and control independently, or apply BAS only as an offline tr ajectory op timization to ol. As a re sult, less attention has been given to implement it in the real-time navigation framework. On the other hand, Mobile robots, in an unseen environment cannot work these problems separately rather they must continuously detect obstacles, update their environment model, keep re-planning safe routes, adjust control gains, and follow a predetermined reference trajectory. An integrated BAS-enabled navigation combining Simultaneous Localization and Mapping framework for an omnidirectional mobile robot in CoppeliaSim is presented in this research. A LiDAR-based occupancy grid is updated continually during motion, and obstacle detection via SLAM is used to improve safety. A modified A* algorithm is used to create collision-free avoidance pathways, which are then smoothed. At the same time, a BAS-driven adaptive PID controller is used to adjust the gains by using trajectory deviation as a feedback signal. To enable precise path following, a nine-sensor infrared array is used for line tracking. The integrated system can follow smoothed avoidance paths, calculate safe bypasses, identify impediments, and return to the original path. In the experiments, the system was tested in two scenarios - no obstacles and with three levels of obstacles. The results indicated higher performance when SLAM and BAS used together than conventional PID based navigation while software simulation of SLAM often hurting the performance as well.

Received on 26 December 2025; accepted on 01 February 2026; published on 11 February 2026

Keywords: Beetle Antennae Search (BAS), Mobile Robot Navigation, OMNI AGV, Obstacle Avoidance, Adaptive PID Control, Occupancy Grid Mapping, V-REP (CoppeliaSim)

doi:10.4108/airo.11438

* Corresponding author. Email: drajm@qq.com

1. Introduction

Autonomous mobile robots that are deployed in industrial, logistics, and service environments has to navigate safely and efficiently through obstacle filled areas while tracking their desired path. In practice, this requires the robot to continuously sense its surroundings for the said obstacles and maintain a map of free and occupied regions. The robot also must detect previously unknown obstacles, and react to them without abandoning their original task. The controller associated with multiple sensors and planning mechanisms must remain robust to errors, sensing noise, and sudden changes in the reference path induced by the frequent recalculation of paths. [1]. These requirements become more challenging when strong constraints are placed on computational budget and real-time responsiveness [2][3].

Robot control methods can be broadly categorized into several classes. Classical control approaches include proportional–integral–derivative (PID) control [4] and state-space control techniques [5]. Model-based methods, such as Model Predictive Control (MPC) [6] and sliding mode control [7], explicitly exploit system dynamics to achieve robust and optimal performance. Intelligent control approaches, including fuzzy logic [8] and neural network–based control [9], are designed to handle nonlinearities and uncertainties in complex robotic systems. Optimization-based algorithms are widely used for controller tuning and trajectory optimization, including Genetic Algorithms (GA) [10] [11], Particle Swarm Optimization (PSO) [12], and enhanced variants of PSO. For example, Song in 2021 [13] formulated automated guided vehicle (AGV) path planning as a multi-objective optimization problem and proposed an improved PSO-based method combined with an inductive steering strategy. Simulation results demonstrated superior performance in generating shorter and smoother paths while effectively avoiding obstacles in both static and dynamic environments. Liao et al. [14] in 2020 proposed SARSA- and DQN-based reinforcement learning approaches for AGV path planning, addressing instability in complex state spaces by introducing a DQN with a potential-field-based reward. Simulation and hardware experiments on a KUKA AGV showed that the DQN achieved an 88.62% success rate in complex environments, outperforming SA-SARSA, which achieved 87.82% in simpler scenarios.Between the various attempt, Beetle Antennae Search (BAS) [15] has attracted increasing attention due to its simplicity, fast convergence, and low computational cost.

This algorithm known as a derivative-free nature-inspired optimization algorithm that mimics the foraging behavior of beetles and is mainly applicable to real-time robotic applications and can be used to tune various controller parameters,such as PID [16]for more accurate trajectory tracking over obstacle avoidance [12].A review of prior research on the implementation of BAS in robotic systems indicates that this approach has demonstrated promising performance in controller tuning, path planning, and real-time navigation tasks.

A.T. Khan et al. [17] in 2020 introduced a new model-free control algorithm for home automation. They unified the problems of obstacle avoidance and trajectory tracking of redundant-manipulators into a single constrained optimization problem to be solved via a ZNN with BAS. They introduced a penalty function to reward the optimizer for obstacle avoidance while tracking the reference. Experiments were conducted using the 7-DOF KUKA LBR IIWA-14 model provided by MATLAB in a 3D environment with obstacles where they used to GJK Algorithm to determine distances between 3D objects. The results showed that the system converges very fast towards the reference trajectory and successfully avoids objects while doing so.

In 2021 Jianning et al. [18] integrated the BAS into the RRT* algorithm for efficient path planning in complex and tight spaces like the GIS maintenance scenarios. BASL-RRT* uses BAS in addition to random point sampling in the RRT* algorithm and then uses a weighted lazy extension method to discard BAS nodes[19]. They modelled the GIS cavity surface as a 2D plane with circles indicating holes and foreign obstacles. Experiments in this environment showed massive reductions in computation time and number of nodes created compared to RRT*. The authors report that RRT* required 133.15 s with 8,487 nodes, while BASL-RRT* reduced the runtime to 29.53 s using only 1,302 nodes.

In 2022, Qian Q. et al. [20] in their paper demonstrated the Enhanced BAS (EBAS) algorithm, an upgraded version of the classical BAS method tailored for complex and high-dimensional optimization tasks. EBAS incorporates a curve-based adaptive step size mechanism to better balance global exploration with local convergence, preventing premature stagnation. The algorithm was tested using the CEC'17 benchmark suite. In one notable case - the 10D F1 function - EBAS reduced the fitness value dramatically to $3.0\times10e6$, a clear leap over the original BAS's $1.1\times10e7$. Overall, EBAS consistently outperformed state-of-the-art algorithms such as Grey Wolf Optimizer (GWO) and Slime Mould Algorithm (SMA), establishing itself as a strong contender for tackling unbiased and complex optimization problems.

Liang, et al. [21] introduced an improved BAS path planning algorithm for vehicles in 2022. It uses map safety thresholds around obstacles, proposes the use of virtual target points and path-smoothening using B-Spline. The use of virtual target points improves the ability of the bot to fall into local extrema. The experiments were carried out of simulated 2D map with static obstacles on MATLAB. Compared to APF algorithm, VBAS generated shorter paths in much faster computation time in both single and multi-obstacle environments. The authors report that for multiple regular obstacles, VBAS achieved a path length of 860.56 m in 0.198 s compared to APF's 958.11 m in 2.326 s, while for multiple irregular obstacles VBAS produced a 1123.15 m path in 0.205 s versus APF's 1272.00 m in 1.220 s.In the paper by Zha et al. [22] in 2023, they have introduced a method to solve the strong randomness and poor stability of the BAS path planning algorithm for Unmanned Surface Vehicles. They put a constraint on the maximum turning angle and incorporated this into the fitness function. The also improved the step-size selection to reduce falls to the local minima. They simulated their approach on a 2D MATLAB environment with circular obstacles. The algorithm saw a 50% reduction in cumulative angle changes and also produced a shorter path length, reduced by almost 17 units. Yu, et al. [23] proposed combining water flow potential method with BAS for better path planning. They started by dividing paths into segments. They used the water flow potential field method for local path planning between waypoints, avoiding obstacles and iteratively optimise waypoints with BAS for shortest, collision-free path. Experiments were done is a custom 2D environment with static obstacles showed that the combined algorithm generally reduced optimal path length and showed faster convergence when compared with other heuristic algorithms. The authors report that WPFBAS achieved the shortest optimal path length of 28.62 with the lowest average runtime of 0.24 s, outperforming PSO at 29.45 and 3.68 s and GA at 31.57 and 0.42 s.Also authors Lyu et al. [24] proposed an Improved Beetle Swarm Optimization (IBSO) algorithm for efficient, collision-free path planning in static environments. Inspired by the Artificial Bee Colony algorithm, IBSO introduces multiple beetle agents for diversified global search and a Levy Flight step-size strategy to avoid local minima. Experiments on a 2D MATLAB map with circular obstacles demonstrate that IBSO achieves the shortest path length (542.01 m) and lowest runtime (1.31 s), outperforming ABC (549.68 m, 2.20 s) and PSO (553.03 m, 3.15 s).

In 2024 Chen C. et al. [25] in their paper presented a survey examining the convergence behavior and real-world applications of the BAS algorithm. While BAS is known for its simplicity and low computational overhead, the study identifies a key limitation that it has relatively slow convergence in high-dimensional optimization tasks. The authors investigated the significant influence of parameters tuning on the convergence behavior, to the end that, provided the appropriate values are chosen, BAS can certainly be used to arrive at optimal solutions. It is also observed in the study that hybrid methods especially such as BAS-PSO are methods that integrate the directional search nature of BAS with global learning nature of Particle Swarm Optimization. Such hybrid methods were said to work better in cases like path planning of robots and scheduling of resources. In particular, with electric load dispatch problems, BAS-PSO performed better when compared to the standalone PSO and Genetic Algorithms, which implies that the basis of hybrid approaches of BAS-based will perform well on complex and real-time optimization problems.

In 2025 Dwivedi et al. [26] studied a comparative analysis between BAS and Particle Swarm Optimization (PSO) for optimizing image transformation parameters relevant to navigation. Using BAS and PSO algorithms, the study fine-tuned BAS parameters for optimal performance in tasks such as 2D translation, rotation, and scaling. Error metrics based on pixel intensity were used to assess performance. BAS outperformed PSO, achieving 12.5% faster convergence and an 8.3% lower final alignment error. The study concluded that BAS effectively avoids local minima and is computationally efficient for image-based optimization tasks relevant to path planning. In the same year, Jiang et al. [27] designed an improved FOPID controller (FBPA-FOPID) for efficient and accurate path tracking on a 6 DOF robotic arm. They used a Fractional Order PID controller to make it more adaptive to complex systems and proposed using BAS along with Particle Swarm Optimization to tune the controller. Experiments were done on the UR5 robotic arm in Simulink/MATLAB environment. Results showed that their controller had the smallest overshoot, fastest response and lowest tracking error when compared to PID and FOPID controllers.

Although numerous studies have been reported on robotic path tracking, most existing works lack experimental validation on omni-directional robot platforms. Omni-directional robots are particularly important because they enable independent control of motion in longitudinal, lateral, and rotational directions without requiring platform reorientation. This holonomic mobility significantly enhances maneuverability in confined and cluttered environments such as warehouses, hospitals, and industrial facilities, while also enabling smoother trajectory tracking and faster responses to dynamic obstacles. Consequently, omni-directional platforms are well suited for advanced autonomous navigation and high-precision AGV applications. Prior work shows that BAS-based and hybrid optimization methods significantly improve convergence speed and path efficiency in robotic and vehicle navigation, though their application to high-dimensional and omni-directional platforms remains relatively underexplored. In contrast, a mobile robot deployed in realistic scenarios must coordinate multiple tasks simultaneously. It must build or update an environment map from onboard sensors, detect collisions or near-collisions in real time [28], compute avoidance trajectories, follow them with sufficient smoothness, and then rejoin its nominal mission path. The control gains that work well during nominal line following may be sub-optimal when the robot is executing sharp avoidance maneuvers in the vicinity of obstacles. This creates a coupled problem: path planning and controller tuning

cannot be considered in isolation if the goal is end-to-end robust navigation [29].To address this challenge, an integrated navigation framework is required in which perception, replanning, and control adaptation operate within a unified online loop. In this type of framework, the planner is required to consider the dynamic response of the robot, while the controller adjusts its gains based on how much the trajectory deviates and the surrounding environment. This close coupling helps the robot to keep stability, smooth movement, and good tracking accuracy, even when operating in cluttered and partly unknown environments.
Although Moshayedi et al. [30] evaluated an omnidirectional robot in CoppeliaSim, focusing on navigation accuracy, efficiency, and safety across diverse path conditions using SLAM. As the metrics time, velocity, body orientation, and tracking error were analyzed, and results shows that higher speeds increase deviation and require improved speed control. These findings support optimizing AGV performance in real-world applications including automation, healthcare, and service robotics while reducing development cost and risk through virtual prototyping and Niu et al. in 2024 [10] addressed AGV path planning using mechanum-wheeled omnidirectional platforms by improving the traditional Genetic Algorithm. They employed Ant Colony Optimization-based initialization for higher-quality initial paths and incorporated path smoothness constraints, along with advanced selection, crossover, and mutation strategies, to reduce redundant nodes, avoid local minima [31], and accelerate convergence. Experiments using a ROS-based software stack show that the improved algorithm reduces path length by almost 30%, convergence count by 50%, and turn counts by over 75%.There are relatively few studies that have implemented control and navigation strategies on AGV platforms with omni-directional wheels ; however, these platforms are expected to play a crucial role in the future of robotics due to their superior maneuverability, holonomic motion capability, precise positioning, and ability to operate efficiently in constrained and dynamic environments [32]
And as a lightweight single-agent metaheuristic with minimal parameterization, BAS offers competitive performance across a wide range of optimization problems while maintaining strong real-time capability. Its derivative-free structure and online adaptability make it especially suitable for automated guided vehicle (AGV) control, including dynamic obstacle avoidance and controller tuning. Recent studies indicate that BAS can reduce path length, improve convergence speed, and lower computational complexity compared with more computationally intensive optimization methods, although many existing works apply BAS primarily as an offline planner or gain tuner[12].Integrating BAS directly into the online navigation and control loop enables more robust, adaptive, and efficient robot operation in dynamic and partially unknown environments. In parallel, learning-based approaches, particularly deep reinforcement learning, have enabled robots to exhibit adaptive and autonomous behaviors in complex and dynamic environments. Hybrid control strategies that combine classical, optimization-based, and learning-based methods are increasingly adopted to balance stability, adaptability, and computational efficiency.Considering the objective of demonstrating the efficiency and potential of the BAS algorithm, as well as its effectiveness when combined with obstacle detection techniques on an omni-directional robotic platform, the objectives of this paper are as follows:

1. To design and implement an integrated navigation and control framework for an omni-directional mobile robot operating in environments with unknown static obstacles.

2. To demonstrate the effectiveness of the BAS algorithm for real-time adaptive tuning of PID controller gains during path tracking.

3. To employ the BAS algorithm as a path-smoothing and trajectory refinement method to transform discrete A*-based paths into dynamically feasible motions, in conjunction with SLAM-based mapping.

4. To test the online obstacle detection and local map updating using onboard sensors without prior knowledge of the environment.

5. To test and ensure smooth obstacle avoidance and reliable rejoining of the original reference path with bounded deviation.

6. To evaluate and study the proposed framework using measurable performance metrics such as path deviation, total distance traveled, execution time, and changes in angular motion.

7. To check the practical feasibility of integrating BAS into both the planning and control layers when implemented on an omni-directional robotic platform.

The using of Simultaneous Localization and Mapping (SLAM), BAS, and the A* search algorithm give a good opportunity for autonomous robotic navigation and global optimization. SLAM serves as the primary mechanism for real-time environment mapping and state estimation, while BAS offers a computationally efficient, gradient-free metaheuristic for searching optimal solutions based on the biological sensing mechanics of beetle antennae. When coupled with the A* algorithm the standard for deterministic pathfinding and cost-minimal trajectory planning.

These technologies enable a multilayered approach to navigation, balancing local obstacle avoidance, global path optimization, and efficient search in complex multidimensional spaces. The remainder of this paper is organized as follows. Following the introduction and statement of objectives, Section 2 describes the proposed methodology, including an overview of the BAS algorithm and detailed specifications of the environment, robot platform, and sensor hardware and software. Section 3 reports the simulation and testing results, accompanied by a detailed analysis and discussion. Finally, Section 4 concludes the paper by summarizing the findings and outlining the limitations of the study and directions for future work.

2. Methodology

As mentioned , the main algorithms employed in this work are the BAS algorithm and a modified A* search method along with SLAM , which are described in the following sections.SLAM and A* search are chosen in this work because they play complementary roles in autonomous navigation. SLAM allows the robot to build a map of an unknown environment while at the same time estimating its own position using onboard sensors, which is important for stable and reliable operation in dynamic environments. Accurate localization and mapping support effective obstacle avoidance and informed decision-making [30]. A* search provides an efficient and optimal path-planning mechanism by computing collision-free paths between start and goal states, and when combined with SLAM-generated maps, it ensures safe and optimal navigation [32]

2.1. Beetle Antenna Search (BAS)

BAS is inspired from the exploratory behavior of beetles when sensing and moving toward food sources. The algorithm works by updating positions in an iterative manner using local comparisons of objective function values, which allows the search space to be explored and exploited efficiently with relatively low computational cost. [25] Let the solution space be defined in $\mathbb{R}^d$, where each beetle represents a candidate solution vector $\mathbf{p}_k \in \mathbb{R}^d$, for $k = 1, 2, \ldots, M$. At initialization, all beetles are randomly distributed within the feasible domain. The quality of a candidate solution is evaluated using an objective function $\mathcal{J}(\mathbf{p})$, which is problem-dependent and maps a position vector to a scalar fitness value. A key component of BAS is the perception range, denoted by $\rho(t)$, which specifies the spatial neighborhood within which a beetle can sense alternative solutions at iteration t. This sensing range is initialized to a predefined maximum value ρ_0 and is progressively reduced to promote convergence.Meanwhile of each iteration, every beetle $\mathbf{p}_k(t)$ samples a neighboring candidate $\mathbf{p}_\ell(t)$ within its sensing radius. If the neighboring beetle exhibits superior fitness, i.e.,

$$\mathcal{J}(\mathbf{p}_\ell(t)) > \mathcal{J}(\mathbf{p}_k(t)),$$

the beetle updates its position by moving in the direction of the superior solution according to

$$\mathbf{p}_k(t+1) = \mathbf{p}_k(t) + \alpha \left(\mathbf{p}_\ell(t) - \mathbf{p}_k(t)\right), \tag{1}$$

where $\alpha > 0$ denotes the motion gain controlling the step magnitude.If no improvement is detected, the beetle performs exploratory motion within its local neighborhood to avoid premature stagnation. This random perturbation is modeled as

$$\mathbf{p}_k(t+1) = \mathbf{p}_k(t) + \boldsymbol{\varepsilon}(t), \tag{2}$$

where $\boldsymbol{\varepsilon}(t)$ is a bounded stochastic vector sampled uniformly within the sensing radius.Optionally, a refinement operator may be applied after the movement phase to further enhance solution quality. This operator encapsulates problem-specific heuristics and is expressed as

$$\mathbf{p}_k(t+1) = \Phi\left(\mathbf{p}_k(t+1)\right), \tag{3}$$

where $\Phi(\cdot)$ denotes a local improvement mapping.

To balance global exploration and local exploitation, the sensing radius is updated over time using a decay rule:

$$\rho(t+1) = \gamma\, \rho(t), \quad 0 < \gamma < 1, \tag{4}$$

ensuring a gradual transition from coarse search to fine optimization. If $\rho(t)$ falls below a predefined threshold $\rho_{\min}$, it may be reinitialized to ρ_0 to restore diversity.

The iterative process continues until a stopping condition is met, such as reaching a maximum number of iterations or observing negligible improvement in the objective function. Upon termination, the best solution obtained across all beetles is reported as

$$\mathbf{p}^* = \arg\max_{\mathbf{p}_k} \mathcal{J}(\mathbf{p}_k). \tag{5}$$

2.2. Modified A* Pathfinding

The A* algorithm is a widely-used path finding algorithm that finds the shortest path from a start node to a goal node in a weighted graph. The algorithm 1 as it shown, maintains a priority queue of nodes to explore, where each node n is assigned a cost function:

$$f(n) = g(n) + h(n) \tag{6}$$

where, $g(n)$ is the actual cost from the start node to node n; $h(n)$ is the heuristic estimate of the cost from node n to the goal; $f(n)$ is the estimated total cost of the path through node n.The heuristic function used in this

Algorithm 1 Beetle Antenna Search (BAS) Algorithm

```
 1: Randomly initialize beetle positions $\mathbf{p}_k(0) \in \mathbb{R}^d$, $k = 1, 2, \ldots, M$, within the feasible domain
 2: Set sensing radius $\rho(0) = \rho_0$
 3: Evaluate objective function $\mathcal{J}(\mathbf{p}_k(0))$ for all beetles
 4: for $t = 0$ to $T_{\max} - 1$ do
 5:   for each beetle $k = 1$ to $M$ do
 6:     Select a neighboring beetle $\mathbf{p}_\ell(t)$ such that $\|\mathbf{p}_\ell(t) - \mathbf{p}_k(t)\| \le \rho(t)$
 7:     if $\mathcal{J}(\mathbf{p}_\ell(t)) > \mathcal{J}(\mathbf{p}_k(t))$ then
 8:       Update position using directed motion:
 9:       $\mathbf{p}_k(t+1) = \mathbf{p}_k(t) + \alpha\big(\mathbf{p}_\ell(t) - \mathbf{p}_k(t)\big)$
10:     else
11:       Generate random perturbation $\varepsilon(t)$ with $\|\varepsilon(t)\| \le \rho(t)$
12:       $\mathbf{p}_k(t+1) = \mathbf{p}_k(t) + \varepsilon(t)$
13:     end if
14:     Apply optional refinement operator:
15:     $\mathbf{p}_k(t+1) = \Phi\big(\mathbf{p}_k(t+1)\big)$
16:     Evaluate fitness $\mathcal{J}(\mathbf{p}_k(t+1))$
17:   end for
18:   Update sensing radius: $\rho(t+1) = \gamma\,\rho(t)$
19:   if $\rho(t+1) < \rho_{\min}$ then
20:     $\rho(t+1) \leftarrow \rho_0$
21:   end if
22: end for
23: Determine best solution: $\mathbf{p}^* = \arg\max_{\mathbf{p}_k} \mathcal{J}(\mathbf{p}_k)$
24:
25: return $\mathbf{p}^*$ =0
```

implementation is the Euclidean distance:

$$h(a, b) = \sqrt{(a_x - b_x)^2 + (a_y - b_y)^2} = \|a - b\|_2 \tag{7}$$

To enhance safety and generate paths that maintain distance from obstacles, we introduce a modified A* algorithm that incorporates an obstacle penalty term into the cost function. The modified cost function becomes:

$$f'(n) = g(n) + h(n) + \lambda \cdot \phi(n) \tag{8}$$

where, λ is a weighting factor (set to 1 in this implementation); $\phi(n)$ is the obstacle penalty function. The obstacle penalty function $\phi(n)$ counts the number of occupied cells within a specified radius r around node n. For a grid $\mathcal{G}$ where $\mathcal{G}(x, y) = 1$ indicates an obstacle and $\mathcal{G}(x, y) = 0$ indicates free space, the penalty is defined as:

$$\phi(n) = \sum_{\substack{dx=-r \\ dy=-r \\ (dx,dy)\neq(0,0)}}^{r} \mathbb{I}_B(n_x + dx, n_y + dy) \cdot \mathcal{G}(n_x + dx, n_y + dy) \tag{9}$$

where, $n = (n_x, n_y)$ is the position of node n in the grid; $r = 4$ is the inspection radius; $\mathbb{I}_B(x, y)$ is an indicator function that equals 1 if (x, y) is within grid bounds, 0 otherwise The indicator function is defined as:

$$\mathbb{I}_B(x, y) = \begin{cases} 1 & \text{if } 0 \le x < \text{rows and } 0 \le y < \text{cols} \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

The algorithm uses a 4-directional movement model, allowing transitions only in cardinal directions:

$$\mathcal{N}(n) = \{(n_x + dx,\ n_y + dy) : (dx, dy) \in \{(-1, 0), (1, 0), (0, -1), (0, 1)\}\} \tag{11}$$

The transition cost between adjacent nodes is computed as:

$$c(n, n') = \sqrt{dx^2 + dy^2} \tag{12}$$

For the 4-directional model, this simplifies to $c(n, n') = 1$ for all valid transitions. The following 2 shows the modified A* search algorithm tuned with obstacle avoidance.

Algorithm 2 Modified A* with Obstacle Avoidance

```
 1: Input: Grid $\mathcal{G}$, Start $s$, Goal $g$, Radius $r$
 2: Output: Path from $s$ to $g$
 3: Initialize OpenSet $\leftarrow \{(f'(s), g(s), s, [s])\}$
 4: Initialize Visited $\leftarrow \emptyset$
 5: while OpenSet $\neq \emptyset$ do
 6:   $(f', g_{\text{cost}}, n, \text{path}) \leftarrow$ PopMin(OpenSet)
 7:   if $n = g$ then
 8:
 9:     return path
10:   end if
11:   if $n \in$ Visited then
12:     continue
13:   end if
14:   Visited $\leftarrow$ Visited $\cup \{n\}$
15:   for all $n' \in \mathcal{N}(n)$ where $\mathcal{G}(n') = 0$ and $n' \notin$ Visited do
16:     $g_{\text{new}} \leftarrow g_{\text{cost}} + c(n, n')$
17:     $\phi(n') \leftarrow$ CountObstaclesInRadius$(\mathcal{G}, n', r)$
18:     $f'(n') \leftarrow g_{\text{new}} + h(n', g) + \phi(n')$
19:     OpenSet $\leftarrow$ OpenSet $\cup \{(f'(n'), g_{\text{new}}, n', \text{path} + [n'])\}$
20:   end for
21: end while
22:
23: return $\emptyset$ {No path found} =0
```

In Algorithm 2, standard A* algorithm is extended by augmenting the evaluation function with an obstacle-density penalty, discouraging paths that pass close to cluttered regions. At each expansion, only collision-free neighboring nodes are considered, and their costs are updated using the accumulated path cost, heuristic distance to the goal, and a local obstacle term within a specified radius. The search terminates when the goal is reached with a feasible path or when no valid nodes remain to be explored. This work develops an integrated autonomous navigation framework for a mobile robot operating in cluttered and partially unknown environments. The proposed system adopts a closed-loop architecture that fuses local perception, online occupancy-grid mapping, global replanning, and adaptive control. The primary objective is to enable the robot to track a predefined reference path while robustly responding to unexpected obstacles, reconstructing the local environment, computing collision-free trajectories using a modified A* search algorithm, and smoothly rejoining the original route.

2.3. Integrated BAS–SLAM Navigation

The complete execution flow of the integrated framework is summarized in Algorithm 3.

BAS-PID Line Following. Trajectory tracking is performed using a PID controller whose gains (K_p, K_i, K_d) are adapted online through the BAS metaheuristic as shown in Algorithm 3. This strategy allows the controller to deal with nonlinear dynamics and changing tracking requirements that occur during obstacle avoidance. BAS imitates the sensing behavior of long-horned beetles by creating mirrored perturbed values around the current gain vector, which represent the left and right antennae. At every two control iterations, the deviation of the robot's trajectory is measured and used as the fitness value. Based on this, the PID gains are updated toward the direction that gives better tracking performance. An exponentially decreasing step size is used so that convergence is achieved while keeping the closed-loop system stable.

Obstacle Avoidance and Global Re-planning. The normal line-following behavior is stopped once an obstacle is detected by the forward LiDAR within a predefined safety distance. At this point, the robot is forced to halt and a global replanning process is started. First, a suitable rejoining point on the original global path is chosen outside the detected obstacle areas. This selection is mainly based on geometric closeness and basic reachability constraints.Next, path planning is carried out on the filled occupancy grid using a modified A* search. Here the cost function $f(n)$ is developed as a result of integrating the Euclidean path length, a heuristic parameter that is the distance that is left to be covered to reach the target and another penalty factor that is a local congestion. This hybrid action plan would direct the planner on safer and more dependable directions, especially on situations that are crowded with clutters. On obtaining the first

Algorithm 3 Integrated BAS–SLAM Navigation

```
Require: Initial pose $\mathbf{x}_0$, reference path $\mathcal{P}$, occupancy grid $G$, PID gains $(K_p, K_i, K_d)$, BAS parameters $(d_0, d_{inc}, d_{decay})$.
 1: Preprocessing: Fill closed regions in $G$; inflate obstacles by 3 cells.
 2: Initialize BAS: $d \leftarrow d_0$, $D \leftarrow d_{decay} \cdot d$, $\mathbf{b} \sim \mathcal{N}(0, I_3)$
 3: while simulation active do
 4:     Acquire LiDAR scan $\mathcal{L}$; acquire vision sensors; compute line error $e$.
 5:     if $\min(\text{dist}(\mathcal{L}_{\text{front}})) < \tau_{\text{obs}}$ and SLAM enabled then
 6:         Suspend robot motion: $\mathbf{v} \leftarrow \mathbf{0}$
 7:         Determine rejoin target $\mathbf{g}$ on $\mathcal{P}$ (forward-biased selection).
 8:         Modified A* Pathfinding:
 9:             Initialize priority queue: $f(\mathbf{x}_0) = h(\mathbf{x}_0, \mathbf{g})$
10:         while queue not empty do
11:                 Pop node $n$ with minimum $f(n)$
12:             if $n = \mathbf{g}$ then
                        break
13:             end if
14:             for all free neighbors $n'$ of $n$ do
15:                 Obstacle penalty: $\phi(n') = 0.1 w_{\text{obs}} \sum_{r \le 4} \mathbb{I}_{G(n'+r)=1}$
16:                 Priority: $f(n') = g(n') + h(n', \mathbf{g}) + \phi(n')$
17:             end for
18:         end while
19:         Extract path $\Gamma$; subsample to waypoints (step $\approx |\Gamma|/6$).
20:         Smooth: $\Gamma' \leftarrow$ BAS-Smooth($\Gamma; \alpha = 15, \beta = 5, \gamma = 0.01$, 150 iter)
21:         Waypoint Tracking:
22:         for all waypoint $\mathbf{w} \in \Gamma'$ do
23:             while $\|\mathbf{w} - \mathbf{x}_{\text{robot}}\| > \tau_{\text{arrival}}$ do
24:                 Transform $\mathbf{w}$ to robot frame; compute heading $\alpha$
25:                 Apply: $\mathbf{v} \leftarrow (0.8 v_{\text{target}}, 0, -8\alpha^3 - 16\alpha)$
26:             end while
27:         end for
28:         Align orientation to $\theta_{\mathbf{g}}$; reset integral $I \leftarrow 0$
29:     else
30:         BAS-PID Line Following:
31:         if iteration mod 2 = 1 then
32:             Sample direction: $\mathbf{b} \sim \mathcal{N}(0, I_3)$; normalize
33:             Left antenna: $\mathbf{K}_L = [K_p, K_i, K_d] + d \cdot \mathbf{b}$; measure $e_L$
34:         else
35:             Right antenna: $\mathbf{K}_R = [K_p, K_i, K_d] - d \cdot \mathbf{b}$; measure $e_R$
36:             Update: $[K_p, K_i, K_d] \leftarrow [K_p, K_i, K_d] + \text{sign}(e_R - e_L) \cdot D \cdot \mathbf{b}$
37:             Decay: $d \leftarrow d_{decay} \cdot d + d_{inc}$, $D \leftarrow d_{decay} \cdot D$
38:         end if
39:         Compute PID: $u = K_p e + K_i I + K_d(e - e_{\text{prev}})$
40:         Apply velocities: $\mathbf{v} \leftarrow (v_{\text{target}}, 0, u)$
41:     end if
42: end while=0
```

path likely generated by the AST algorithm, the path is down sampled and then refined to a solution obtained via a smoothing process that is inspired by the BAS algorithm. The step of refinement minimizes spikes in directional change and undesired oscillations and produces a smooth curve to provide a much improved fit to the kinematic and dynamic constraints of the robot.

Trajectory Execution and Path Rejoining:. The optimized avoidance trajectory is carried out by converting the planned waypoints into the robot's local coordinate frame and then generating the required linear and angular velocity commands. During this phase, the BAS-tuned PID controller stays active to continuously correct tracking errors. After the robot reaches the selected rejoin point, its yaw is aligned again with the original reference path, and normal line-following operation is smoothly resumed.All components of the proposed framework, including BAS-based gain adaptation, SLAM-based mapping, and A*-based replanning, operate within a fixed-rate simulation control loop. Due to the lightweight, single-agent formulation of BAS and the limited dimensionality of the PID gain space, the additional computational overhead introduced by online optimization remains low compared to mapping and global replanning. While the present study focuses on navigation performance rather than detailed timing analysis, the framework was executed without observable control-loop violations in simulation, supporting its suitability for real-time operation. Quantitative profiling of replanning latency, control-cycle timing, and CPU usage is left for future hardware-based evaluation.

2.4. Used platform Overview and Architecture

As mentioned earlier, although various robotic configurations have been explored in the literature, relatively limited work has been reported on the implementation and experimental validation of navigation and control frameworks on omni-directional AGV platforms. Addressing this gap is the primary focus of this paper. To this end, the KUKA YouBot-an omni-directional AGV equipped (Figure 1) with mecanum wheels-is selected as the experimental platform due to its flexibility, maneuverability, and relevance to real-world industrial applications. The detailed hardware specifications of the KUKA YouBot platform, as modeled in the simulation environment, are summarized in Table 1.

SLAM is then applied to project both the LiDAR endpoints and intermediate points into a global 2D occupancy grid. In simulation this is done via ray-casting. Cells crossed by a ray are marked as free space, the end points are labeled as occupied, and unknown areas are initialized first and gradually updated as the robot moves. The line-tracking subsystem is basing on a vision sensor array mounted at the front that constantly supplies data on the lateral offset that the robot has of the black reference line. The sensors give an averaged value of grayscale intensity and a symmetric weighted plan is employed amongst the sensors to compute a single deviation error. This error signal is subsequently given to the steering controller and it is the primary input in controlling the path-tracking behavior.

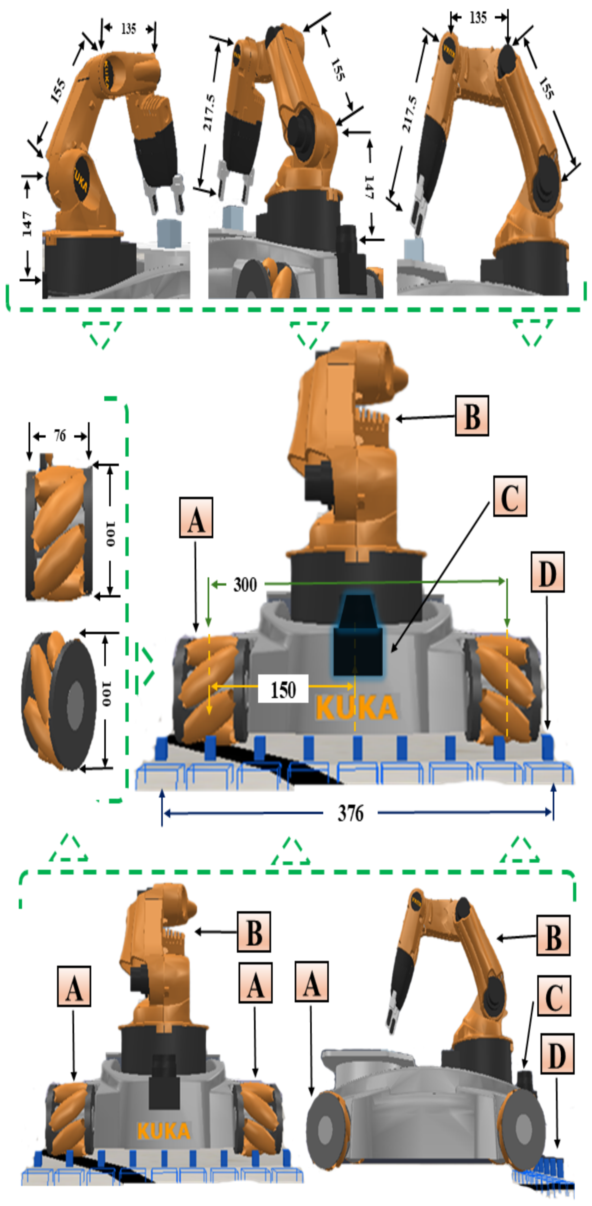

Figure 1. Components and sections of the omni-directional KUKA robot equipped with perception sensors: (A) omni-wheel, (B) robot arm, (C) Hokuyo URG 04LX (approx. 5.6 m range) LiDAR, and (D) nine-vision-sensor array configured as a line sensor.

Table 1. The KUKA YouBot Platform Specification

Parameter	Specification
Dimensions (L × W × H)	580 mm × 376 mm × 140 mm
Total Weight	20 kg
Drive System	4 Mecanum Wheels (Omnidirectional)
Max. Payload	20 kg
Manipulator	5 DoF (not utilized for navigation)
LiDAR Sensor	Hokuyo URG-04LX (Range: ≈ 5.6 m)
Line Sensors	Custom 9-Element IR Array
Communication	ZeroMQ Remote API (Simulated Wireless)
Omni-Wheel	2 rims and 9 free-running rollers (mounted @ 45 degree)

2.5. Tested and Simulation Environment

The simulation setup is implemented using the CoppeliaSim platform (formerly V-REP) [33], where a KUKA mobile manipulator is utilized, as depicted in Figure 1. To ensure experimental reliability, all simulations are carried out in a controlled indoor environment. The workspace features a rigid ground plane with dimensions of $10 \times 10 \times 0.40$ m along the x, y, and z axes, respectively. All the simulations are done in controlled indoor atmosphere to guarantee the reliability of the experiments.
The workstation has such a surface ground plane dimensions of $10 \times 10 \times 0.40$ m along the x, y, and z axis. One of its workspaces has a predetermined reference trajectory which is held fixed during all experiments. The trajectory has a total length of 32.3 m and a uniform width of 0.04 m making it impose a uniform geometrical assumption on the assessment of the path-following performance. There are two experimental conditions, which include the environment free of obstacles and one with obstacles. The sensing set up, the control settings and the reference trajectory, as well as the robot model, are maintained in both instances.

Scene A: Obstacle-Free Environment:. In the obstacle-free configuration, the workspace contains no external objects. This scene serves as a baseline to evaluate path-following section only and motion control performance without the sensing-driven interference.

Scene B: Obstacle Environment:. In the obstacle-based configuration, the environment is populated with static cubic obstacles, each with identical dimensions of $0.50 \times 0.50 \times 0.50$ m. The obstacles are placed at predefined locations to introduce spatial constraints and perception challenges. Different experimental runs activate subsets of these obstacles to realize environments with increasing obstacle density, specifically three, seven, and nine obstacles.All obstacle locations used across the experiments are summarized in Table 2. For all obstacles, the vertical coordinate is fixed at $z = 0.25$ m.

Table 2. Obstacle(OBS)locations for different obstacle-density scenes. all obstacle with the size of 0.50m x 0.50m x 0.50m.

Three-Obstacle(OBS) Scene			
OBS	x (m)	y (m)	z (m)
OB1	7.50	-1.35	0.25
OB2	-3.775	0.625	0.25
OB3	2.550	1.850	0.25
Seven-Obstacle(OBS) Scene			
Obstacle	x (m)	y (m)	z (m)
OB1	-1.475	-1.100	0.25
OB2	0.750	-1.375	0.25
OB3	2.400	-1.725	0.25
OB4	2.700	0.925	0.25
OB5	1.275	3.050	0.25
OB6	-2.550	3.550	0.25
OB7	-3.800	0.425	0.25
Nine-Obstacle(OBS) Scene			
Obstacle	x (m)	y (m)	z (m)
OB1	-1.625	-1.325	0.25
OB2	0.700	-1.650	0.25
OB3	1.575	-3.775	0.25
OB4	2.900	-1.250	0.25
OB5	2.750	1.125	0.25
OB6	1.275	3.050	0.25
OB7	-1.600	4.025	0.25
OB8	-3.750	2.025	0.25
OB9	-3.650	-0.800	0.25

Obstacle geometry and dimensions are held constant across all experiments, and only the number and spatial distribution of active obstacles vary. This controlled design ensures that observed differences in navigation performance arise solely from changes in environmental complexity.

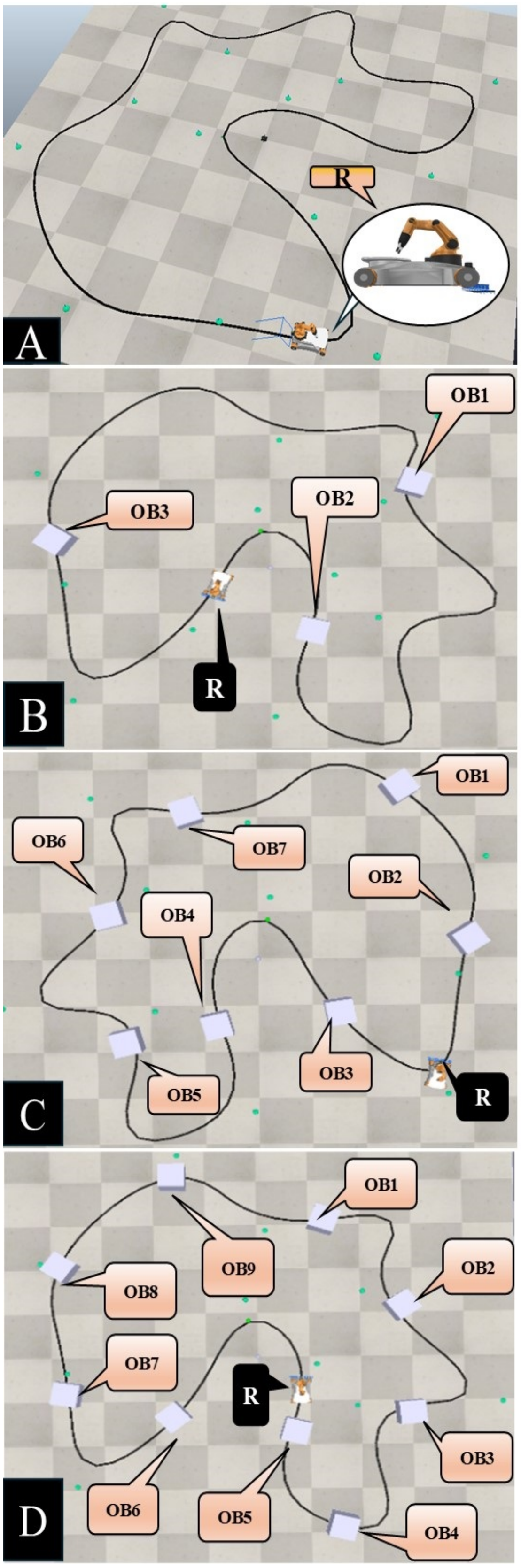

Figure 2. Kuka Robot with in different scenarios in CoppeliaSim: A: Line Follower Scenario without obstacle; B: Seven-Obstacle Scene ; C: Seven-Obstacle Scene D:Nine-Obstacle Scene

2.6. Performance Metrics

To quantify navigational quality, the system records several performance metrics during simulation (Table 3), including the total distance traveled, total elapsed time and average velocity, angular stability measured through cumulative, maximum, and minimum yaw-rate variations, and path deviation quantified as the minimum distance to a predefined reference trajectory. In addition, the evolution and convergence behavior of the PID controller parameters optimized using the BAS algorithm are monitored throughout the navigation process. After each simulation run, all generated maps, motion logs, and controller state variables are exported for post-simulation analysis. The recorded results include the navigation type, target speed, traveled distance, elapsed time, average speed, minimum, maximum, and average yaw-rate variations, and average path deviation errors, task success status, and the final converged values of the PID gains (Kp, Ki, and Kd).

Trajectory Tracking Accuracy The average lateral deviation is computed as

$$\bar{d} = \frac{1}{N}\sum_{i=1}^{N} \min_{j} \|\mathbf{p}_i - \mathbf{t}_j\|,$$

Here $\mathbf{p}_i$ shows the robot position at time step i and $\mathbf{t}_j$ represents the reference trajectory points. Along with this, the maximum and minimum lateral deviations are recorded to capture worst-case tracking errors and closest path adherence, respectively.

Control Smoothness Control smoothness is evaluated through changes in the robot's heading angle. The average angular change between consecutive time steps is calculated as

$$\bar{\Delta\theta} = \frac{1}{N-1}\sum_{i=2}^{N} |\theta_i - \theta_{i-1}|,$$

where θ_i is the normalized heading angle at time step i. Maximum and minimum angular deviations are also reported to quantify abrupt control actions and stable tracking intervals.

Efficiency Metrics Task efficiency is measured using the total distance traveled, total execution time, and average velocity. The total traveled distance is computed as

$$D_{total} = \sum_{i=2}^{N} \|\mathbf{p}_i - \mathbf{p}_{i-1}\|,$$

while execution time is defined as $T_{total} = t_N - t_1$. The average velocity is then given by $\bar{v} = D_{total}/T_{total}$.

Task Success Criteria Each experimental run is evaluated in a clear and consistent manner by classifying it as either a success or a failure. A run is considered successful when the robot is able to complete the full closed-loop trajectory, return to within 0.3 m of its initial starting position after traveling at least 2.0 m, and maintain stable tracking behavior by keeping its lateral deviation below 1.5 m for the entire duration of the task. In contrast, a run is labeled as a failure if the robot collides with any obstacle, deviates excessively from the reference path beyond the specified threshold, or becomes effectively stuck. Immobilization is identified when the robot's displacement falls below 0.05 m over a continuous 3-second interval, indicating a loss of meaningful forward progress. Finally, if the robot largely crosses the total path length of 32 meters then it is designated as failure also. .

Adaptive Parameter Analysis For configurations employing BAS optimization, the final converged PID gains (K_p^{final}, K_i^{final}, K_d^{final}) are recorded. These values are compared against the initial manually tuned gains to analyze the adaptive behavior of the controller and its influence on tracking performance.

3. Experimentation and Results

Table 4 presents the hyperparameter values used in the integrated BAS–SLAM navigation system, covering SLAM navigation, BAS optimization, path smoothing, control and tracking, environmental setup, and termination logic. Obstacle avoidance is handled using a 0.5 m detection threshold, with A* planning that applies a weighted obstacle penalty of 5 over a 4-cell neighborhood, along with a 3-cell grid inflation to keep a safe clearance. For BAS, a relatively small initial step size of 0.0015 is used together with an exponential decay factor of 0.95 to keep the adaptive PID tuning stable. Path smoothing gives more importance to obstacle clearance than to trajectory smoothness, with only limited curvature regularization, and typically converges within 150 iterations. Waypoint tracking applies a cubic–linear heading correction rule with velocity scaled to 80%, which helps achieve better transient response.

3.1. Experimental Configurations

The experimental research takes in consideration four varied control settings, which each depicts a gradual advancement in freedom, adaptability, and consciousness to the environment around. All the configurations are experimented in the predefined scenes that are proposed above and this will enable a direct comparison of behavior of controllers as the environment becomes intricate. The configuration is deliberately made to allow observing the individual effects of the adaptive control, obstacle avoidance and their combined effect on the performance of navigation to a greater degree.

Experiment 1: Baseline PID Control (Conventional) The first experiment uses a standard proportional–integral–derivative (PID) controller with empirically tuned gains ($K_p = 0.01$, $K_i = 0.0006$, $K_d = 0.002$) [1]. In this case, the robot performs only trajectory tracking along a predefined path. No adaptive gain tuning or obstacle avoidance is included. This configuration is treated as the baseline and is used as a reference point for evaluating all later experiments.

Experiment 2: BAS-Optimized PID Control The second experiment adds adaptivity by using a BAS-optimized PID controller. Here, the BAS algorithm adjusts the PID gains online based on feedback from trajectory deviation, which allows the controller to adapt to changes in path curvature and control disturbances. The initial step size is set to $d_0 = 0.002$, with the antenna distance defined as $D_0 = 0.9d_0$. Both values decay by 95% at each iteration, and the gain updates are applied every two control cycles. Obstacle detection and avoidance are not included in this setup, so the effect of adaptive control can be evaluated on its own.

Experiment 3: SLAM-Based Obstacle Avoidance The third experiment focuses on environmental perception and obstacle avoidance while keeping the PID gains manually tuned. A 360-degree LiDAR sensor is used to build an occupancy grid map with a resolution of 0.05 m over a 20×20 m workspace. When an obstacle is detected within a frontal safety zone of 0.8 m and $\pm 30°$, a collision-free path is planned using the A* algorithm. This configuration is used to study how mapping and replanning affect navigation performance without any adaptive control optimization.

Experiment 4: BAS–SLAM Integrated Control The final experiment represents the complete proposed framework, where BAS-based adaptive PID control is combined with SLAM-driven obstacle avoidance. In this setup, the controller gains are continuously optimized during execution, and the SLAM module provides real-time perception of the environment along with dynamic replanning. For safety, obstacles in the occupancy grid are inflated by a three-cell radius. Each re-calculated path is further improved using BAS-based smoothing with parameters $\alpha = 20$, $\gamma = 0.03$, and 200 optimization iterations before execution. This experiment is used to evaluate the combined impact

Table 3. Navigation Performance Metrics, Units, and Significance, Parameter(P),Target Speed(TS),Distance (Dist.),Average Speed (AS),Minimum Yaw Rate (Min θ),Minimum Yaw Rate (Min θ),Maximum Yaw Rate (Max θ),Average Yaw Rate (Avg θ),Maximum Error (Max Err),Average Error (Avg Err)

P	Definition	Unit	Significance
TS	Desired linear velocity set for the robot	m/s	Defines operating conditions and navigation difficulty
Dist	Total distance traveled during navigation	m	Indicates path efficiency; shorter distances imply more optimal trajectories
Time	Total elapsed time to complete the navigation task	s	Reflects navigation efficiency and system responsiveness
AS	Mean linear velocity over the entire trajectory	m/s	Measures motion smoothness and velocity consistency
Min θ	Minimum angular velocity variation observed during navigation	rad/s	Indicates lower bound of rotational motion and stability
Max θ	Maximum angular velocity variation observed during navigation	rad/s	High values may indicate abrupt turns or instability
Avg θ	Mean angular velocity variation over the trajectory	rad/s	Represents overall angular stability and smoothness
Min Err	Minimum deviation from the reference trajectory	m	Indicates best-case trajectory tracking accuracy
Max Err	Maximum deviation from the reference trajectory	m	Reflects worst-case deviation and robustness
Avg Err	Mean deviation from the reference trajectory	m	Primary indicator of path-following accuracy
Success	Task completion status	0/1	Measures reliability and feasibility of the navigation system
K_p	Proportional gain after BAS convergence	-	Controls responsiveness to instantaneous error
K_i	Integral gain after BAS convergence	-	Eliminates steady-state error and improves long-term accuracy
K_d	Derivative gain after BAS convergence	-	Provides damping and reduces oscillations

of adaptive control and perception-based planning, especially as obstacle density increases.

3.2. Scene I: No-Obstacle Scene (Baseline Path Following)

This experiment evaluates the four control configurations in an obstacle-free environment to isolate pure path-following and control behavior. Although the dataset label indicates three obstacles, no obstacles are present in the environment for this experiment. The reference trajectory remains unchanged, and no replanning or collision avoidance is required.Each controller is tested over a range of target speeds, from 3.5 m/s to 7.5 m/s, as summarized in Table 5. This set of tests is mainly used to establish a baseline in terms of stability, tracking accuracy, and how well performance scales with speed, before adding environmental complexity in later obstacle-based experiments.

3.3. Scene II : Obstacle Scene

The results for Experiment II with three obstacles are reported in Table 6. Since the obstacle layout is sparse, all methods show relatively stable navigation behavior. The Conventional approach achieves consistently high success rates across most target speeds, along with low average deviation and narrow minimum–maximum deviation ranges. This indicates a predictable and somewhat conservative behavior. The BAS-based method shows moderate performance, where successful runs are limited to certain speed ranges and the average deviation is slightly higher. This indicates that BAS remains sensitive to parameter tuning, even when the

Table 4. Hyperparameter Configuration for Integrated BAS–SLAM Navigation System

Category	Parameter	Value	Description
SLAM Navigation	Obstacle detection threshold (τ_{obs})	0.5 m	LiDAR distance trigger for path replanning
	Waypoint arrival threshold (τ_{arrival})	0.05 m	Distance for waypoint completion
	Trajectory deviation threshold	1.5 m	Maximum allowable path deviation
	Path subsampling step	$\lvert\Gamma\rvert/6$	Adaptive waypoint reduction factor
	Grid inflation thickness	3 cells	Obstacle boundary expansion
	A* obstacle penalty weight (w_{obs})	5	Cost multiplier for proximity to obstacles
	Obstacle penalty radius	4 cells	Neighborhood size for penalty computation
BAS Optimization	Initial step size (d_0)	0.0015	Starting perturbation magnitude
	Step decay factor (d_{decay})	0.95	Exponential decay per iteration
	Step increment ratio (d_{inc})	0.01	Additive step size adjustment
Path Smoothing	Smoothness weight (α)	15	Penalty for path curvature
	Obstacle avoidance weight (β)	5	Penalty for obstacle proximity
	Curvature penalty (γ)	0.01	Regularization for sharp turns
	Optimization iterations	150	Maximum BAS smoothing cycles
PID Controller	Proportional gain (K_p)	0.01	Initial proportional control gain
	Integral gain (K_i)	0.0006	Initial integral control gain
	Derivative gain (K_d)	0.002	Initial derivative control gain
Waypoint Tracking	Velocity scaling factor	0.8	Speed reduction during waypoint following
	Nonlinear control law	$-8\alpha^3 - 16\alpha$	Heading correction for angular error α
Grid Environment	Grid dimensions	400×400 cells	Occupancy map resolution
	Cell size	0.05 m	Spatial discretization
	Workspace coverage	20 m × 20 m	Total navigable area
Termination	Loop completion threshold	0.3 m	Distance to start for success
	Minimum completion distance	2.0 m	Required travel before loop check
	Stuck check interval	3.0 s	Time window for movement validation

environment has a low density of obstacles. In comparison, the SLAM and BAS-SLAM approaches have higher deviation values of the mean path and higher rotation of the heading angles. The main cause of this effect is the extra number of computations involved in terms of localization and mapping processes that remains even in the context of limited obstacles. Nevertheless, overall successes of SLAM-based methods are always relatively high, which means great robustness, but at the cost of the generation of relatively complicated trajectories.

In the case of 7 obstacles mentioned in Table 7 the influence on the performance is more significant and varies widely across the strategies used in control. Success rate of the conventional PID controller is greatly reduced especially at low and medium speed as well as a higher average deviation of the path, and the response becomes less stable with a bigger variance and less repeatability of results in the repeated trials. On the contrary, SLAM shows a similar success percentage regardless of all the assessed speeds. This dependability is, however, paired with great increases in the maximum and average deviations which means that the trajectories are more lengthy and complex as opposed to leading to complete failures. BAS-SLAM approach lies in the middle ground in that it has high success rates yet minimal growth on deviation as opposed to standalone SLAM. These findings indicate that incorporation of BAS is a valuable path direction process within a more cluttered environment.

In the worst-case scenario where there are nine obstacles as depicted in the Table 8, the drawbacks of the traditional PID as well as the BAS enhanced PID controllers are amplified. The traditional PID strategy has a sharp reduction in the success rate and failures are observed at almost all-speed settings and less chances of recovery. In the same sense, the BAS-enhanced PID technique fails, and the success rate is low with increased angular variation and path error especially in high speed where obstacles collision occur more often.On the other hand, SLAM is the most reliable in terms of completing a task with the mode of success rate being high with a significant change in angular variation and path deviation. This shows the extra calculation and planning efforts that are involved in US global mapping in highly cluttered surrounding. A similar situation can be seen with BAS -SLAM strategy, which is moderately successful and can partially reduce the rise in deviation, but its solutions become more unstable than in less overcrowded settings.There is also a noticeable trend to the results of the three

Table 5. Experiment I: No-Obstacle Scene (Baseline Path Following)

Exp Type	Spd	Dist	Time	AvgS	AvgA	MinD	MaxD	AvgD	S	Kp	Ki	Kd
	3.5	32.2099	84.566	0.3809	0.020772	0.0176	0.9184	0.4711	S	0.01	0.0006	0.002
	4.0	32.2788	80.8188	0.3994	0.019904	0.0204	0.9139	0.4674	S	0.01	0.0006	0.002
	4.5	32.3911	72.4722	0.4469	0.022658	0.0125	0.9254	0.4755	S	0.01	0.0006	0.002
	5.0	32.4868	62.4534	0.5202	0.023316	0.0124	0.9463	0.4725	S	0.01	0.0006	0.002
Conventional	5.5	32.6263	57.5834	0.5666	0.025775	0.0082	0.9569	0.4715	S	0.01	0.0006	0.002
	6.0	32.7826	56.0861	0.5845	0.029119	0.0131	0.9603	0.4711	S	0.01	0.0006	0.002
	6.5	32.9053	49.9718	0.6585	0.03149	0.0024	0.9762	0.4766	S	0.01	0.0006	0.002
	7.0	33.0074	47.3448	0.6972	0.032467	0.0075	0.9868	0.4766	S	0.01	0.0006	0.002
	7.5	25.861	35.5465	0.7275	0.033087	0.0108	1.5083	0.5626	F	0.01	0.0006	0.002
	Avg	31.8388	60.7603	0.5535	0.0265	0.01165	1.0102	0.4827	88.9	0.01	0.0006	0.002
	3.5	32.2321	86.8375	0.3712	0.020852	0.0106	0.9152	0.4741	S	0.01	0.0006	0.002
	4.0	32.2651	80.9576	0.3985	0.01959	0.0151	0.9196	0.4751	S	0.01	0.0006	0.002
	4.5	32.4333	67.3368	0.4817	0.022885	0.0103	0.9438	0.4762	S	0.01	0.0006	0.002
	5.0	32.4904	62.2868	0.5216	0.02286	0.0073	0.9495	0.4714	S	0.01	0.0006	0.002
SLAM	5.5	32.6267	56.9445	0.573	0.026236	0.0057	0.9515	0.4754	S	0.01	0.0006	0.002
	6.0	32.7803	65.3865	0.5013	0.028627	0.0087	0.9567	0.4742	S	0.01	0.0006	0.002
	6.5	32.9135	50.0195	0.658	0.030519	0.0146	0.9769	0.4728	S	0.01	0.0006	0.002
	7.0	32.9966	47.7979	0.6903	0.031662	0.0159	0.9907	0.475	S	0.01	0.0006	0.002
	7.5	25.8206	35.3885	0.7296	0.030269	0.0279	1.5419	0.5596	F	0.01	0.0006	0.002
	Avg	31.8398	61.4395	0.54724	0.0259	0.0129	1.0162	0.4837	88.9	0.01	0.0006	0.002
	3.5	21.1771	37.7285	0.5613	0.109175	0.0226	1.5039	0.5065	F	0.010045	0.006701	-0.002852
	4.0	2.0098	4.7304	0.4249	0.074898	0.0102	0.5297	0.1892	S	0.000941	-0.002847	-0.003252
	4.5	31.9996	67.9545	0.4709	0.021151	0.0131	0.8668	0.4717	S	0.020417	0.006252	0.004821
	5.0	13.7625	24.4988	0.5618	0.057204	0.0478	1.5305	0.5481	F	0.004109	0.000574	0.006249
BAS+PID	5.5	31.9331	55.9335	0.5709	0.026092	0.0169	0.8735	0.4703	S	0.009818	0.006986	0.00315
	6.0	32.7305	67.0634	0.4881	0.031139	0.0229	0.885	0.4723	S	0.009435	-0.000916	0.003552
	6.5	21.7079	36.3978	0.5964	0.094513	0.0282	0.9391	0.3585	F	0.01011	-0.003856	0.005581
	7.0	48.8759	81.8994	0.5968	0.115049	0.02	0.5615	0.2739	F	0.019023	-0.007633	0.008548
	7.5	9.8938	14.7839	0.6692	0.049691	0.0364	1.5165	0.5952	F	0.016737	0.000296	0.000499
	Avg	23.7878	43.443	0.5489	0.0643	0.0242	1.0229	0.43174	44.4	0.0111	0.0006	0.0029
	3.5	73.866	102.8525	0.7182	0.166291	0.0627	1.5034	1.0339	F	0.008611	-0.000964	-0.003087
	4.0	6.2473	12.0625	0.5179	0.118092	0.0434	0.4225	0.2291	F	0.010126	-0.007275	0.009231
	4.5	31.9082	67.9069	0.4699	0.021042	0.0106	0.8633	0.4702	S	0.010719	0.008812	0.0074
	5.0	31.7764	61.3877	0.5176	0.022927	0.013	0.868	0.4693	S	0.017894	0.009042	0.011543
BAS_SLAM	5.5	4.8464	8.5399	0.5675	0.058871	0.0733	1.5085	0.697	F	0.006711	-0.001194	0.00012
	6.0	34.843	64.7123	0.5384	0.042355	0.0172	0.8966	0.4697	S	0.008436	0.001298	0.006061
	6.5	32.3748	54.6155	0.5928	0.098192	0.0423	0.8928	0.3623	F	0.017141	0.00438	-0.005184
	7.0	12.8185	19.5899	0.6543	0.060689	0.023	1.5078	0.5826	F	0.009695	-0.00232	0.00165
	7.5	104.5929	107.6397	0.9717	0.15245	0.015	0.9211	0.4295	F	0.013515	0.00678	-0.000427
	Avg	37.0303	55.4785	0.6164	0.0823	0.0333	1.0426	0.5270	33.3	0.0114	0.0020	0.0030

result tables together (merged) (Table 6, Table 7, and Table 8), as autonomous results with obs7, and as autonomous results with obs9. The performance of non-SLAM methods worsens as the obstacle density increases, and the performance of SLAM methods does not reduce significantly, only increasing more complicated and lengthy paths. The hybrid BAS-SLAM scheme always trades-off between the success rate and trajectory efficiency, achieving better results in dense environments compared to the conventional and the BAS controllers and a smaller amount of deviation overhead as compared to the pure SLAM. The noted observations highlight the Cy peers importance of obstacle density in determining the most appropriate control strategy to use in autonomous navigation.

Some trials took negative or near-zero values of the integral and derivative gains although this is largely explained by the fact that the PID tuning of the BAS is not constrained. Since the optimizer focuses on short-term error reduction, it can suppress integral buildup or derivative action, especially during transient phases or when conditions change quickly. Although these gain values are mathematically acceptable and did not cause instability or divergence in the simulations, they can reduce robustness margins and may introduce stability risks if applied directly to real hardware. From a physical point of view, very small or negative gains can sometimes help limit windup and soften aggressive corrections when feedback is noisy. At the same time, their appearance also suggests a degree of overfitting to short-term or transient behaviors rather than long-term stability. This observation points to the need for stability-aware constraints and bounded gain optimization, which will be addressed in future work to ensure that the resulting controller parameters remain physically meaningful and safe for deployment on real robotic systems.

Table 6. Experiment II: Obstacle Scene (3 Obstacles)

Exp Type	Spd	Dist	Time	AvgS	AvgA	MinD	MaxD	AvgD	Succ	Kp	Ki	Kd
BAS	3.5	7.4493	60.5343	0.1232	0.08823	0.02633	0.85973	0.3924	F	0.00839	0.00105	0.00509
	4.0	11.2219	40.7454	0.2862	0.05355	0.01355	0.82993	0.35863	F	0.00988	0.00317	0.00573
	4.5	20.0679	71.0671	0.2706	0.17214	0.16953	0.88267	0.5393	S	0.01060	-0.00024	0.00411
	5.0	6.9174	26.9071	0.2573	0.15200	0.22903	0.91700	0.5628	F	0.01099	-0.00031	-0.00276
	5.5	7.7625	20.7646	0.3746	0.03697	0.01090	0.80520	0.3248	F	0.01104	0.00626	0.00183
	6.0	12.2938	35.3638	0.3555	0.07030	0.01543	1.02833	0.48253	S	0.00996	0.00264	0.00007
	6.5	12.6917	32.6743	0.4004	0.08072	0.03143	1.07383	0.38783	S	0.00837	0.00199	0.00214
	7.0	22.2447	65.2406	0.3689	0.11934	0.01530	0.99050	0.43497	F	0.00556	-0.00157	-0.00050
	7.5	5.8249	18.1939	0.3471	0.07951	0.10210	0.96367	0.6199	F	0.01077	0.00776	-0.00141
	Avg	11.8305	41.2768	0.3093	0.09475	0.06818	0.92787	0.45591	33.33	0.00951	0.00231	0.00159
Conventional	3.5	7.6006	55.8043	0.1376	0.05819	0.02500	0.92157	0.3828	F	0.01	0.0006	0.002
	4.0	12.7047	37.2931	0.3406	0.02633	0.01223	0.84513	0.3705	S	0.01	0.0006	0.002
	4.5	33.3897	89.8016	0.3586	0.04935	0.00850	0.93107	0.43833	S	0.01	0.0006	0.002
	5.0	12.5739	31.6186	0.3974	0.03116	0.00813	0.91940	0.37077	S	0.01	0.0006	0.002
	5.5	13.7548	35.2678	0.3841	0.03581	0.00850	0.94387	0.36567	S	0.01	0.0006	0.002
	6.0	9.0859	22.4835	0.3985	0.03156	0.00847	0.87927	0.35793	F	0.01	0.0006	0.002
	6.5	17.0436	34.8000	0.4737	0.04784	0.01103	1.13917	0.39543	S	0.01	0.0006	0.002
	7.0	16.5471	32.1148	0.5155	0.05097	0.00670	1.06723	0.40857	S	0.01	0.0006	0.002
	7.5	19.3360	41.4046	0.4919	0.05713	0.01143	1.23557	0.43727	S	0.01	0.0006	0.002
	Avg	15.7818	42.2876	0.3887	0.04315	0.01111	0.98692	0.39192	77.78	0.01	0.0006	0.002
SLAM	3.5	24.2039	182.1867	0.1329	0.12803	0.01920	0.94930	0.9440	S	0.01	0.0006	0.002
	4.0	24.1520	145.9888	0.1654	0.11335	0.02090	0.95910	0.9577	S	0.01	0.0006	0.002
	4.5	11.8873	67.1640	0.1770	0.15249	0.01090	0.82850	0.5541	S	0.01	0.0006	0.002
	5.0	12.0384	63.8797	0.1885	0.17445	0.01060	0.81340	0.5534	S	0.01	0.0006	0.002
	5.5	10.6880	52.9275	0.2019	0.11277	0.00640	0.81270	0.5439	S	0.01	0.0006	0.002
	6.0	10.5855	51.9019	0.2040	0.12630	0.01580	0.80810	0.5125	S	0.01	0.0006	0.002
	6.5	25.6007	100.2224	0.2554	0.12334	0.02110	1.24150	1.2546	S	0.01	0.0006	0.002
	7.0	13.6213	55.7034	0.2445	0.08779	0.03490	1.40630	0.5520	S	0.01	0.0006	0.002
	7.5	20.4292	69.6245	0.2934	0.15505	0.01610	1.04260	1.0397	S	0.01	0.0006	0.002
	Avg	17.0229	87.7332	0.2070	0.13040	0.01732	0.98461	0.76799	100	0.01	0.0006	0.002
BAS_SLAM	3.5	51.4601	336.7750	0.1528	0.70845	0.06900	0.92010	0.7049	S	0.00503	-0.00648	0.00393
	4.0	30.9795	161.2184	0.1922	0.15251	0.00590	1.55290	1.7550	S	0.00808	0.00009	0.00195
	4.5	40.0635	207.8815	0.1927	0.12706	0.00940	1.55090	1.3064	S	0.00781	-0.00011	-0.00191
	5.0	14.1063	57.0231	0.2474	0.13650	0.03390	1.65730	3.9050	S	0.00788	0.00054	0.00407
	5.5	7.1845	31.6622	0.2269	0.04951	0.03600	1.41150	0.5570	F	0.01006	-0.00022	-0.00207
	6.0	10.3217	51.3752	0.2009	0.12721	0.01700	0.85050	0.4977	S	0.01757	0.00350	0.00624
	6.5	24.7639	105.4360	0.2349	0.21781	0.02560	1.55530	3.2763	S	0.01402	-0.00047	0.00790
	7.0	10.6358	45.2603	0.2350	0.13338	0.00830	0.78010	0.5220	S	0.00859	0.00051	0.00107
	7.5	16.8908	59.1310	0.2857	0.28768	0.01050	1.55570	3.4930	S	0.00943	0.00703	0.00539
	Avg	22.9340	117.3070	0.2187	0.21557	0.02396	1.31492	1.7797	88.89	0.00983	0.00049	0.00295

4. Conclusion & Future Work

This paper presented a fully integrated, real-time navigation framework for an omnidirectional mobile robot that unifies local perception, occupancy-grid mapping, global replanning, and adaptive control through the application of the BAS metaheuristic. Unlike traditional approaches that utilize BAS as a static or offline optimizer, this work embedded a single-agent BAS into the online control loop and trajectory smoothing pipeline.To this end, we propose an integrated BAS-enabled navigation framework implemented on an omnidirectional robot in CoppeliaSim. The system combines several components into a single pipeline. First, a nine-sensor IR array provides a line-tracking error signal, while a 2D LiDAR sensor is used to construct a global occupancy grid around the robot. Unknown cells are progressively classified as free or occupied, closed regions of unknown space are filled to avoid spurious voids, and obstacles are inflated to account for the robot footprint and a safety margin. When the forward LiDAR arc detects an obstacle within a pre-specified distance, the robot temporarily abandons pure line following and triggers an avoidance behavior. A modified A* search is then executed on the occupancy grid to compute a collision-free path to a rejoin point selected on the original reference trajectory. The resulting discrete path is subsampled and smoothed through a BAS-based optimization routine that reduces curvature and eliminates unnecessary zig-zag segments.In parallel, the low-level tracking controller is implemented as a PID regulator whose gains are adapted online using BAS. Rather than fixing (K_p, K_i, K_d) a priori, the algorithm maintains a single beetle agent in the PID gain space. At each iteration,

Table 7. Experiment II: Obstacle Scene (7 Obstacles)

Exp Type	Spd	Dist	Time	AvgS	AvgA	MinD	MaxD	AvgD	Succ	Kp	Ki	Kd
BAS	3.5	4.7494	22.6557	0.2109	0.04298	0.01143	0.29863	0.11287	S	0.00947	0.00152	0.00052
	4.0	3.8410	22.7126	0.1689	0.05997	0.01268	0.27660	0.12720	F	0.01071	0.00031	0.00550
	4.5	4.6137	20.4845	0.2183	0.05647	0.00657	0.54167	0.20380	F	0.01074	0.00102	-0.00023
	5.0	23.5717	65.2458	0.3336	0.16729	0.00887	0.66203	0.24757	S	0.00701	-0.00296	-0.00441
	5.5	13.1728	41.3683	0.2536	0.10943	0.01190	0.67010	0.29260	F	0.01097	-0.00242	0.00189
	6.0	9.1851	39.9578	0.3108	0.12205	0.01780	1.01543	0.47770	F	0.00943	0.00016	0.00087
	6.5	4.0156	15.1621	0.2650	0.05684	0.01570	0.47870	0.20627	F	0.00998	0.00428	0.00098
	7.0	8.1918	24.8762	0.3296	0.09119	0.01993	0.92630	0.37707	S	0.00601	0.00366	0.00033
	7.5	13.1505	48.6654	0.2743	0.12692	0.00843	0.95863	0.53717	S	0.01440	0.00100	-0.00075
	Avg	9.3880	33.4587	0.2628	0.09257	0.01259	0.64757	0.28691	44.44	0.00986	0.00073	0.00052
Conventional	3.5	2.1903	21.9816	0.1018	0.02903	0.00893	0.22503	0.11083	F	0.01	0.0006	0.002
	4.0	4.8981	19.1405	0.2567	0.04290	0.00550	0.31470	0.13057	S	0.01	0.0006	0.002
	4.5	4.7945	18.7748	0.2592	0.04003	0.00427	0.31513	0.12330	S	0.01	0.0006	0.002
	5.0	3.4328	13.2534	0.2548	0.03160	0.00530	0.29117	0.14190	F	0.01	0.0006	0.002
	5.5	8.4042	22.3932	0.3685	0.04766	0.00467	0.87610	0.26690	S	0.01	0.0006	0.002
	6.0	6.5749	20.7882	0.3207	0.05036	0.00607	0.84130	0.23690	S	0.01	0.0006	0.002
	6.5	5.9606	14.9639	0.3805	0.04957	0.00457	0.81300	0.26700	S	0.01	0.0006	0.002
	7.0	7.9478	20.4572	0.3850	0.05151	0.00470	0.91063	0.27080	S	0.01	0.0006	0.002
	7.5	5.9598	18.0509	0.3376	0.04141	0.00800	0.56070	0.26573	F	0.01	0.0006	0.002
	Avg	5.5737	18.8671	0.2961	0.04267	0.00578	0.57197	0.20155	66.67	0.01	0.0006	0.002
SLAM	3.5	25.0572	112.8654	0.2220	0.35877	0.11520	1.60310	14.0136	S	0.01	0.0006	0.002
	4.0	25.0957	104.7046	0.2397	0.51103	0.12640	1.60230	13.0315	S	0.01	0.0006	0.002
	4.5	30.0875	121.3141	0.2480	0.66170	0.01890	1.58140	14.1179	S	0.01	0.0006	0.002
	5.0	30.4273	115.2417	0.2640	0.82801	0.11660	1.60630	18.9272	S	0.01	0.0006	0.002
	5.5	25.4030	84.9669	0.2990	0.67981	0.12670	1.60710	13.4289	S	0.01	0.0006	0.002
	6.0	25.2238	79.0657	0.3190	0.74735	0.01780	1.58430	9.9544	S	0.01	0.0006	0.002
	6.5	26.0128	82.0594	0.3170	1.00553	0.01940	1.53010	10.2244	S	0.01	0.0006	0.002
	7.0	25.7552	75.6562	0.3404	0.92545	0.01750	1.54930	9.8860	S	0.01	0.0006	0.002
	7.5	24.2461	77.3088	0.3136	1.08091	0.02330	1.52230	9.6842	S	0.01	0.0006	0.002
	Avg	26.3676	94.7981	0.2847	0.75539	0.06464	1.57624	12.5853	87.50	0.01	0.0006	0.002
BAS_SLAM	3.5	25.1635	117.0608	0.2150	0.42967	0.12600	1.59430	12.7116	S	0.01336	0.00216	0.00430
	4.0	86.1082	369.0984	0.2333	1.14430	0.01310	1.61420	31.3109	F	0.00372	0.00036	0.00001
	4.0	24.9911	149.9671	0.1666	0.34753	0.12220	1.19670	1.3761	F	0.00824	-0.00199	0.00048
	4.5	29.8955	118.9916	0.2512	0.73463	0.02420	1.58560	14.8764	S	0.01274	0.00192	0.00125
	5.0	26.2946	93.0735	0.2825	0.21619	0.01400	1.25670	3.6004	S	0.00539	0.00063	0.00745
	5.5	24.8504	83.6719	0.2970	0.71138	0.01800	1.58310	9.6071	S	0.01006	0.00625	0.00249
	6.0	28.9711	122.2886	0.2369	0.24113	0.04370	1.57420	0.8717	S	0.00935	-0.00109	0.00752
	6.5	25.0118	76.3589	0.3276	0.82024	0.02060	1.60220	9.7978	S	0.00930	0.00999	0.00880
	7.0	24.1488	75.0051	0.3220	0.28092	0.01280	1.17720	3.0404	S	0.00904	0.00095	0.00215
	7.5	25.3340	72.4251	0.3498	0.85156	0.11550	1.60370	12.5244	S	0.01478	0.00061	0.00221
	Avg	32.8451	128.9867	0.2741	0.59421	0.04268	1.46596	9.6672	87.50	0.00918	0.00196	0.00359

mirrored perturbations of the gain vector are evaluated using the path deviation with respect to the reference trajectory as the performance signal.The BAS update mechanism changes the PID gains towards the direction that makes tracking performance better and increasingly reduces the step size such that convergence remains constant. Thus, the controller dynamically adjusts its responsiveness and damping characteristics based on the prevailing operation environment, which allows it easily to respond to perturbation like obstacle encounters and properly act appropriately during the transition between usual path following and avoidance responses. In this experiment various control and navigation strategies were analyzed in conditions where the environment became more complex with initially a no-obstacle baseline to the complex experiment with 3, 7 and 9 obstacles. The displayed success rates show that the core constraints are to do with the robustness at the software level, i.e. the stability of the replanning process, the responsiveness of the controllers to stress, etc. and not with the inherent weaknesses of the underlying algorithms.In the no-obstacle scenario, both the Conventional controller and SLAM achieved a high success rate of 88.9%, indicating stable and reliable path following under minimal environmental uncertainty. In contrast, the BAS-based approaches performed noticeably worse: BAS+PID achieved a success rate of only 44.4%, while BAS+SLAM dropped further to 33.3%. This behavior suggests that BAS-driven online gain adaptation is particularly sensitive in simple, unconstrained settings, where limited corrective feedback can amplify issues related to timing, synchronization, and convergence at the software level. By comparison, Conventional PID and SLAM rely on

Table 8. Experiment II: Obstacle Scene (9 Obstacles)

Exp Type	Spd	Dist	Time	AvgS	AvgA	MinD	MaxD	AvgD	Succ	Kp	Ki	Kd
BAS	3.5	2.3447	13.3058	0.1831	0.08081	0.10177	0.35867	0.22917	F	0.01280	0.00355	-0.00016
	4.0	4.0593	15.6739	0.2402	0.10748	0.07933	0.45280	0.21660	F	0.00868	0.00529	-0.00251
	4.5	5.3791	21.4226	0.2546	0.10028	0.07960	0.74463	0.35640	S	0.00893	0.00078	-0.00259
	5.0	2.4946	15.1983	0.1970	0.04217	0.06390	0.32523	0.16807	F	0.01147	0.00026	0.00065
	5.5	9.9706	38.2207	0.2567	0.07135	0.09013	0.78607	0.26040	F	0.00862	0.00165	0.00459
	6.0	10.8729	64.5105	0.1684	0.09788	0.04997	0.92687	0.50597	F	0.00672	0.00042	0.00079
	6.5	8.0722	26.8512	0.2857	0.08226	0.03343	0.52140	0.21823	F	0.01101	-0.00254	0.00238
	7.0	5.4526	17.1516	0.3175	0.05818	0.07233	1.03487	0.33040	S	0.00733	0.00022	0.00079
	7.5	5.8368	20.1635	0.3067	0.07354	0.06720	0.90437	0.44853	S	0.01045	-0.00038	-0.00003
	Avg	6.0536	25.8331	0.2455	0.07933	0.07085	0.67277	0.30375	33.33	0.00956	0.00103	0.00044
Conventional	3.5	1.6949	15.2141	0.1114	0.02774	0.10043	0.22540	0.16670	F	0.01	0.0006	0.002
	4.0	1.7557	12.2375	0.1434	0.03743	0.09767	0.22933	0.17743	F	0.01	0.0006	0.002
	4.5	1.6493	10.2556	0.1725	0.03170	0.09627	0.22733	0.16793	F	0.01	0.0006	0.002
	5.0	2.2728	17.2475	0.1336	0.02525	0.09427	0.24523	0.17400	F	0.01	0.0006	0.002
	5.5	2.2025	15.2075	0.1451	0.02721	0.09047	0.24287	0.17753	F	0.01	0.0006	0.002
	6.0	3.5024	29.2713	0.1211	0.03119	0.07207	0.27943	0.17530	F	0.01	0.0006	0.002
	6.5	2.9323	17.8229	0.1580	0.02979	0.07893	0.56057	0.40010	F	0.01	0.0006	0.002
	7.0	7.3496	20.6122	0.2507	0.03627	0.05060	0.56280	0.25597	F	0.01	0.0006	0.002
	7.5	6.5346	14.4870	0.4297	0.06691	0.03990	0.88727	0.37163	S	0.01	0.0006	0.002
	Avg	3.3216	16.9284	0.1851	0.03483	0.08007	0.38447	0.22962	11.11	0.01	0.0006	0.002
SLAM	3.5	24.3111	106.8725	0.2275	1.59158	0.12150	1.60290	62.2333	S	0.01	0.0006	0.002
	4.0	24.9166	126.3872	0.1971	0.94029	0.01690	1.59960	22.6317	S	0.01	0.0006	0.002
	4.5	24.5180	106.0611	0.2312	2.01613	0.11750	1.60110	56.1192	S	0.01	0.0006	0.002
	5.0	0.0000	0.3231	0.0000	0.00008	0.21430	0.21430	0.21430	F	0.01	0.0006	0.002
	5.0	28.4884	105.1486	0.2709	0.46567	0.03120	1.60350	12.6923	S	0.01	0.0006	0.002
	5.5	28.7900	99.1844	0.2903	0.53721	0.03130	1.60270	12.8505	S	0.01	0.0006	0.002
	6.0	28.8432	92.1667	0.3129	0.58034	0.04110	1.60170	12.6876	S	0.01	0.0006	0.002
	6.5	29.4104	92.4220	0.3182	0.73117	0.02750	1.59740	12.9901	S	0.01	0.0006	0.002
	7.0	3.0467	10.6049	0.2873	0.00979	0.03420	2.29520	0.9149	F	0.01	0.0006	0.002
	7.5	26.7282	86.8059	0.3079	0.94434	0.12470	1.60710	11.9550	S	0.01	0.0006	0.002
	Avg	21.6379	79.9004	0.2462	0.69167	0.07097	1.52473	15.8951	77.78	0.01	0.0006	0.002
BAS_SLAM	3.5	24.3204	112.0670	0.2170	1.65098	0.11400	1.60570	60.9469	S	0.01124	-0.00277	0.00399
	4.0	24.2226	110.7509	0.2187	1.73186	0.11740	1.60290	54.6328	S	0.00779	0.00137	0.00476
	4.5	14.5247	83.6655	0.1736	0.84033	0.08210	1.60020	15.9307	F	0.00863	0.00443	0.00295
	5.0	27.9862	103.0170	0.2717	0.48483	0.02550	1.60330	13.2254	S	0.01376	0.00423	-0.00048
	5.5	6.9787	34.6390	0.2015	0.07698	0.02320	2.27910	0.8946	F	0.01407	-0.00427	-0.00434
	6.0	29.0182	83.8885	0.3459	0.13463	0.03340	1.82720	3.1230	S	0.00891	-0.00209	0.00285
	6.5	20.7813	73.6020	0.2823	0.24098	0.01720	2.25210	2.5618	F	0.01192	-0.00309	-0.00100
	7.0	2.6025	9.5713	0.2719	0.02053	0.04740	1.84900	0.7324	F	0.00936	-0.00061	0.00423
	7.0	28.9745	119.3043	0.2429	0.33037	0.12920	1.26990	1.3492	S	0.01344	-0.00440	0.00373
	7.5	28.4239	84.7526	0.3354	1.02754	0.12460	1.58840	14.9128	S	0.00897	-0.00124	0.00470
	Avg	19.9113	74.0550	0.2657	0.39452	0.06033	1.78365	6.59124	50.00	0.01113	-0.00088	0.00158

fixed gains or map-based feedback, which tend to remain stable in open environments.When constraints are added to the objective, it becomes more constraint-based. The situations where the obstacle is met also lead to more frequent avoidance actions, replanning, and updates of the localization, which will also provide more comprehensive feedback to the controller. These interactions appear to implicitly regularize the BAS adaptation process by restricting excessive exploration and guiding gain updates toward more meaningful corrective directions. As a result, performance improves in obstacle-rich scenes, particularly for the BAS+SLAM configuration. This effect is most evident in the 3-obstacle scenario, where BAS+SLAM attains an 88.9% success rate compared to only 33.3% in the no-obstacle case, while SLAM achieves a full 100% success rate.This stabilizing effects start to be counteracted by the increasing complexity of the environment as density of obstacles grows further resulting in increasingly difficult trade-offs between navigation difficulty and stability of adaptation.In the 7-obstacle scene, both the BAS+SLAM and SLAM have 87.5% success and it can be concluded that with a moderate level of complexity, structured environmental feedback has full potential of making up the adaptive instability. In contrast, in the 9-obstacle scenario, the excessive constraints start to amplify software-level sensitivities. BAS+SLAM success drops to 50%, while SLAM still maintains 77%. This shows that although obstacles can initially improve the stability of adaptive control, very dense environments expose weaknesses related to software synchronization and controller switching. Future work will aim to verify whether this feedback-driven stabilization effect also appears beyond simulation by carrying out experiments on real robotic platforms. Real-world testing is necessary to understand how sensor noise, actuator delays, and hardware limitations affect BAS-based adaptive control, especially in low-feedback situations such as open, obstacle-free spaces. Although a single-beetle BAS formulation is employed for computational efficiency and real-time feasibility, we acknowledge that this choice may limit robustness in high-dimensional or noisy gain spaces. More advanced variants, such as multi-agent BAS or improved schemes like IBSO, could offer enhanced exploration and noise resilience, and are therefore identified as promising directions for future work to further strengthen robustness and performance. In addition, improving software robustness through tighter coordination between perception, planning, and control modules will be important for reducing failures in dense obstacle scenarios. Adding explicit stabilization or convergence constraints to the BAS adaptation process could help limit uncontrolled exploration in unconstrained environments, improving overall reliability without depending on environmental obstacles to provide corrective feedback.

5. Acknowledgements

The authors would like to thank RARL (Robotic Automation Research Lab, School of Information Engineering, Jiangxi University of Science and Technology, No. 86, Hongqi Ave., Ganzhou 341000, China.

References

[1] Moshayedi AJ, Zanjani SM, Xu D, Chen X, Wang G, Yang S. Fusion BASED AGV robot navigation solution comparative analysis and VREP simulation. In: 2022 8th Iranian Conference on Signal Processing and Intelligent Systems (ICSPIS). IEEE; 2022. p. 1-11.
[2] Zhang GB, Li HB, Liu XT, Peng YJ. Simulation budget allocation for improving scheduling and routing of automated guided vehicles in warehouse management. Journal of the Operations Research Society of China. 2025;13(3):775-809.
[3] Lepore M, Serra D, Maccioni R. Leveraging artificial intelligence and optimization for agile AGV scheduling in an edge-to-cloud manufacturing framework. Soft Computing. 2025;29(13):5041-69.
[4] Mumuni Q, Olayiwola-Mumuni A, Yussouff A. The advent of the proportional integral derivative controller: a review. Journal of Advances in Engineering Technology. 2023;(2):5-22.
[5] Tan W, Han W, Xu J. State-space PID: a missing link between classical and modern control. IEEE Access. 2022;10:116540-53.
[6] de Omena RA, Santos DF, Perkusich A, Valadares DC. Two-tier MPC architecture for AGVs navigation assisted by edge computing in an industrial scenario. Internet of Things. 2023;21:100666.
[7] Jiandong Q, Minan T, Fan Y, Jiajia R, Dingqiang L. Research on port AGV trajectory tracking control based on improved fuzzy sliding mode control. Archives of Transport. 2024;69(1):7-20.
[8] Wang Q, Xiang Z, Liu J, Zhang D, Wu S. Path Tracking Method of Forklift AGV Based on Fuzzy Adaptive PID. In: 2024 4th International Conference on Industrial Automation, Robotics and Control Engineering (IARCE). IEEE; 2024. p. 127-32.
[9] Lin Y, Hue G, Wang L, Li Q, Zhu J. A multi-AGV routing planning method based on deep reinforcement learning and recurrent neural network. IEEE/CAA Journal of Automatica Sinica. 2023;11(7):1720-2.
[10] Niu Q, Fu Y, Dong X. Omnidirectional AGV Path Planning Based on Improved Genetic Algorithm. World Electric Vehicle Journal. 2024;15(4):166.
[11] Azimi M, Kolahdooz A, Eftekhari SA. An optimization on the DIN1. 2080 alloy in the electrical discharge machining process using ANN and GA. Journal of Modern Processes in Manufacturing and Production. 2017;6(1):33-47.
[12] Moshayedi AJ, Li J, Sina N, Chen X, Liao L, Gheisari M, et al. Simulation and validation of optimized pid controller in agv (automated guided vehicles) model using pso and bas algorithms. Computational Intelligence and Neuroscience. 2022;2022(1):7799654.
[13] Song J. Automatic Guided Vehicle Global Path Planning Considering Multi-objective Optimization and Speed Control. Sensors & Materials. 2021;33.
[14] Liao X, Wang Y, Xuan Y, Wu D. AGV path planning model based on reinforcement learning. In: 2020 Chinese automation congress (CAC). IEEE; 2020. p. 6722-6.
[15] Moshayedi AJ, Li J, Sina N, Chen X, Liao L, Gheisari M, et al. Simulation and Validation of Optimized PID Controller in AGV (Automated Guided Vehicles) Model Using PSO and BAS Algorithms. Computational Intelligence and Neuroscience. 2023;2022:1-22.
[16] Durojaye A, Kolahdooz A, Nawaz A, Moshayedi AJ. Immersive horizons: exploring the transformative power of virtual reality across economic sectors. EAI Endorsed Trans AI Robot. 2023;2:e6.
[17] Khan AT, Li S, Li Z. Obstacle avoidance and model-free tracking control for home automation using bio-inspired approach. Advanced Control for Applications: Engineering and Industrial Systems. 2022;4(1):e63.
[18] Jianning Z, Jiaxin L, Songyi D, Rui G. Beetle Antennae Search guided RRT* for path planning of GIS inspection and maintenance robot. In: 2021 6th International Conference on Automation, Control and Robotics Engineering (CACRE). IEEE; 2021. p. 102-7.
[19] Latifi Rostami SA, Kolahdooz A, Chung H, Shi M, Zhang J. Robust topology optimization of continuum structures with smooth boundaries using moving morphable components. Structural and Multidisciplinary Optimization. 2023;66(6):121.
[20] Qian Q, Deng Y, Sun H, Pan J, Yin J, Feng Y, et al. Enhanced beetle antennae search algorithm for complex and unbiased optimization. Soft Computing. 2022;26(19):10331.

[21] Liang Q, Zhou H, Yin Y, Xiong W. An improved beetle antennae search path planning algorithm for vehicles. PLoS one. 2022;17(9):e0274646.

[22] Zha Y, Huang Z, Shi H. Global Path Planning Method Based on Improved Beetle Antenna Search Algorithm for Unmanned Surface Vehicle. In: 2023 2nd International Symposium on Control Engineering and Robotics (ISCER). IEEE; 2023. p. 356-9.

[23] Yu Z, Yuan J, Li Y, Yuan C, Deng S. A path planning algorithm for mobile robot based on water flow potential field method and beetle antennae search algorithm. Computers and Electrical Engineering. 2023;109:108730.

[24] Lyu Y, Mo Y, Yue S, Hong L. A mobile robot path planning using improved beetle swarm optimization algorithm in static environment. Journal of Intelligent & Fuzzy Systems. 2023;45(6):11453-79.

[25] Chen C, Cao L, Chen Y, Chen B, Yue Y. A comprehensive survey of convergence analysis of beetle antennae search algorithm and its applications. Artificial Intelligence Review. 2024;57(6):141.

[26] Dwivedi A, Khan AT, Li S. Comparative Analysis of BAS and PSO in Image Transformation Optimization. 2025.

[27] Jiang Z, Zhang X, Liu G. Trajectory tracking control of a 6-DOF robotic arm based on improved FOPID. International Journal of Dynamics and Control. 2025;13(4):137.

[28] Roy AS, Das A. Advanced Path Tracking and Traffic Management Using IR Sensors and Timed Automata. Journal of Robotics Research (JRR). 2024;1(1).

[29] Moshayedi AJ, Nasab STM, Khan ZH, Khan AS. Meta-heuristic Algorithms as an Optimizer: Prospects and Challenges (Part II). Engineering Applications of AI and Swarm Intelligence. 2024:155-80.

[30] Moshayedi AJ, Xie Y, Sharifdoust M, Khan AS. Evaluating OMNI Robot Navigation with SLAM in CoppeliaSim: Hemangiomas and Nonhomogeneous Paths. Journal of Robotics Research (JRR). 2024;1(1):7-14.

[31] Latifi Rostami SA, Kolahdooz A, Lim HJ. Enhancing structural efficiency with robust evolutionary topology optimization for sustainable engineering. Engineering Optimization. 2025:1-22.

[32] Moshayedi AJ, Li J, Khan AS, Khan ZH, Khoojine AS, Hu J, et al. Chapter 10 - Navigating the field: SLAM implementation for service robots in sports environment. In: Boubaker O, editor. Robotics and Artificial Intelligence in Sports Medicine and Sports Services. Medical Robots and Devices: New Developments and Advances. Academic Press; 2026. p. 241-75. Available from: https://www.sciencedirect.com/science/article/pii/B9780443217340000081.

[33] Robot simulator CoppeliaSim: create, compose, simulate, any robot - Coppelia Robotics;. Available from: https://www.coppeliarobotics.com/.

[34] Khan AT, Cao X, Li S. Dual beetle antennae search system for optimal planning and robust control of 5-link biped robots. Journal of Computational Science. 2022;60:101556.

Appendix

Table 9. Summary of Prior Research on BAS Applications in Robotic and Optimization Systems

Year	Authors	Application	BAS Integration	Environment	Key Outcomes
2020	[17]	Manipulator control	BAS + ZNN	KUKA IIWA (3D, MATLAB)	Unified obstacle avoidance and tracking; fast convergence using GJK-based distance evaluation.
2021	[18]	Path planning	BASL-RRT*	2D GIS cavity	Reduced runtime (29.53 s) and nodes (1,302) versus RRT* (133.15 s, 8,487 nodes).
2022	[20]	Optimization	Enhanced BAS (EBAS)	CEC'17 (10–100D)	Adaptive step size improved convergence; outperformed GWO and SMA on benchmark functions.
2022	[21]	Vehicle navigation	VBAS + B-Spline	2D MATLAB map	Generated shorter paths faster than APF in single and multi-obstacle scenarios.
2022	[34]	Control optimization	BAS vs PSO, GA	Simulation	Faster convergence in iterations; higher execution time due to single-agent search.
2023	[22]	USV navigation	Improved BAS	2D MATLAB	Reduced turning angle by 50% and shortened path length by 17 units.
2023	[23]	Path planning	WPFBAS	2D static map	Achieved shortest path (28.62) and lowest runtime (0.24 s), outperforming PSO and GA.
2024	[25]	Survey	BAS & hybrids	Multiple domains	Identified convergence limits; BAS-PSO hybrids showed superior real-world performance.
2025	[26]	Image optimization	BAS vs PSO	2D transformations	12.5% faster convergence and 8.3% lower error than PSO.
2025	[27]	Robot control	BAS + PSO FOPID	UR5 (MATLAB)	Lowest overshoot, fastest response, and minimal tracking error.