Erciyes Üniversitesi
Fen Bilimleri Enstitüsü Dergisi
Cilt 41, Sayı 2, 2025

Erciyes University
Journal of Institute of Science and Technology
Volume 41, Issue 2, 2025

The Effect of Path Linearity on Mobile Robot Path Planning and Tracking Control

Sertac Savas[1], Mustafa Yusuf Yildirim[*2], Rustu Akay[1]

[1] Erciyes University, Faculty of Engineering, Department of Mechatronics Engineering, Kayseri, Türkiye
[*2] Nigde Omer Halisdemir University, Faculty of Engineering, Department of Mechatronics Engineering, Niğde, Türkiye

(Alınış / Received: 25.04.2025, Kabul / Accepted: 03.06.2025, Online Yayınlanma / Published Online: 30.08.2025)

Keywords
Mobile Robot,
Path Planning,
Artificial Bee Colony,
Path Tracking Control,
Linearity

Abstract: In recent years, mobile robots have begun to be used in sectors such as social life, education and health as well as in the industrial sector. For this reason, mobile robots and their problems have become the focus of attention by researchers. Path planning and tracking control are among the basic problems of mobile robots. However, the literature generally focuses on algorithmic improvements, ignoring the disadvantages that the path type can cause such as long computation times and high costs. This study focuses on the problem simplifying instead of the algorithmic improvements, and examines the effect of path linearity on a robot's tracking the planned path. Therefore, a pure pursuit controller is designed for curve and line-type optimal paths planned by artificial bee colony algorithm and path tracking performances are compared. For path planning, the line-type paths outperformed the curve-type paths by 3–11% in terms of path length and 13–22% in terms of running time. For tracking control, although both path types demonstrated similar accuracy, the curve-type paths showed slightly better performance, particularly in sharp turns. The maximum tracking errors were approximately 0.0018 m in the X-axis, 0.012 m in the Y-axis and 0.046 rad in orientation.

Mobil Robot Yol Planlaması ve Takip Kontrolünde Yol Doğrusallığının Etkisi

Anahtar Kelimeler
Mobil Robot,
Yol Planlama,
Yapay Arı Koloni,
Yol Takip Kontrolü,
Doğrusallık

Öz: Son yıllarda mobil robotlar sosyal yaşam, eğitim ve sağlık gibi sektörlerin yanı sıra endüstriyel alanda da kullanılmaya başlanmıştır. Bu nedenle mobil robotlar ve problemleri araştırmacıların ilgi odağı haline gelmiştir. Yol planlama ve takip kontrolü mobil robotların temel problemleri arasındadır. Ancak literatür genellikle algoritmik iyileştirmelere odaklanmış, uzun hesaplama süreleri ve yüksek maliyetler gibi yol tipinin neden olabileceği dezavantajları göz ardı etmiştir. Bu çalışma algoritmik iyileştirmeler yerine problemin basitleştirilmesine odaklanmış ve yol doğrusallığının bir robotun planlanan yolu takip etmesi üzerindeki etkisini incelemiştir. Bu nedenle yapay arı kolonisi algoritması ile planlanan eğri ve çizgi tipi optimal yollar için bir saf takip denetleyicisi tasarlanmış ve yol takip performansları karşılaştırılmıştır. Yol planlaması için, çizgi tipi yollar yol uzunluğu açısından %3–11 ve çalışma süresi açısından %13–22 oranında eğri tipi yollardan daha iyi performans göstermiştir. Takip kontrolü için ise her iki yol türü benzer doğruluk göstermesine rağmen, eğri tipi yollar özellikle keskin dönüşlerde biraz daha iyi performans göstermiştir. Maksimum takip hataları X ekseninde yaklaşık 0.0018 m, Y ekseninde 0.012 m ve yönelimde 0.046 rad olarak ölçülmüştür.

*Corresponding author, email: myyildirim@erciyes.edu.tr

1. Introduction

In recent years, mobile robots have become increasingly important in social life as well as in industrial areas. In this context, practical applications such as robot vacuum cleaners in homes are an indication of the widespread use of mobile robots. However, the potential for use of mobile robots in areas such as education and health sectors is also quite large. Unlike fixed robotic arms, mobile robots stand out with their ability to move freely within a certain work area. In addition, these robots can have an autonomous structure, meaning they can perform their tasks without human intervention. As mechatronic systems, mobile robots have generally been the subject of research addressing four fundamental problems: mapping, localization, path planning, and path tracking. Mapping is about a robot recognizing its environment and creating a detailed map of the area. Localization means that the robot accurately estimates its position in the environment. Path planning is a problem that involves calculating the path that a robot should reach target points at optimum cost without collisions. Finally, path tracking is a problem that involves designing the control systems required to ensure that the robot moves accurately and efficiently along a planned path [1-2].

The rapid development and widespread use of mobile robot technology has brought with it some technical difficulties. Among these difficulties, the inability of robots to move as desired on slippery surfaces and their tendency to deviate from the determined path while traveling at high speeds stand out. Especially in industrial applications, such problems can prevent robots from operating efficiently. In order to overcome such technical difficulties, it is important to design and implement path tracking controllers. Path tracking controllers are used to regulate the movement of the robot, direct it to the desired path and provide the most efficient performance. These controllers monitor the robot's movement in real time, evaluate environmental conditions and ensure that it travels on the desired path by making the necessary corrections [3].

This study focuses on path planning and path tracking problems of mobile robots. Therefore, the literature is reviewed in terms of these two problems. Recent studies based on path planning can be summarized as follows: Lu et al. proposed an improved teaching learning based optimization algorithm. This method effectively optimized mobile robot path planning and outperformed the leading algorithms in terms of efficiency [4]. Psotka et al. proposed an improved wavefront algorithm for ground mobile robots. This algorithm removed unnecessary waypoints and smoothed the generated path using B-spline [5]. Ali et al. used firefly algorithm in mobile robot path planning. This algorithm created smooth paths by finding suitable control points in various environments [6]. Wang et al. proposed a method that includes genetic algorithm (GA) and grid-based methods. This method effectively optimized the path planning of mobile robots in complex environments [7]. Abed et al. proposed a hybrid approach of particle swarm optimization (PSO) and bat algorithm for multi-objective optimization of autonomous mobile robots for moving targets in dynamic environments [8]. Qin et al. used proximal policy optimization algorithm for safe path planning of mobile robots using local information from LiDAR in unknown environments [9]. Li et al. proposed a machine vision based mobile robot path planning system and a method based on ant colony optimization. This method planned safe and efficient paths with optimal parameter combination in mobile robot areas [10]. Yanhua et al. proposed an improved spider swarm algorithm. This algorithm effectively improved mobile robot path planning, enhanced search ability and avoided local optimization problems [11]. Alabdalbari et al. proposed a hybrid version of grey wolf optimization and PSO algorithm. This method effectively optimized the feasible paths in static environments and outperformed different methods such as PSO and artificial bee colony algorithm [12]. Yildirim et al. proposed a model in which PSO and k-means clustering algorithms are used as a hybrid for multi-obstacle environments and the problem size is reduced by clustering the obstacles [13]. Akay et al. stated that the sine cosine algorithm (SCA) could not produce satisfactory results due to a single update strategy for multi-robot path planning, and proposed a new algorithm based on SCA [14]. Galarza-Falfan et al. proposed an approach by thoroughly analyzing the integration of deep learning-based artificial vision systems into autonomous mobile robots. By comparing ResNet18 and YOLOv3 algorithms for real-time object detection, it introduced a method to improve the adaptability and efficiency of robots in dynamic environments [15]. Li et al. proposed an improved deep deterministic policy gradient algorithm (DDPG) to address the low success rate and slow training speed. This method adds dynamic delay update strategy and Ornstein-Uhlenbeck noise with prior experience repetition and transfer learning to improve the learning efficiency, thus improving the success rate and training speed in path planning [16].

Recent studies based on path tracking can be summarized as follows: Wang et al. proposed an improved pure pursuit path control method based on heading error rate, effectively reduced the tracking error in agricultural applications, and improved the convergence and the working quality of the robot [17]. Nawawi et al. proposed a PID controller optimized with PSO and a two-wheeled mobile robot controller based on pure pursuit algorithm.

This controller enabled autonomous movement to specific waypoints in various applications [18]. Anurag et al. compared model predictive controller, pure pursuit and linear quadratic regulator control schemes in robo-taxi maneuvers and concluded that the best controller is the model predictive controller [19]. Zhao et al. proposed an adaptive fuzzy neural network for the path tracking problem. This control method improved the tracking and trajectory starting point optimization [20]. Shi et al. proposed a tracking control method based on AVRX operating system, and this controller improved the robot's path tracking accuracy [21]. Mérida-Calvo et al. proposed an improved motor control system composed of PID controller, Smith predictor, anti-windup scheme and Coulomb friction compensator. This method significantly improved the tracking accuracy for low-cost mobile robots [22]. Zeng et al. proposed a hybrid version of interval type II fuzzy logic and sliding mode control. This controller improved the tracking and motion quality of mobile robots under unknown environment conditions [23]. A hybrid controller of backstepping and fractional-order PID proposed by Xu et al. enabled effective path tracking of wheeled differential-driven mobile robots. The authors also proposed an improved beetle swarm optimization algorithm to tune the parameters [24]. Fadlo et al. proposed a different backstepping controller for a wheeled mobile robot. This method effectively reduced the energy consumption by providing fast convergence and proved to be effective in various scenarios [25]. Zaman et al. proposed a controller called fuzzy reinforcement learning and showed that autonomous mobile robots effectively track the planned path [26]. The authors proposed a deep reinforcement learning (DRL)-based method for path following and formation control of unmanned surface vehicles (USVs), incorporating a novel random braking mechanism to prevent training from getting stuck in local optima. This enhanced control strategy improves the adaptability and robustness of the formation, enabling flexible and automatic adjustment during navigation [27].

According to the path planning-based studies, although there are some studies on curve-type paths, line-type paths and smoothing line-type paths, improvements have been made on the algorithm side in these studies. The path tracking-based studies have also aimed at improving a controller. However, it is ignored that the path type may cause some disadvantages such as long calculation times and high cost. In order to overcome these disadvantages, curve and line-type paths should be planned and evaluated separately. In [28], curve and line-type optimal paths were planned using GA and the performances of these path types were compared for the path planning problem. It was observed that line-type paths gave better results than curve-type paths in terms of both length and calculation time. However, the effect of path linearity on path tracking control was not investigated in [28] and no study has been conducted on this investigation in the recent literature. Understanding the effect of path geometry (path linearity) on tracking performance is essential for reducing computational cost and ensuring reliability in real-world mobile robot applications. This necessity forms the main motivation of this study. The contribution of this study is to focus on the problem improvements instead of the algorithmic improvements, and to examine the effect of path linearity on a robot's tracking the planned path. Therefore, a pure pursuit controller was designed for curve and line-type paths planned using artificial bee colony algorithm and the path tracking performances of these path types were compared.

The rest of this study is organized in three sections: Section 2 examines the method in which ABC algorithm, formulation of path planning and pure pursuit controller are mentioned. Section 3 presents simulation data and results. The last part of the study, Section 4, is the conclusion.

2. Method

2.1. Artificial bee colony algorithm

Artificial Bee Colony (ABC) algorithm, introduced in 2005, is a population-based optimization technique inspired by the foraging behavior of honey bees [29]. The algorithm operates as follows: it begins by randomly generating an initial population, as described in Equation 1.

$$\mathbb{X} = X_j^{min} + \left(X_j^{max} - X_j^{min}\right) \times R, \;\; R \sim U(0,1)^{N \times D}, \;\; i \in \{1,2,\dots,N\}, \;\; j \in \{1,2,\dots,D\} \tag{1}$$

Where $\mathbb{X}$ represents the population, X_{ij} denotes the parameter j of the solution i, $\left[X_j^{min}, X_j^{max}\right]$ are boundary values for the parameter j, N is the population size and D is the problem size. This initial population is subjected to employed bee, onlooker bee and scout bee stages. A random parameter is selected in the employed bee stage and this parameter of the solutions is updated using Equation 2.

$$V_{ij} = X_{ij} + \Phi_{ij}\left(X_{ij} - X_{kj}\right), \;\; k \neq i, \;\; k \in \{1,2,\dots,N\} \tag{2}$$

where V_{ij} represents the updated value of the parameter j of the solution i, X_{kj} denotes the parameter j of another randomly chosen solution from the population, and Φ_{ij} is a randomly generated number within the

interval [-1,1]. The modified population is then evaluated using the objective function, and the fitness value is computed as shown in Equation 3.

$$fit_i = \begin{cases} \dfrac{1}{1+f_{V_i}} & if\ f(i) \geq 0 \\ 1+\left|f_{V_i}\right| & otherwise \end{cases} \tag{3}$$

where fit_i represents the fitness of the solution i and f_{V_i} denotes its corresponding cost. Prior to moving on to the onlooker bee phase, the selection probability for each solution is determined using Equation 4.

$$\delta_i = \frac{fit_i}{\sum_{i=1}^{N} fit_i} \tag{4}$$

where δ_i is the selection probability of the solution i. In the onlooker bee phase, these selection probabilities guide the algorithm toward solutions with higher fitness, improving its exploitation capability. At this stage, a random number is generated for each solution, and if it is less than the solution's selection probability, the solution is updated using Equation 2. The revised solution is then evaluated through the objective function, and its fitness is recalculated using Equation 3. In the scout bee phase, solutions that haven't improved within a certain limit are replaced by new ones generated via Equation 1, highlighting the algorithm's exploration ability. The best-performing solution from the updated population is stored, and this cycle continues until the stopping condition is satisfied [30]. The core steps of the ABC algorithm are outlined in Algorithm 1.

Algorithm 1. The basic steps of ABC

1. Set control parameters of the algorithm
2. Generate population using Equation 1
3. Evaluate the population using the objective function
4. Store the optimum solution
5. **while** (unless the stopping criterion is satisfied)
6. Update the population using Equation 2
7. Evaluate the population using Equation 3
8. Calculate the selection probabilities using Equation 4
9. Update the population using Equation 2 according to the selection probabilities
10. Evaluate the population using Equation 3
11. Generate a solution using Equation 1 instead of the solution that are not updated up to a limit
12. Update the optimum solution
13. **end while**
14. Report the optimum solution

2.2. Formulation of path planning

For modeling the path planning, different types of maps are created. In the case of continuous space maps, the process involves specifying the robot's start and target points, as well as identifying any obstacles in the map. The map data and start-target points are defined as shown in Equation 5 and 6, respectively.

$$E = \{p \in \mathbb{R}^2 \mid \exists \mathcal{F} \in C([0\ 1], \mathbb{R}^2)\} \tag{5}$$

$$\{p_s, p_t\} \in E \backslash O, \quad p_s \neq p_t \tag{6}$$

where E is the map data, $\mathcal{F}$ is a continuous function, p_s, p_t are the start and target points, O is the obstacle cluster. In this study, the obstacles are designed as circles, and defined as in Equation 7.

$$O = \left\{\left\{p_j^o, r_j\right\} \mid p_j^o \in E, j \in \{1,2,\dots,n_o\}, r_j \in \mathbb{R}^+, n_o \in \mathbb{N}^+\right\} \tag{7}$$

where p_j^o is the center of the obstacle j, r_j is the radius of the obstacle j and n_o is the number of the obstacles. In the continuous space maps, the paths are planned by interpolation method. The solutions produced by ABC algorithm represent data points to be used for the interpolation. The number of these points determines dimension of the problem (D). The data points are defined as shown in Equation 8 and 9.

$$P_d = \{p_s, p_t\} \cup \{p_m^d \mid p_m^d \in E, m \in \{1,2,\dots,D\}, D \in \mathbb{N}^+\} \tag{8}$$

$$P_d = \left\{ p_k^d \mid p_k^d \in E, k \in \{1,2,\dots,(D+2)\} \right\} \tag{9}$$

where p_i^d is the data point i and D is the number of the data points. The data points represent the dimension of the problem. Since the start and target points are included, the dimension of P_d is $(D\ +\ 2)$. The data points are subjected to the interpolation between each two consecutive data points, and multiple waypoints are generated that define the path as in Equation 10 and become ready to evaluate.

$$P_y = \left\{ p_i^y \mid p_i^y \in E, i \in \{1,2,\dots,n_y\}, n_y \in \mathbb{N}^+, n_y \geq (D+2) \right\},\ \ P_y \supseteq P_d \tag{10}$$

where p_i^y is the point of the waypoint j in the path, n_y is the number of the waypoints. The interpolated path is sampled at n_y waypoints, which define the planned path and are used in the objective function for length evaluation and collision avoidance. The paths are evaluated using the objective function in Equation 11 [31].

$$\arg\min_{P_y} F = L(1+\beta V) \tag{11}$$

where F is the objective function, L is path length cost, V is obstacle violation cost and β is obstacle violation coefficient. L and V are calculated as shown in Equation 12 and 13, respectively. Thus, the objective function returns the planned path and overall cost of the path to ABC algorithm. $\|\cdot\|$ is the calculation of the Euclidean distance.

$$L = \sum_{i=1}^{n_y-1} \left\| p_i^y - p_{i+1}^y \right\| \tag{12}$$

$$V = \sum_{i=1}^{n_y} \sum_{j=1}^{n_o} \begin{cases} 1 - \dfrac{\left\| p_i^y - p_j^o \right\|}{r_j}, & if\ \left(1 - \dfrac{\left\| p_i^y - p_j^o \right\|}{r_j}\right) > 0 \\ 0, & otherwise \end{cases} \tag{13}$$

2.2.1. Curve-type paths

p_k^d is interpreted as a two-dimensional coordinate point, representing a position in the Cartesian plane. This interpretation is illustrated in detail in Equation 14, where its components are explicitly defined and used to describe spatial relationships within the modeled system.

$$p_k^d = \begin{bmatrix} x_k^d \\ y_k^d \end{bmatrix},\ \ k \in \{1,2,\dots,(D+2)\} \tag{14}$$

For curve-type paths, cubic spline interpolation is used in this paper. This interpolation uses a segmented polynomial function that creates a smooth curve between the data points. In this method, curves are created using third-degree polynomials for each segment and these curves are connected at the data points. For each segment $[x_k^d\ , x_{k+1}^d\ ]$, the cubic polynomial is defined as in Equation 15.

$$S_k(x) = a_k + b_k\left(x - x_k^d\right) + c_k\left(x - x_k^d\right)^2 + d_k\left(x - x_k^d\right)^3 \tag{15}$$

$a_k\ ,\ b_k\ ,\ c_k$ and d_k are coefficients. These coefficients are determined and each polynomial accurately represents the data points. Cubic spline interpolation satisfies the following conditions:

- For each segment, $S_k(x_k) = y_k$ and $S_k(x_{k+1}) = y_{k+1}$.
- The polynomials are continuous at the point x_k: $S_k(x_k) = S_{k+1}(x_k)$
- The first derivatives are continuous at the same point: $S_k'(x_k) = S_{k+1}'(x_k)$
- The second derivatives are continuous at the same point: $S_k''(x_k) = S_{k+1}''(x_k)$

These conditions create a system of linear equations for determining the coefficients and by solving each polynomial. In this way, cubic spline interpolation provides a smooth and continuous curve between the data points.

2.2.2. Line-type paths

For line-type paths, linear interpolation is used in this paper. This interpolation is a simple method used to estimate an unknown point by establishing a linear relationship between two data points. Mathematically, the goal of linear interpolation is to determine y corresponding to any x in between using two consecutive data points (x_k^d, y_k^d) and (x_{k+1}^d, y_{k+1}^d). This is accomplished by defining a linear function for x and is calculated as in Equation 16.

$$S_k(x) = y_k^d + \left(x - x_k^d\right)\frac{\left(y_{k+1}^d - y_k^d\right)}{\left(x_{k+1}^d - x_k^d\right)} \tag{16}$$

Thus, a linear function is defined for the segment $[x_k^d, x_{k+1}^d]$. Linear interpolation provides a linear transition between data points for each segment. In this way, lines between the data points is created.

2.3. Pure pursuit controller

Pure pursuit controller is a path-tracking algorithm that attempts to geometrically calculate the curvature that will move a mobile robot from its current position to a target [32]. The algorithm is modeled after people looking slightly ahead of their cars and moving toward that point while driving. Therefore, the robot is thought to be tracking a point a little further down the path. This target point is a point on the path that is one lookahead distance away from the current position of the robot. An arc is created connecting the current position and the target position. The geometry of the algorithm is shown in Figure 1.

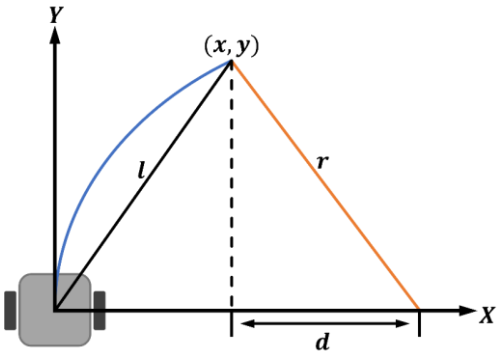


Figure 1. Geometry of pure pursuit controller

The point (x, y) in Figure 1 is the position that is lookahead distance (l) away from the origin position of the mobile robot. Therefore, Equation 17 can be written for the triangle formed on the left, and Equation 18 can be written for the triangle on the right. r in Equation 18 is the radius of the arc, and d is the difference between r and the projection of the target position on $\boldsymbol{X}$ axis, as seen in Equation 19.

$$x^2 + y^2 = l^2 \tag{17}$$

$$d^2 + y^2 = r^2 \tag{18}$$

$$d = r - x \tag{19}$$

If the expression for d in Equation 19 is written and rearranged in Equation 18, the expression for $(1/r)$, will be obtained as in Equation 20.

$$\gamma = \frac{1}{r} = \frac{2x}{l^2} \tag{20}$$

The choice of lookahead distance in path tracking is also very important. A large lookahead distance will show a slow reflex in tracking the path, while small lookahead distances will cause an aggressive oscillate to track the path. This effect is shown in Figure 2.

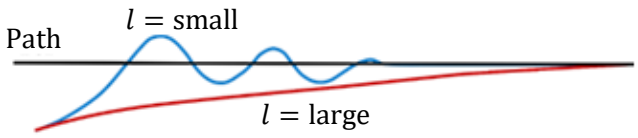


Figure 2. Effect of lookahead distance on the path

Although the pure pursuit controller is a useful path-tracking algorithm, it has two important limitations. The first of these is that the points between the waypoints cannot be directly tracked. The other limitation is that the mobile robot cannot be stabilized at any point. For this reason, in the control loop, the distance between the robot's current position and the target position must be constantly checked and the control loop must be terminated when it falls below the desired value. In addition, the linear speed of the mobile robot is fixed, and the controller produces angular velocity as a control signal. Figure 3 presents the closed-loop control block diagram of the system, where the Pure Pursuit controller is used to generate steering commands based on the tracking error. This figure highlights how the reference path and robot states interact through the control loop.

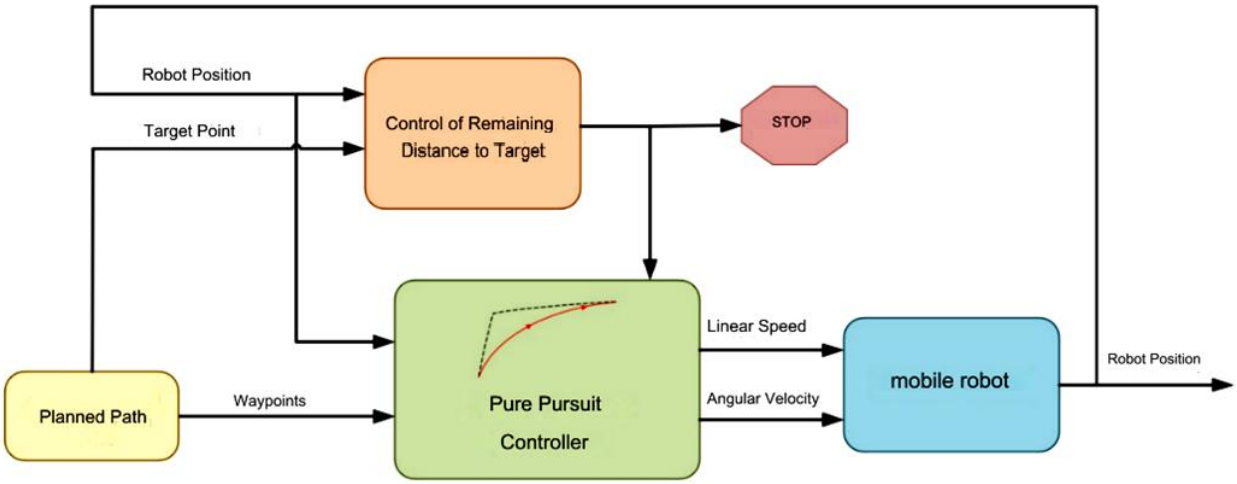


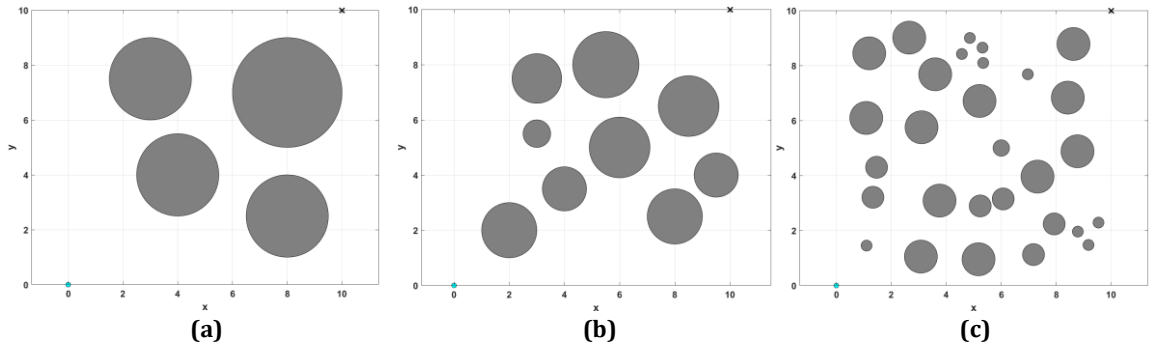
Figure 3. Pure pursuit closed-loop control block diagram

3. Results

All simulations in this paper were coded in the MATLAB programming language and run on a computer with 16 GB RAM and 2.6 GHz processor. Firstly, the effects of curve and line-type paths on path planning were briefly analyzed. Secondly, the effects of these path types on path tracking were analyzed and interpreted comprehensively, using the paths in the path planning analysis.

3.1. Effect of path types in path planning

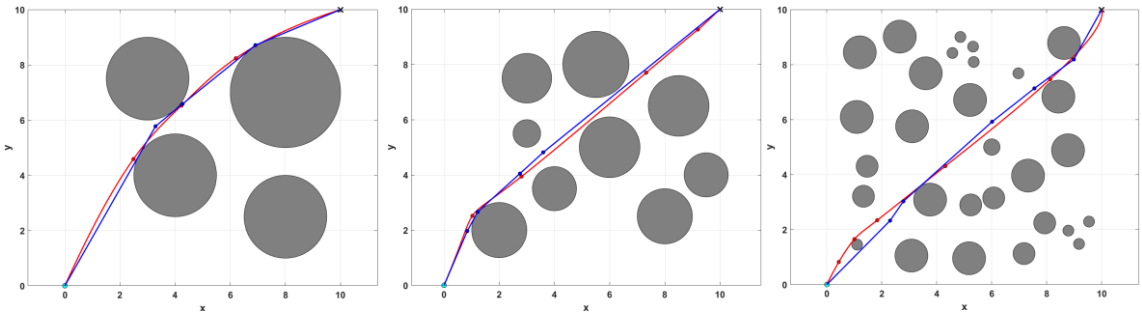
To analyze the effect of path type, different maps in meter were designed and curve and line-type paths were planned with ABC algorithm. In this paper, three maps were designed with different levels of difficulty and shown in Figure 4. In all maps, the coordinate of the start point is (0,0) and the coordinate of the target point is (10,10). Obstacles were randomly distributed in the maps. Number of obstacles is 4 in Map 1, 9 in Map 2, and 29 in Map 3.



(a) **(b)** **(c)**

Figure 4. The maps designed with different levels of difficulty: **(a)** Map 1, **(b)** Map 2, **(c)** Map 3 (The turquoise circle, the black cross and the gray circles represent the start point, the target point and the obstacles, respectively.)

For path planning, β and n_y were set to 100. Number of data points (D) is 3 for Map 1, 4 for Map 2, and 5 for Map 3. For ABC algorithm, maximum number of fitness evaluation and population size were set to 5000 and 50, respectively. The algorithm was run 30 times for both maps. Planned curve and line-type paths are shown in Figure 5, path length and running time comparisons are shown in Table 1. The planned paths are the solutions with the minimum fitness among 30 runs.



(a) **(b)** **(c)**

Figure 5. Planned curve and line-type paths: **(a)** Map 1, **(b)** Map 2, **(c)** Map 3 (The turquoise circle, the black cross and the gray circles represent the start point, the target point and the obstacles, respectively. The red line, the blue line, the red dots and the blue dots represent the curve-type path, the line-type path, the data points of the curve-type path, and the data points of the line-type path, respectively)

Table 1. Path length and running time comparisons of the path types (These results represent the mean of 30 runs.)

Path Type	Map 1		Map 2		Map 3	
	Path Length (m)	Running Time (s)	Path Length (m)	Running Time (s)	Path Length (m)	Running Time (s)
Curve-Type	15,3579	6,2743	15.3686	6.9937	16,7069	14,5349
Line-Type	14,8176	4,8335	14.5158	5.5418	14,8417	12,5704

Considering Table 1, line-type paths produced better results than curve-type paths in terms of path lengths for both maps. The line-type paths outperformed curve-type paths by 3.52% for Map 1, 5.55% for Map 2, and 11.15% for Map 3. It can be said that as maps becomes more complex, the advantage of the line-type paths becomes more obvious. Moreover, it is seen that the line-type paths were planned faster than curve-type paths for all maps. The line-type paths outperformed curve-type paths by 22.97% for Map 1, 20.76% for Map 2, and 13.5% for Map 3. This rate of Map 3 is lower than Map 1-2, and this is due to the difficulty of the problem. As a result, it can be said that the line-type paths showed better performance in all maps.

3.2. Effect of path types in path tracking

In this simulation, the tracking performance of a mobile robot on the planned paths in Subsection 3.1 was analyzed. A pure pursuit controller is designed and tracking performances of these two types of paths are compared. The parameters of the pure pursuit controller are given in Table 2.

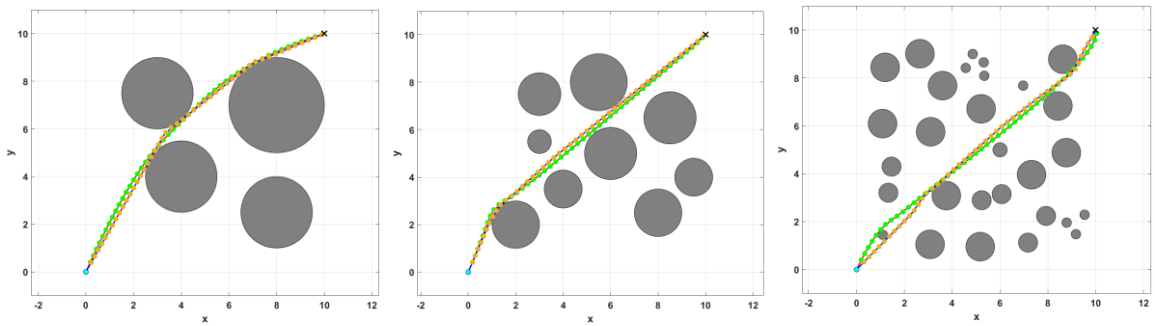
Table 2. Pure pursuit controller parameters

Controller Parameters	Parameter Values
Reference Linear Velocity	1.5 m/s
Maximum Angular Velocity	2 rad/s
Lookahead Distance	0.3 m
Target Point Radius	0.1 m
Sampling Time	0.1 s

Tracking time and distance left comparisons are shown in Table 3, and the path tracking performance of the mobile robot is shown in Figure 6. Distance left measures how close the robot is to the target point at the end of the simulation.

Table 3. Tracking time and distance left comparisons of the path types

Path Type	Map 1		Map 2		Map 3	
	Tracking Time (s)	Distance Left (m)	Tracking Time (s)	Distance Left (m)	Tracking Time (s)	Distance Left (m)
Curve-Type	9,7	0,0433	9,5	0,0306	9,5	0,0580
Line-Type	9,7	0,0323	9,5	0,0166	9,4	0,0425



(a) **(b)** **(c)**

Figure 6. Path tracking performance of the mobile robot: **(a)** Map 1, **(b)** Map 2, **(c)** Map 3 (The turquoise circle, the black cross and the gray circles represent the start point, the target point and the obstacles, respectively. The red line and the blue line represent the curve-type and the line-type paths, respectively. The green and yellow circles represent the tracking points of the curve-type and line-type paths, respectively.)

Considering the distance left, there is a difference of 0.011 m in Map 1, 0.014 m in Map 2, and 0.0155 m in Map 3. As can be seen, there is a very small difference for all maps. However, although the tracking times are the same for Map 1-2, it is seen that it is slightly less for the line-type path in Map 3. Nevertheless, it cannot be said that there is a significant difference. It can be interpreted that these two types of paths give successful results for all maps. It will be more explanatory to examine this tracking performance on the path components. x, y and θ components of the desired path and actual path tracked by the mobile robot are given in Figure 7 for Map 1, Figure 9 for Map 2, and Figure 11 for Map 3. The errors in X and Y axes and the orientation angle error of the robot for these paths are given in Figure 8 for Map 1, Figure 10 for Map 2, and Figure 12 for Map 3. Moreover, RMSE of these tracking errors are given in Figure 13.

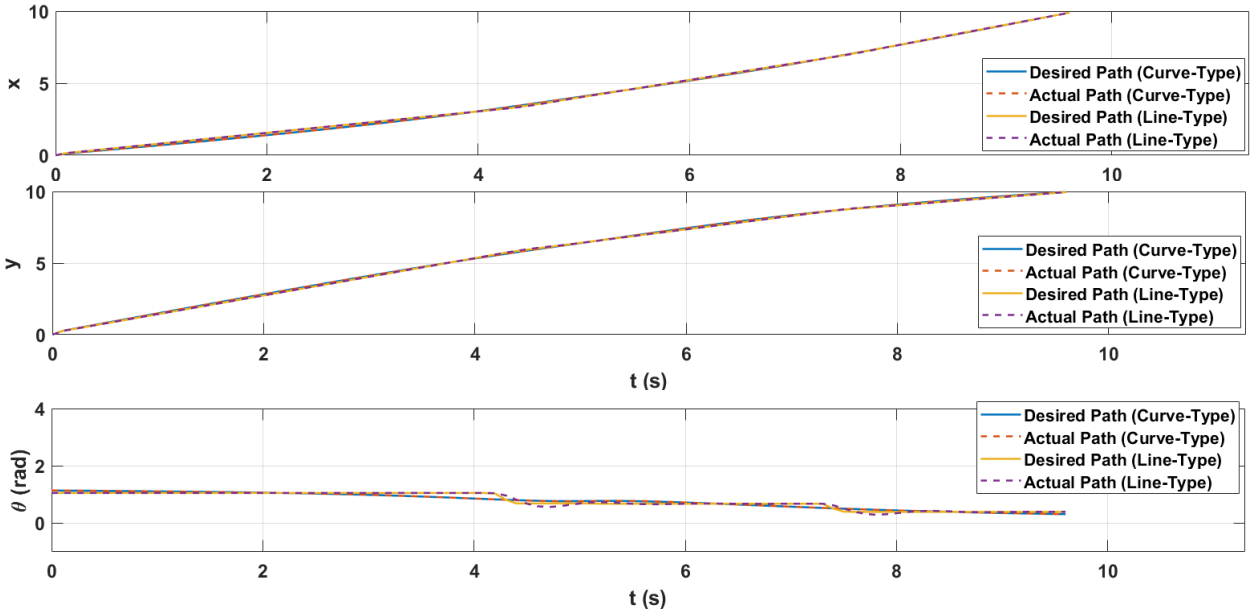


Figure 7. Desired and actual path components (x, y, θ) in Map 1

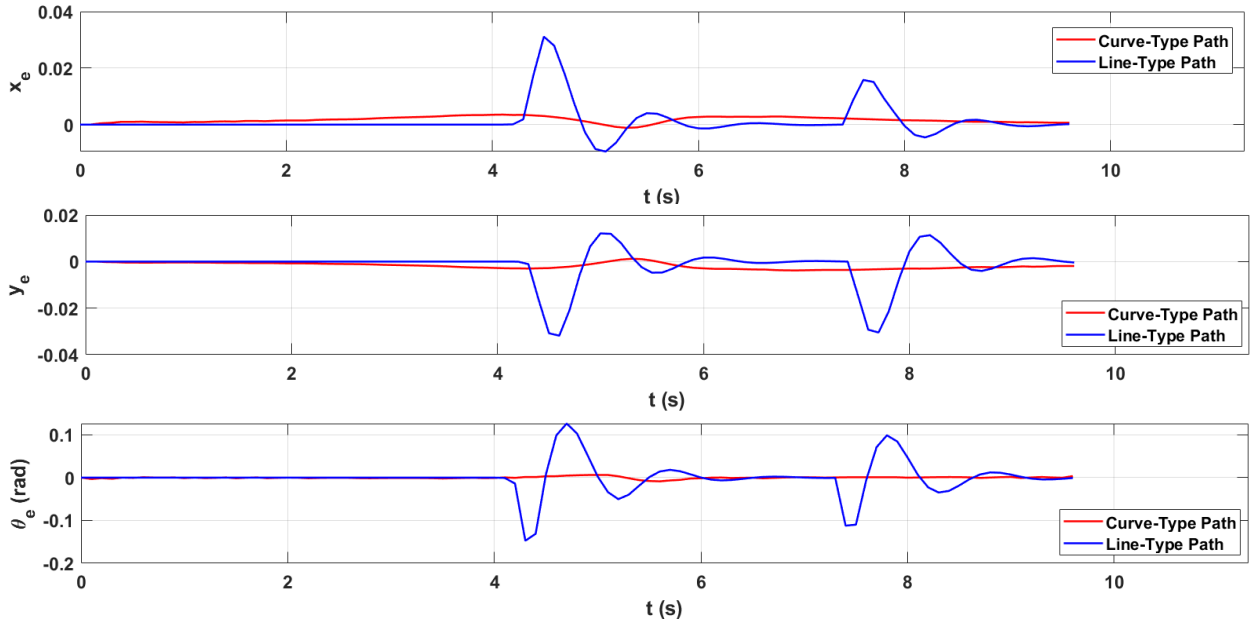


Figure 8. Tracking error components (x_e, y_e, θ_e) in Map 1

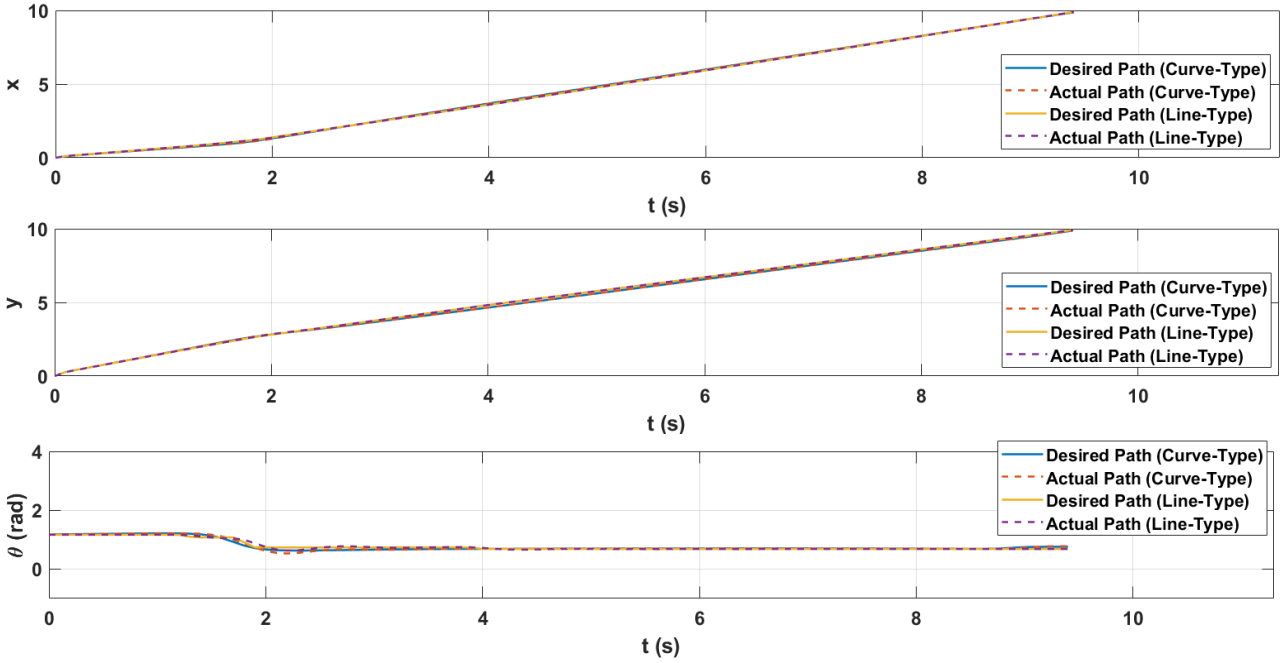


Figure 9. Desired and actual path components (x, y, θ) in Map 2

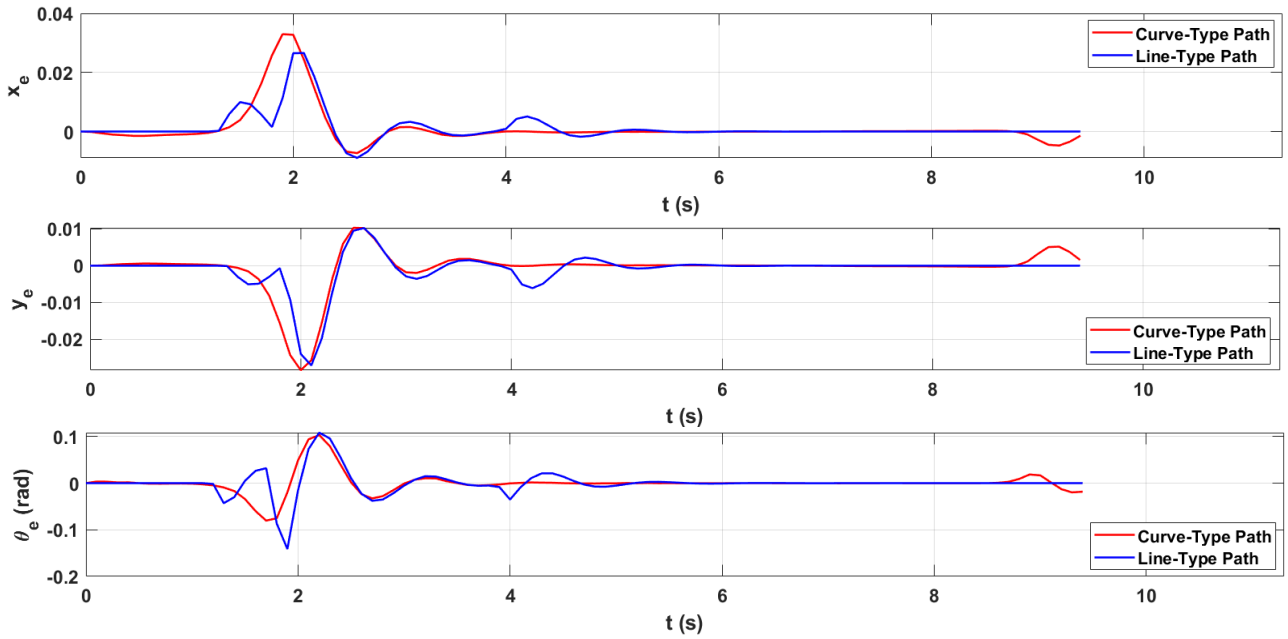


Figure 10. Tracking error components (x_e, y_e, θ_e) in Map 2

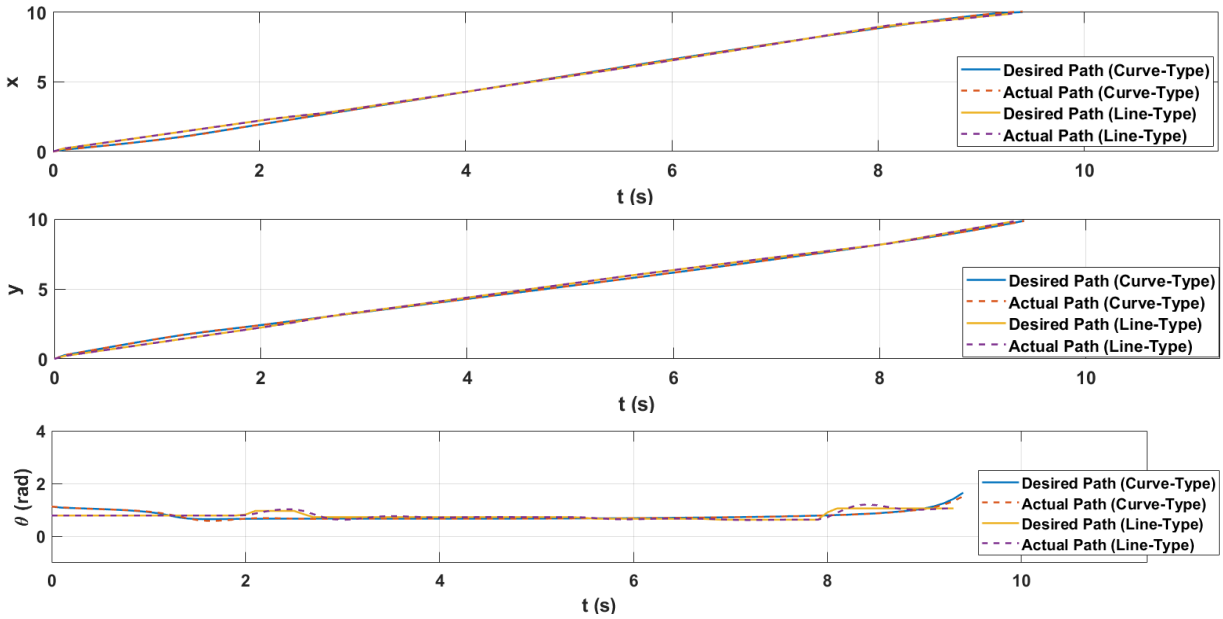


Figure 11. Desired and actual path components (x, y, θ) in Map 3

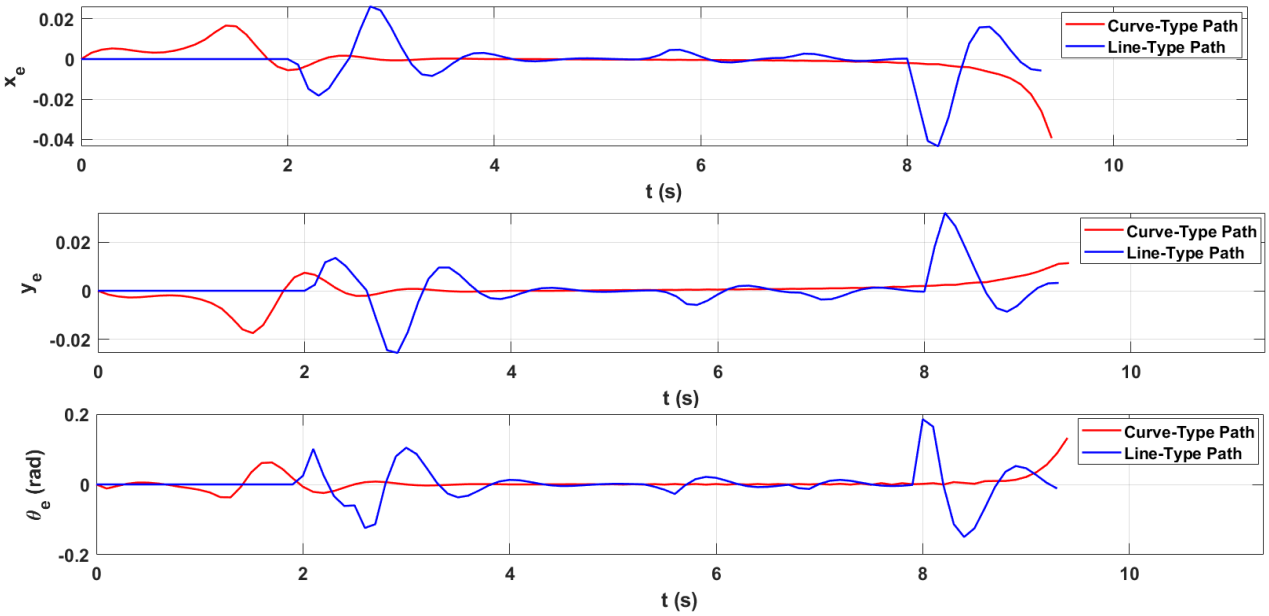


Figure 12. Tracking error components (x_e, y_e, θ_e) in Map 3

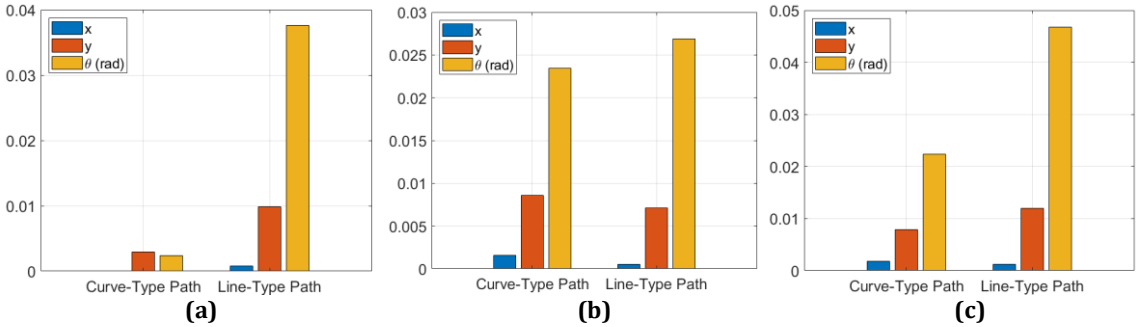


Figure 13. RMSE tracking errors: **(a)** Map 1, **(b)** Map 2, **(c)** Map 3

Considering Figure 7-12, tracking of the curve-type paths is more successful than the line-type paths for both maps. This tracking performance is especially evident in turns. In the line-type paths, which has sharper turns, the orientation angle of the robot is delayed in reaching the target orientation, and even some oscillation occurs in the orientation. This situation causes serious deviations from the path in both X and Y axes during turns. This is also seen in Figure 13, where RMSE errors are given. However, it can be said that pure pursuit controller is a high-performance method and exhibits good tracking performance especially in the curve-type paths. In addition, considering the RMSE errors, the fact that the maximum error in X-axis is approximately 0.0018 m, the maximum error in Y-axis is approximately 0.012 m, and the maximum orientation error is approximately 0.046 rad shows that there is no significant difference in tracking performance between the curve and line-type paths.

It is seen that the line-type paths are advantageous in the metrics of path length and running time for path planning, tracking time and distance left for path tracking. Considering the tracking errors, it can be said that although the line-type paths produce partially worse results, there is no significant difference between these two types of paths.

4. Discussion and Conclusion

In this study, the effects of path linearity on both path planning and path tracking performance of mobile robots were investigated. Two different maps with various numbers of static obstacles were designed, and optimal curve and line-type paths were planned using ABC algorithm. The results showed that in both maps, line-type paths achieved shorter path lengths and lower running times compared to curve-type paths by 3–11% in path length and 13–22% in running time. This highlights the computational efficiency of line-type paths, especially in relatively complex environments.

For path tracking, a pure pursuit controller was applied to both path types. While the overall tracking accuracy was similar, the curve-type paths yielded slightly better results, particularly in sharp turns. The maximum tracking errors remained low (approximately 0.0018 m in X-axis, 0.012 m in Y-axis, and 0.046 rad in

orientation), indicating that both path types are feasible in terms of control performance. However, due to the simpler structure of the test maps, the differences in tracking errors were not statistically significant.

Compared to existing approaches in the literature that focus primarily on algorithmic improvements, this study presents a unique perspective by emphasizing problem-level improvements; specifically, analyzing the impact of path geometry. This approach offers a practical and low-cost alternative for improving overall performance without increasing algorithmic complexity.

In summary, line-type paths were more advantageous in terms of path length, running time, and tracking efficiency, while curve-type paths provided smoother control responses in turning maneuvers. This trade-off should be considered when designing navigation strategies for mobile robots in real-world applications.

For future work, both comparative analyses involving various types of controllers and more comprehensive, detailed studies under dynamic and realistic scenarios are planned to thoroughly evaluate the performance and robustness of the proposed approach.

References

[1] Cebollada, S., Payá, L., Flores, M., Peidró, A., Reinoso, O. 2021. A State-of-the-Art Review on Mobile Robotics Tasks Using Artificial Intelligence and Visual Data. Expert Systems with Applications, 167, 114195.

[2] Tzafestas, S. G. 2018. Mobile Robot Control and Navigation: A Global Overview. Journal of Intelligent & Robotic Systems, 91, 35-58.

[3] Hadi N. H., Younus, K. K. 2020. Path Tracking and Backstepping Control for A Wheeled Mobile Robot (WMR) in A Slipping Environment. IOP Conference Series: Materials Science and Engineering 3rd International Conference on Engineering Sciences, November 4-6, Kerbala, Iraq, 012005.

[4] Lu, S., Liu, D., Li, D., Shao, X. 2023. Enhanced Teaching-Learning-Based Optimization Algorithm for the Mobile Robot Path Planning Problem, Applied Sciences, 13(4), 2291.

[5] Psotka, M., Duchoň, F., Roman, M., Michal, T., Michal, D. 2023. Global Path Planning Method Based on A Modification of the Wavefront Algorithm for Ground Mobile Robots, Robotics, 12(1), 25.

[6] Ali, S., Yonan, J., Alniemi, O., Ahmed, A. 2022. Mobile Robot Path Planning Optimization Based on Integration of Firefly Algorithm and Cubic Polynomial Equation, Journal of ICT Research and Applications, 16(1), 1-22.

[7] Wang, W., Ng, C., Chen, R. 2022. Vision-Aided Path Planning Using Low-Cost Gene Encoding for A Mobile Robot, Intelligent Automation & Soft Computing, 32(2), 991-1006.

[8] Abed, B., Jasim, W. 2022. Hybrid Approach for Multi-Objective Optimization Path Planning with Moving Target, Indonesian Journal of Electrical Engineering and Computer Science, 29(1), 348-357.

[9] Qiao, B., Wu, W., Deng, Y. 2022. A Path Planning Algorithm Based on Deep Reinforcement Learning for Mobile Robots in Unknown Environment, IEEE 5th Advanced Information Management, Communicates, Electronic and Automation Control Conference, December 16-18, Chongqing, China, 1661-1666.

[10] Li, J., Cai, L. 2022. Research on Robot Path Planning Based on Machine Vision, International Conference on Electronic Information Technology, March 18-20, Chengdu, China, 671-676.

[11] Yanhua, C., Fenggang, L., Bei, Y. 2022. Research on Path Planning Method of Mobile Robot Based on Improved Spider Swarm Algorithm, 5th International Conference on Machine Vision and Applications, February 18-20, Singapore, 123-128.

[12] Alabdalbari, A., Abed, I. 2022. New Robot Path Planning Optimization Using Hybrid GWO-PSO Algorithm, Bulletin of Electrical Engineering and Informatics, 11(3), 1289-1296.

[13] Yildirim, M. Y., Akay, R. 2021. Fast Path Planning in Multi-Obstacle Environments for Mobile Robots, Journal of the Faculty of Engineering and Architecture of Gazi University, 36(3), 1551-1564.

[14] Akay, R., Yildirim, M. Y. 2023. Multi-strategy and Self-Adaptive Differential Sine-Cosine Algorithm for Multi-Robot Path Planning, Expert Systems with Applications, 232, 120849.

[15] Galarza-Falfan, J., García-Guerrero, E. E., Aguirre-Castro, O. A., López-Bonilla, O. R., Tamayo-Pérez, U. J., Cárdenas-Valdez, J. R., Hernández-Mejía, C., Borrego-Dominguez, S., Inzunza-Gonzalez, E. 2024. Path Planning for Autonomous Mobile Robot Using Intelligent Algorithms. Technologies, 12(6), 82.

[16] Li, P., Chen, D., Wang, Y., Zhang, L., Zhao, S. 2024. Path Planning of Mobile Robot Based on Improved TD3 Algorithm in Dynamic Environment. Heliyon, 10(11), e32167.

[17] Wang, L., Chen, Z., Zhu, W. 2022. An Improved Pure Pursuit Path Tracking Control Method Based on Heading Error Rate. Ind. Robot, 49, 973-980.

[18] Nawawi, S., Abdeltawab, A., Samsuria, N., Sirkunan, N. 2022. Modelling, Simulation and Navigation of a Two-Wheel Mobile Robot Using Pure Pursuit Controller, ELEKTRIKA-Journal of Electrical Engineering, 21(3), 69-75.

[19] Anurag, R., Jisha, V. 2022. Performance Analysis of Path Tracking Controllers for Robotaxi Manoeuvres, IEEE International Conference on Signal Processing, Informatics, Communication and Energy Systems, March 10-12, Thiruvananthapuram, India, 152-157.

[20] Zhao, T., Qin, P., Zhong, Y. 2023. Trajectory Tracking Control Method for Omnidirectional Mobile Robot Based on Self-Organizing Fuzzy Neural Network and Preview Strategy, Entropy, 25(2), 248.

[21] Shi, Q. 2022. Trajectory Tracking Control Method of Mobile Robot Based on AVRX Operating System, Journal of Computational Methods in Sciences and Engineering, 23(1), 253-265.

[22] Mérida-Calvo, L., Rodríguez, A., Ramos, F., Feliu-Batlle, V. 2022. Advanced Motor Control for Improving the Trajectory Tracking Accuracy of a Low-Cost Mobile Robot, Machines, 11(1), 14.

[23] Zeng, H., Di, Y. 2022. Trajectory Tracking Control of Mobile Robot Based on Interval Type II Fuzzy Sliding Mode, IEEE 2nd International Conference on Electronic Technology, Communication and Information, May 27-29, Changchun, China, 1420-1425.

[24] Xu, L., Du, J., Song, B., Cao, M. 2022. A Combined Backstepping and Fractional-Order PID Controller to Trajectory Tracking of Mobile Robots, Systems Science & Control Engineering, 10, 134-141.

[25] Fadlo, S., Elmahjoub, A., Rabbah, N. 2022. Optimal Trajectory Tracking Control for A Wheeled Mobile Robot Using Backstepping Technique, International Journal of Electrical and Computer Engineering, 12(6), 5979-5987.

[26] Zaman, M., Wu, H. 2022. Fuzzy Reinforcement Learning Based Trajectory-Tracking Control of an Autonomous Mobile Robot, 22nd International Conference on Control, Automation and Systems, November 27-December 1, Jeju, Korea, 840-845.

[27] Zhao, Y., Ma, Y., Hu, S. 2021. USV Formation and Path-Following Control via Deep Reinforcement Learning with Random Braking. IEEE Transactions on Neural Networks and Learning Systems, 32(12), 5468-5478.

[28] Yildirim, M. Y., Akay, R. 2021. Investigation of linearity in path planning of mobile robot, European Journal of Science and Technology, 24, 138-142.

[29] Karaboga, D., Akay, B., Karaboga, N. 2020. A Survey on The Studies Employing Machine Learning (ML) for Enhancing Artificial Bee Colony (ABC) Optimization Algorithm, Cogent Engineering, 7(1), 1855741.

[30] Saleh, R. A., Akay, R. 2021. Artificial Bee Colony Algorithm with Directed Scout", Soft Computing, 25(21), 13567-13593.

[31] Yildirim, M. Y., Akay, R. 2021. A Comparative Study of Optimization Algorithms for Global Path Planning of Mobile Robots, Sakarya University Journal of Science, 25(2), 417-428.

[32] Coulter, R. C. 1992. Implementation of the Pure Pursuit Path Tracking Algorithm, Carnegie Mellon University, The Robotics Institute, 92-01.