Journal of the Brazilian Society of Mechanical Sciences and Engineering (2024) 46:612
https://doi.org/10.1007/s40430-024-05165-6

TECHNICAL PAPER

Urban mobile robot routing using fast search random tree method (RRT) in obstacle environments

Hsin-Yin Hsieh[1] · Kuan-Hung Chen[2] · Chich-Jen Shieh[3] · Shavan Askar[4] · Mostafa Jalalnezhad[5]

Received: 24 November 2023 / Accepted: 20 August 2024

Abstract
In this paper, the objective of path planning and control systems for robot is multifactorial. The method used for navigation algorithms based on is closed-loop random trees with quick search or CL-RRT. In this algorithm, each robot will grow a tree from its current location to the target or zone target is developed. The main advantage of this method is the ability to function in complex environments. For use this method, it is necessary that the CL-RRT algorithm for robot navigation online, be extended. In online mode, the robot does not have any information about the environment and through its sensors, which detect the environment and the ability to function in environments with obstacles dynamic and could face a new obstacle or the possibility of an obstacle's dynamic (the other a robot) to change its direction during motion. Performance of the design path used with design the controller for robot in an environment with various obstacles is evaluated, and then this design is for a group of robots used. To coordinate among agents and to ensure that there is no conflict between them, there is strategy based on priority assignment and LOS method. The strategy used in a way that ensures that the collision between agents does not happen and benefits of the design methods path used to keep. As a result of this strategy can be divided into two parts, the first part of the prioritization robots and the second part of the strategy, no conflict robot is in motion. In this paper, the problem of forming a group of robots has been investigated. In other words, the agents move in the number of agents must be a specific form. The agents according to a number try to set form (originally defined) to its motion to follow, also when faced with obstacles or other agents of its priorities is no conflict with obstacles or other factors and this moment it is possible to change the shape of their, if this will happen again the situation changes be considered form to make its motion.

Keywords CL-RRT algorithm · Path planning · Multi-agent systems · Formation · Obstacle environments

1 Introduction

In recent years, the use of urban mobile robots for various applications has increased significantly, leading to an increased demand for efficient routing algorithms in complex and dynamic environments [1–3]. The fast search random tree method (RRT) has emerged as a popular approach for path planning and navigation of mobile robots in urban settings with obstacles [4–7]. This method has shown promise in effectively finding feasible paths for mobile robots to navigate through cluttered environments while considering non-holonomic constraints and dynamic obstacles [8]. With the growing interest in autonomous vehicles and robotic systems for urban transportation and logistics, the development and implementation of RRT-based algorithms for mobile robot routing in obstacle environments have become a focal point of research and development efforts [9, 10]. This study aims to investigate the utilization of the RRT method for urban mobile robot routing and to propose enhancements to improve its efficiency, adaptability and reliability in real-world urban scenarios [11]. By addressing the challenges associated with urban mobile robot routing in obstacle environments [12], this research contributes to the advancement of autonomous navigation systems and paves the way for the practical deployment of mobile robots in urban settings [13]. The RRT method offers a promising solution for mobile robot routing in urban environments due to its ability to navigate around obstacles and dynamic constraints [14]. As urban settings become more crowded, the RRT method offers a probabilistically complete framework for rapidly exploring potential paths in complex urban environments [15]. By utilizing random sampling and incrementally

Technical Editor: Rogério Sales Gonçalves.

Extended author information available on the last page of the article growing a tree of feasible paths, the RRT method allows for efficient path planning in cluttered urban environments, improving the mobility and navigation of mobile robots in dynamic settings [16]. The rapid generation of feasible paths through random sampling enables the RRT method to quickly adapt to changes in the environment, making it suitable for urban mobile robot routing in dynamic obstacle environments [17]. This adaptability is crucial for ensuring safe and efficient navigation of mobile robots in urban environments.

In addition, the RRT method allows for real-time re-planning to avoid dynamic obstacles, making it well-suited for urban mobile robot routing in dynamic obstacle environments [18]. This capability to quickly adapt to changing conditions makes the RRT method highly versatile for navigating through crowded urban spaces [19]. Additionally, the probabilistic completeness of the RRT method ensures that it can efficiently find a feasible path for the mobile robot in complex urban environments [20]. This reliability is essential for the safe and effective navigation of mobile robots in urban environments, where unpredictable obstacles and dynamic constraints are common. By incorporating the RRT method into urban mobile robot routing [21], this research aims to improve the overall efficiency and safety of autonomous navigation in urban environments [22]. With its ability to quickly adapt to changing conditions and find feasible paths, the RRT method offers a promising solution for enhancing the mobility and navigation of mobile robots in crowded urban settings. This could lead to improved efficiency and safety in autonomous navigation. Furthermore, the RRT method's ability to efficiently explore the search space and find feasible paths makes it suitable for navigating through complex urban environments with dynamic obstacles [23, 24]. This adaptability is crucial for ensuring safe and efficient navigation of mobile robots in urban environments, particularly in the presence of dynamic obstacles and changing conditions. This adaptability is a key advantage of the RRT method for urban mobile robot routing [25]. Its ability to quickly adapt to changing conditions and find feasible paths makes it a valuable tool for enhancing autonomous navigation in crowded urban settings. One of the key advantages of the RRT method is its ability to handle dynamic obstacles and changing conditions in urban environments [26]. Its probabilistic completeness ensures efficient pathfinding, improving the overall efficiency and safety of autonomous navigation in urban environments. The adaptive nature of the RRT method allows for real-time adjustments to dynamic obstacles, increasing the reliability of mobile robot [27].

The Petri nets modeling method [28] is a powerful tool for performance analysis in industrial systems, addressing real working conditions and uncertainties. It provides long-term availability of sub-systems through virtual simulation. The method helps identify sub-systems affecting system availability, enabling maintenance planning to reduce production loss. This paper demonstrates the methodology using a complex repairable manufacturing system.

The article [29] discusses the use of routing protocols (RPL) for load balancing in IoT networks, a protocol designed for remote organizations with low power usage and large defenseless to parcel mishaps. It is based on distance vectors and works on IEEE 802.15.4. RPL can support various connection layers, including IoT, which collects data from actuators and sensors. The paper presents an inclusive survey of existing load balancing schemes, matrices, objective functions and different RPL-based routing protocols related to load imbalance. The paper highlights the impact of load balancing when combined with RPL, highlighting the potential benefits of IoT networks in various applications such as urban communities, home automation, smart health and modern transportation strategies.

This paper presents [30] a performance assessment of a paint manufacturing unit using the stochastic Petri nets modeling method. The method provides realistic availability results for complex industrial systems and can handle multiple failures simultaneously. The system's availability is found to be 80.27%, and the results are compared with a Markov model solution, providing better maintenance planning understanding [31].

If the routing methods are based on predicting the behavior of the robot, better results will be obtained in the routing problem. For this reason, the discussion of controller design (referring to the controllers used in the closed-loop system) is very important for the robot. In this paper, two fuzzy controllers [32] are used for control. Because the rules of the fuzzy controller are usually determined based on experience, if designed correctly, this controller can, in addition to guaranteeing the stability of the robot, create smoother movements compared to the results obtained using other controllers. The robot used in this research is the four-wheeled vehicle used in [33]. In [33], for the control of the robot, a PI controller is used in this research; according to the dynamics of the desired robot using fuzzy control systems, controllers for this robot are designed and the results of using the PI controller and control. The fuzzifiers are also compared with each other.

2 Dynamic car model

The designer grows the tree considering the goal and constantly checks the possibility of adding new branches. The growth of the tree is done for a certain period of time. For the growth of the tree, a path is given to the controller as reference path 1. The method of determining the reference speed will be discussed in detail in the next sections. By taking the inputs of the track and the reference speed, the controller determines and applies to the car the control parameters of the car, which are the steering wheel angle and gas or brake. Similar to [25], the design of the path is done every 0.1 s (10 Hz), and every 0.1 s the path and the corresponding reference speed are sent to the controller. Then the controller calculates the command u (model inputs) in the form of gas/brake and steering angle and sends it to the device every 0.04 s (25 Hz). The update rate of the environment is equal to the execution rate of the designer (every 0.1 s).

The basic solution is that the designer provides a reference input r, which includes the new position on the page x and y, in other words (x, y goal) and also specifies the desired speed when traveling from the starting position to the goal position. A closed-loop system that includes the controller and the desired process (vehicle model) to send.

The dynamics used in this research is generally nonlinear dynamics ($x = f(x, u)$) and is related to a four-wheeled vehicle [34] x is the state variable and u is the command signal. The environment around the robot is a dynamic and non-static environment. For this reason, the robot's motion is planned online; for this purpose, the reference input update rate for the system is considered to be 0.1 s.

The control command u, which creates gas and brake conditions for the device, is generated by the controller at a rate of 25 Hz and according to the reference inputs and provided to the process [34].

3 Guidance controller

The guidance controller is considered based on following the reference path. The tracking problem is widely used in the case of ground robots and recently in the case of flying vehicles. In fact, the performance of this additional controller is similar to the performance of x $_{\text{new}}$ points in standard RRT. In standard RRT, the inverse algorithm was used to generate these points, which are known as forward points, but in [24], instead of using this method, using a circle that is similar to Fig. 1 at each moment of the device's position. A point on the reference path (the connecting line between x_{near} and x_{rand}) is drawn as a temporary target.

This method is much more useful compared to the Voronoi algorithm because the temporary targets are placed exactly on the reference path and their calculation is easier. The function of this method is as follows:

Assume that the reference path is a line segment that is specified using the start and end points. (This sample path is generated by the routing program.)

so that

$$p_k = \left[x_k, y_k\right]^T$$

$$p_{k-1} = \left[x_{k-1}, y_{k-1}\right]^T \tag{1}$$

They indicate the beginning of the route and the end of the route, respectively, and α_{k-1} indicates the angle of the route with respect to the x-axis and is calculated as follows.

$$\alpha_{k-1} = a\tan\left[\frac{y_k - y_{k-1}}{x_k - x_{k-1}}\right] \tag{2}$$

The temporary targets are obtained according to the intersection points of the reference path and the circle drawn from the position of the device according to the following relations.

$$\begin{gathered}\left(y_{\text{los}} - y\right)^2 + \left(x_{\text{los}} - x\right)^2 = \left(nL_{pp}\right)^2 \\ \frac{y_{\text{los}} - y_{k-1}}{x_{\text{los}} - x_{k-1}} = \frac{y_k - y_{k-1}}{x_k - x_{k-1}} = tan\left(\alpha_{k-1}\right) \\ \left(x_k - x(t)\right)^2 + \left(y_k - y(t)\right)^2 \le R_k^2\end{gathered} \tag{3}$$

In these relationships,L_{pp} is equal to the length of the device in question, n is a certain value greater than or equal to one, R_k is also equal to the radius of the target range. Another condition that should be considered here is as follows.

$$nL_{pp} \ge R_k \tag{4}$$

In the *CL_RRT* algorithm used in [34], a method similar to the [*LOS*] method is used to design the forward points. In this method, the circle radii are designed using an empirical relationship for the used model in terms of speed (this design is explained below); for this reason, this method is superior to the los method, but from the point of view that the relationship obtained. In the article, it is only designed for a specific device, compared to the los relationship, which is a mathematical relationship and can be used for any device, it is considered a weakness.

Therefore, the equations related to moving the car are stated below:

$$\begin{pmatrix}\dot{x} \\ \dot{y} \\ \dot{\theta}\end{pmatrix} = \begin{pmatrix}v\cos\theta \\ v\sin\theta \\ (v/L)\tan\delta\end{pmatrix} \tag{5}$$

In relation (5) and δ are considered as the inputs of the equation.

So that x and y refer to the position of the center axis of the machine. θ represents the angle of the machine axis in the positive direction of the trigonometric axis. The speed and δ angle of the front wheels of the robot in the direction shown and L is the length of the robot axis. Figure 3 shows the position of the stated variables in forward movement.

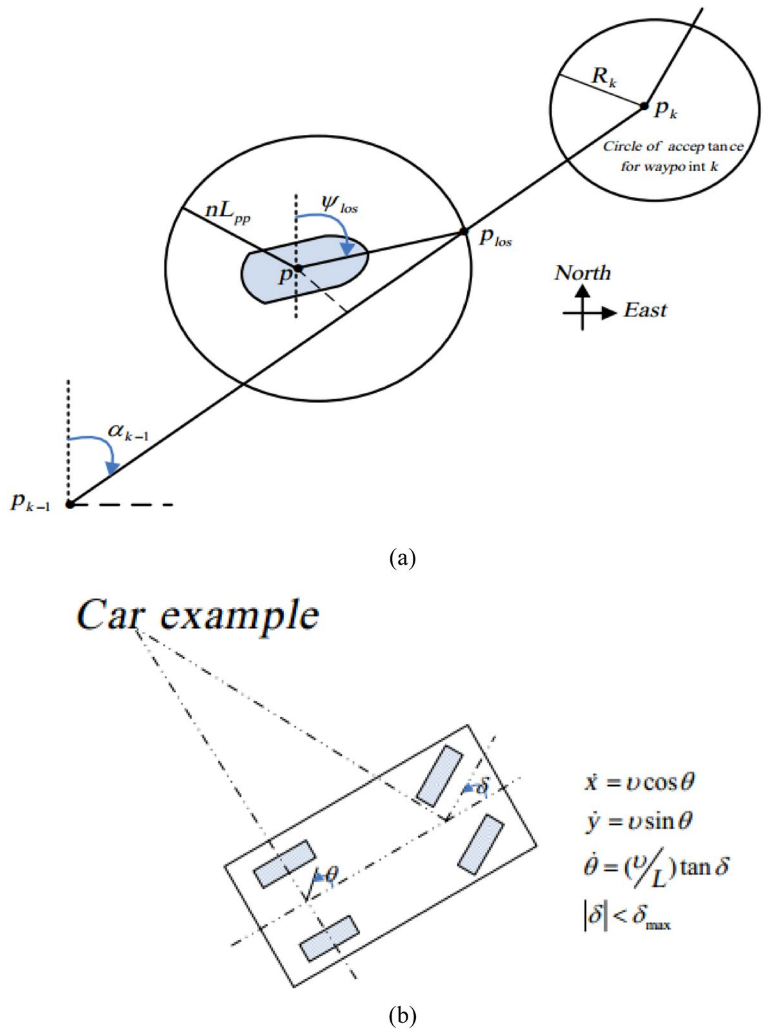

R_k
p_k
Circle of acceptance for waypoint k
nL_{pp}
ψ_{los}
p
p_{los}
North
East
α_{k-1}
p_{k-1}
(a)
Car example
δ
θ
$\dot{x} = \upsilon \cos\theta$
$\dot{y} = \upsilon \sin\theta$
$\dot{\theta} = (\upsilon/L)\tan\delta$
$|\delta| < \delta_{\max}$
(b)

Fig. 1 **a** Dynamic equations related to driving control of a car, **b** calculation of the position of temporary targets on the reference path in the los method

In Fig. 2A, l_{fw} is the distance between the center of the car's axis and the axis of the rear wheels, and L_{fw} is the distance between the car's axis and the reference track.

The value of l_{fw} is a specific and fixed value. η is the angle between the machine axis and the imaginary line drawn along the length L_{fw} from the machine axis to the reference track. It should be noted that this hypothetical line is used to calculate points on the reference path to perform the tracking problem, so that the robot not only has the coordinates of the beginning and end points of the path, but also the coordinates of the points on the path, or the so-called points on the path also see Fig. 2B.

The way to calculate the angle δ according to the value of η and L_{fw} and l_{fw} is as follows.

$$\delta = -\tan^{-1}\left(\frac{L\sin\eta}{\frac{l_{fiv}}{2} + 1_{fiv}\cos\eta}\right) \tag{6}$$

This relationship can be proved according to Fig. 2B and trigonometric relationships. The relationship related to the calculation of η is also as follows.

$$\eta = \theta + \sin^{-1}\left(\frac{y + I_{fw}\sin\theta}{L_{fiv}}\right) \tag{7}$$

If we consider the working point at the position $\delta = \eta = \theta = 0, v > 0$ and the state variables as $z = [y\theta\delta]$

$$\dot{z} = Az \tag{8}$$

Then, using linearization, we will have system equations

$$A = \begin{bmatrix} 0 & v & 0 \\ 0 & 0 & \frac{v}{L} \\ \frac{-L}{\tau L_{fv}\left(\frac{L_{fw}}{2}+1_{fw}\right)} & \frac{-L(L_{fw}+1_{fv})}{\tau L_{fiv}\left(\frac{L_{fiv}}{2}+1_{fiv}\right)} & \frac{-1}{\tau} \end{bmatrix} \tag{9}$$

The characteristic equation of the above matrix is obtained as follows according to $det(sI - A) = 0$.

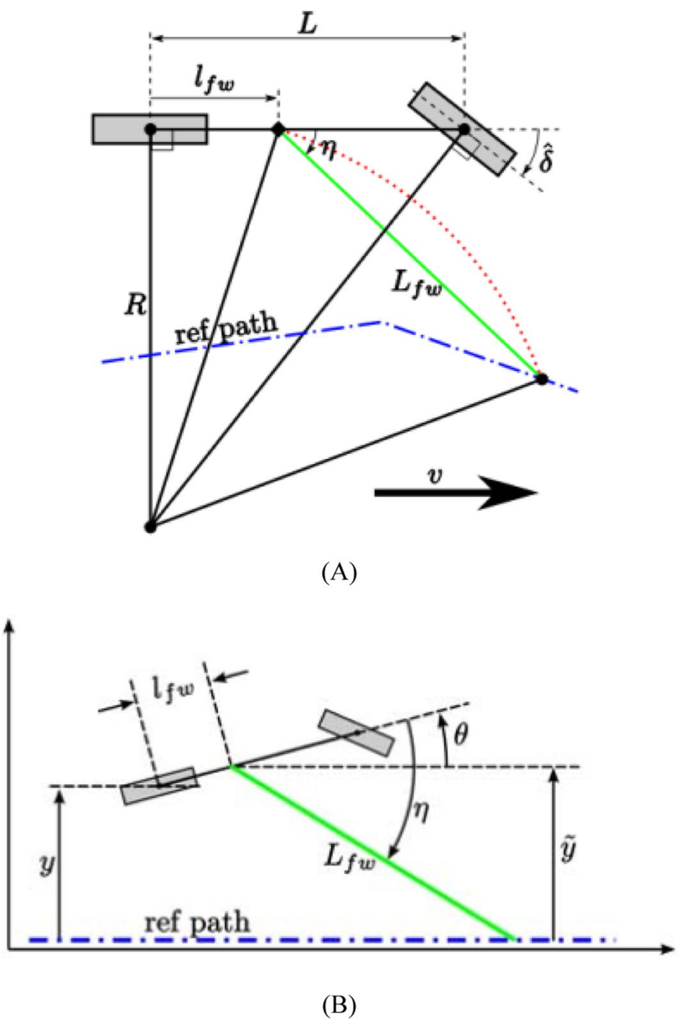


Fig. 2 **A** How to calculate the angles of the robot in the forward direction and **B** How to calculate the angle of the car using the angle of the axis and the line segment (arc) L_{fw}, when the reference path is parallel to the x-axis

$$s^3 + \frac{1}{\tau}s^2 + \frac{v\left(L_{fiv} + 1_{fw}\right)}{\tau L_{fiv}\left(\frac{L_{fi}}{2} + 1_{fw}\right)}s + \frac{v^2}{\tau L_{fiv}\left(\frac{L_{fw}}{2} + 1_{fw}\right)} = 0 \tag{10}$$

According to the Routh stability criteria, the following conditions must be met for the internal stability of this system

$$\begin{aligned} &\frac{1}{\tau} > 0 \\ &\frac{v\left(L_{fiv} + l_{fiv} - v\tau\right)}{\tau L_{fiv}\left(\frac{L_{fi}}{2} + l_{fwv}\right)} > 0 \\ &\frac{v^2}{\tau L_{fiv}\left(\frac{L_{fw}}{2} + I_{fv}\right)} > \end{aligned} \tag{11}$$

According to Eqs. (10) and (11), considering that $\tau > 0$ and $l_{fw} > 0$ and $L_{fw} > 0$ Eq. (12) can be reached.

$$L_{fw} > v\tau - l_{fw} \tag{12}$$

According to the above relationships, the length of the line segment connecting the center of the car to the reference path is dependent on the value of the speed according to various experiments that have been done with different speeds in different conditions such as different paths [34]. In order to ensure the internal stability of the system at different speeds, the following relationship has been obtained experimentally [34].

$$L_{\text{fiv}}\left(v_{cmd}\right) = \begin{cases} 3 & \text{if } v_{cmd} < 1.34\ \text{m/s} \\ 2.24 \times v_{cmd} & \text{if } 1.34\ \text{m/s} < v_{cmd} < 5.36\ \text{m/s} \\ 12 & \text{otherwise} \end{cases} \tag{13}$$

4 Speed controller

In order to track the speed of the *PI* controller, it is considered that the coefficients of this controller are designed according to the robust stability analysis and we have:

$$u = k_p\left(v_{cmd} - v\right) + k_i \int_0^t \left(v_{cmd} - v\right)d\tau \tag{14}$$

4.1 Vehicle speed model

In order to simplify the expression of the control command in order to create gas and brake conditions for the car and use the previous controllers, the function of converting the speed to the input of the device (u) which is as follows has been used.

$$\frac{V(s)}{U(s)} = \frac{K_n(v)}{\tau_v s + 1}$$

$$K_n(v) = 0.1013v^2 + 0.5788v + 49.1208 \tag{15}$$

Now we can consider the block diagram related to the control of the closed-loop system in the form of:

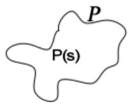
In [24] for the car model, the used speed controller is *PI* type. In this research, in addition to using this controller using fuzzy logic, a fuzzy controller has been designed to control the speed, which will be explained below, and the results of each will be compared with each other.

In Fig. 3, u the speed control signal, v_{cmd} is the desired speed of the robot and v is the speed of the robot and the coefficients k_p and k_I are the proportional and integral components of the controller which are designed according to the topics of resistive control. According to Fig. 3, the input of this speed error controller and its output are considered as the input for the vehicle speed model.

4.2 Durable performance

When we say that $P(s)$ is not an exact transformation function for the process under control, we can imagine that:

$P(s) \in P$



in other words

$$W_r(s).S_0(s)_\infty < 1 \tag{16}$$

where $S_0(s)$ is the sensitivity function and $W_r(s)$ is a suitable weighting function. The necessary and sufficient condition for resistant performance is as follows:

$$\left|W_r(j\omega) \cdot S_0(j\omega)\right| + \left|W_p(j\omega) \cdot T_0(j\omega)\right| < 1\forall\omega \tag{17}$$

in other words

$$\left|W_r(s) \cdot S_0(s)\right| + \left|W_p(s) \cdot T_0(s)\right|_\infty < 1 \tag{18}$$

which in relation (18) $S_0(s)$ is the sensitivity function and $T_0(s)$ is the complementary function of sensitivity, $W_r(s)$ and $W_P(s)$ are appropriate weighting functions. The proof of this relationship is stated in [24].

4.3 The design of k_P and k_I parameters

The parameter space method has been used to design the *PI* controller. The main idea in this method is to design the design specifications using the controller parameters in a desirable range. In this research, the design range means the available range $\left(k_P, k_I\right)$ for the parameters, the parameter space area that includes all the design specifications leads to the creation of a set of controllers that include the desired specifications. In this method, instead of having fixed control parameters we will have a specific range for setting parameters and conditions. The next step in this method is to select a point in this range to determine these values $(\left(k_P, k_I\right))$.

At first, a stable region for the location of the poles of the closed-loop system according to the criteria considered in the system response such as time, amplitude, overshoot value, permanent error value and o00 in the mixed device should be defined as the stability region according to Fig. 4, then with paying attention to this area, stability of controller parameters are designed. According to Fig. 4, because the stability region is completely on the left side of the imaginary axis, therefore, in this range, the closed-loop system is completely stable.

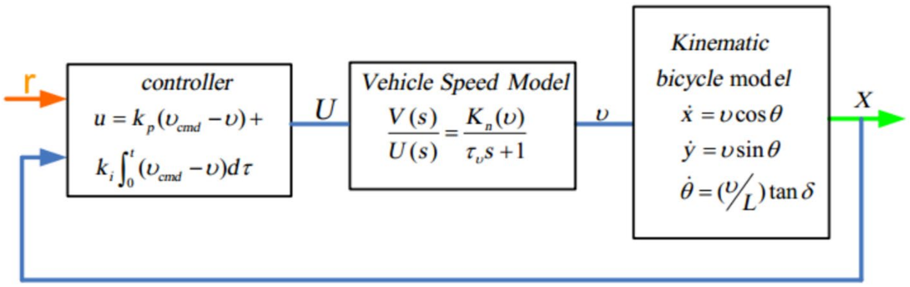

r
controller
$u = k_p(\upsilon_{cmd} - \upsilon) +$
$k_i \int_0^t (\upsilon_{cmd} - \upsilon)d\tau$
U
Vehicle Speed Model
$\frac{V(s)}{U(s)} = \frac{K_n(\upsilon)}{\tau_\upsilon s + 1}$
υ
Kinematic
bicycle model
$\dot{x} = \upsilon\cos\theta$
$\dot{y} = \upsilon\sin\theta$
$\dot{\theta} = (\upsilon/L)\tan\delta$
X

Fig. 3 Block diagram of control system with unit feedback. r is controller input and x is system output

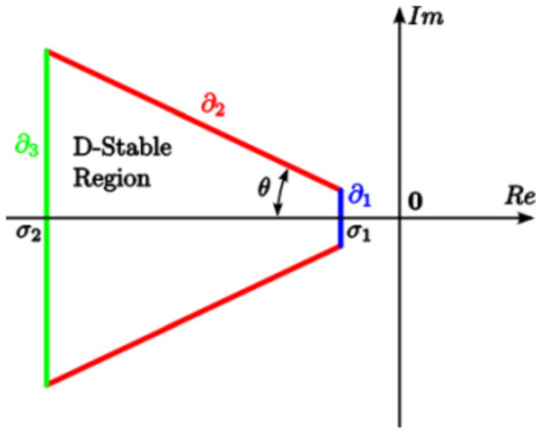


Fig. 4 Designing the stability region in the imaginary coordinate system

∂_1 specifies the lowest value for guarantee, ∂_2 stability specifies the lowest value for damping the system response and ∂_3 specifies the bandwidth range of the system.

The characteristic equation of the system according to Eqs. (10) and (11) is obtained as follows

$$p(s) = \tau_v s^2 + \left(1 + k_n k_p\right)s + k_n k_i \tag{19}$$

By replacing $s = -\sigma 1 + j\omega$ instead of ∂_1 and solving the equation $p(s) = 0$ and ω in the range of $(0, \infty)$ in the complex coordinate system, the values of k_P and k_I will be obtained as follows.

$$\partial_{1c} = \left\{ \left(k_p, k_i\right) \middle| k_p = \frac{2\sigma_1\tau_v - 1}{k_n}, k_i > \frac{\tau_v\sigma_1^2}{k_n} \right\} \tag{20}$$

Also, by putting $s = -\sigma_1$, the range of the real part of the poles will be obtained as follows.

$$\partial_{1r} = \left\{ \left(k_p, k_i\right) \middle| k_p k_i - \left(1 + k_n k_p\right)\sigma_1 + \tau_v\sigma_1^2 = 0 \right\} \tag{21}$$

Now, according to the two relations (19,20) and with the placement, we will have

$$\begin{gathered} \left(y_{los} - y\right)^2 + \left(x_{los} - x\right)^2 = \left(nL_{pp}\right)^2 \\ \frac{y_{los} - y_{k-1}}{x_{los} - x_{k-1}} = \frac{y_k - y_{k-1}}{x_k - x_{k-1}} = \tan\left(\alpha_{k-1}\right) \\ \left(x_k - x(t)\right)^2 + \left(y_k - y(t)\right)^2 \le R_k^2 \end{gathered} \tag{22}$$

Therefore, from the sharing of relationships (20) and (21) and (22) the area of stabilization of the process is obtained.

Now, if we consider a delay of T in seconds for the desired system as (23):

$$\frac{V(s)}{U(s)} = \frac{K_n(v)}{\tau_v s + 1} e^{-Ts} \tag{23}$$

It can be shown that for optimal phase gain m_φ and time delay T we will have the following limitations

$$\partial_{m_\phi,T} = \left\{ \left(k_p, k_i\right) \middle| k_p = f\left(m_\phi, T, \omega\right), k_i = g\left(m_\phi, T, \omega\right), \omega \in (0, \infty) \right\} \tag{24}$$

so that

$$\begin{aligned} f\left(m_\phi, T, \omega\right) &= \frac{\tau_v \omega \sin\left(m_\phi + T\omega\right) - \cos\left(m_\phi + T\omega\right)}{k_n} \\ g\left(m_\phi, T, \omega\right) &= \frac{\tau_v \omega^2 \sin\left(m_\phi + T\omega\right) + \sin\left(m_\phi + T\omega\right)}{k_n} \end{aligned} \tag{25}$$

Therefore, the range of parameters should apply in the following equation

$$\left|W_S S\right| + \left|W_T T\right|_\infty < 1 \tag{26}$$

In the above equation, s and T are sensitivity functions and sensitivity complement, respectively, and w_S and w_T are frequency-dependent weighting functions. The method of selecting functions w_T and w_S which is given in [34] is as follows.

1. To have a suitable input tracking function for the high-frequency range for w_S, it is therefore required that $|S(j\omega)|$=1 for $0 \le \omega \le \omega_S$
2. Dealing with sensor noise at frequencies$[\omega_T, \infty)$, so for $\omega_T \le \omega$ we must have $|T(j\omega)|$=1
3. Stability against multiplicative uncertainties is denoted by $W_\Delta(s)\Delta(s)$ so that $|\Delta(s)| \le$1 so it is necessary that $\left|W_\Delta(s)T(s) <\right|1$.

Since $S(s) + T(s) = 1$ weighting functions should be chosen in such a way that each function has the minimum value in its range. It should be noted that it is desirable to have a small sensitivity function at low frequencies and at above, the complementary function of the sensitivity has small values.

According to the tests performed in [34], it is determined that the maximum acceleration of the robot is $3m/s^2$. In order for the minimum acceleration to be $0.5m/s^2$ and the maximum acceleration not to be higher than $3m/s^2$, $\sigma_1 = 0.224$ and $\sigma_2 = 1.34$ should be considered. Also, in order to have an overshoot value less than 3%, $\theta = 41.92$ should be selected, be made.

The minimum desired gain value for the phase is equal to $m_\varphi = 60$. The robust performance is evaluated using the following weighting functions.

$$\begin{aligned} W_S(s) &= \frac{s + h_S\omega_S}{h_S s + h_S\omega_S 1_S} \\ W_T(s) &= \frac{h_T s + h_T\omega_T I_T}{s + h_T\omega_T} \end{aligned} \tag{27}$$

So that $l_s = 0.5$ and $h_s = 2$ and $\omega_s = 1$ for frequencies less than $1rad/s$, the tracking error value is less than 5%, and for higher frequencies, this value will be twice the reference frequency at most. Also, $l_T = 0.1$, $h_T = 2$ and $\omega_T = 2$. Finally, in [24], the values of kp = 0.04 and $k_I = 0.2$ are designed.

4.4 Speed controller using fuzzy systems

To create speed control, as mentioned, the speed model of the device is as follows.

$$\frac{V(s)}{U(s)} = \frac{K_n(v)}{\tau_v s + 1} \tag{28}$$

where k is a constant value whose value is calculated at any moment using the following relationship and taking into account the speed value.

$$K_n(v) = 0.1013v^2 + 0.5788v + 49.1208 \tag{29}$$

Now, in order to design a stable fuzzy controller, we first examine the conditions a to c according to the process under observation:

A) Considering that the value of τ_w is equal to 12, therefore the eigenvalues of A have (here only one eigenvalue).

The left side of the page is mixed.

b) The function of converting the system to the state space form is as follows (30):

$$\begin{aligned} \dot{v} &= -\frac{1}{\tau_v} v + u \\ y &= \frac{K_n}{\tau_v} v \end{aligned} \tag{30}$$

In this regard, τ is a constant and positive value and the value of K_n depends on the value of the speed, which according to the speed range (we do not have speeds with a value smaller than zero), this value is also always a positive value.

Considering that the process under observation is a single-input–single-output process. Therefore, this transformation function expressed for the speed model is a controllable and visible function.

c) According to the value of K_n and the speed range, the transformation function of the vehicle speed model is strictly positive.

Here, an attempt is made to design a controller using fuzzy logic for the system in such a way that 4 time parameters improve rise, overshoot, settling time and persistent error, and the controller should be such that the output of the system can follow the desired input for any desired initial condition. Meanwhile, the output of the fuzzy controller is considered as the input of the system.

As can be seen, the error value is equal to the difference between the desired input and the actual input

$$e(t) = y_d(t) - y(t) \tag{31}$$

For the design of the fuzzy controller, the speed e and the derivative of the speed changes de/dt obtained from the linearization of the equations are considered as the input of the equations for the fuzzy system. The output signals are calculated through the fuzzy membership functions that are dependent on these variables. The membership functions considered for the inputs and outputs of the fuzzy system are considered triangular.

In this section, for the input variables of error (e) and error changes (Δ e) and the output variable on the stickers (and Negative Small NS) (and Zero Z (and Positive Small PS) (and Positive Large) PL language Negative L e NL) is used Is.

The membership functions for the error and the derivative of the error, which are the inputs of the fuzzy system, are considered as (Fig. 5).

Both the theoretical approach and the trial-and-error method have been utilized in the introduction of laws. On the other hand, the stability of the controller is taken into account throughout its design. This controller is stable with respect to many error ranges, including the presence of measurement noise, based on how membership functions are constructed. Table 1 displays the information basis for we take the procedure in question to be symmetrical.

We also consider the centers of the membership functions for the output of the fuzzy system as follows:

$$\mathrm{NS} = -2, \mathrm{NL} = -2.5$$

$$\mathrm{Z} = 0$$

$$\mathrm{PL} = 2.5, \mathrm{PS} = 2$$

We make the centers of the output membership functions dependent on the error value between the desired input and the initial value because we want the fuzzy system to function correctly for every initial condition and every input value. In order to achieve the necessary performance under various conditions, the fuzzy controller must be able to modify its membership functions in accordance with the beginning conditions and the intended output value of the centers. The desired value and the error value arising from the beginning conditions can be multiplied by the centers of the output membership functions after this modification.

4.5 Speed design

In calculating the speed, we must calculate the value of the Vcoast speed (the second part of the movement that is done at a constant speed). This speed depends on the initial speed. We also show the duration of time when the device moves at this speed with tmin—if the relationship (32) holds, then the value of Vcoast = Vmax is considered.

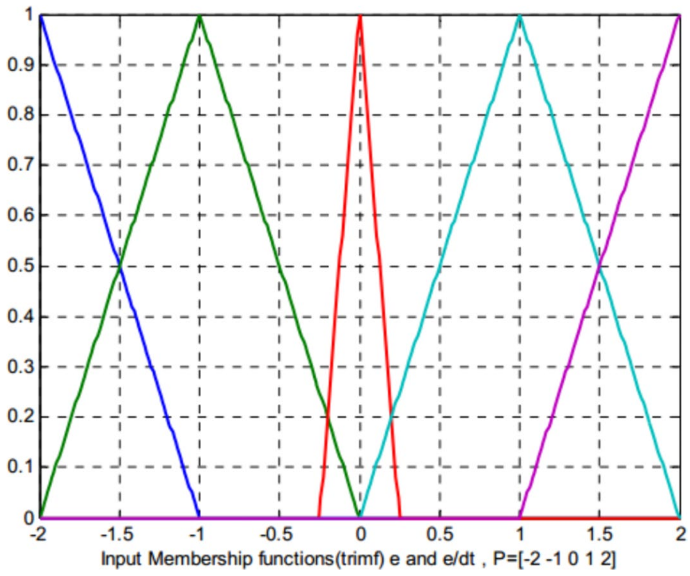


Fig. 5 Input membership functions, error and error changes

Table 1 Fuzzy system decision rules table

e	Δ*e*				
	PL	*PS*	*Z*	*NS*	*NL*
NS	*NS*	*NS*	*NL*	*NL*	*NL*
NS	*NS*	*NS*	*NS*	*NL*	*NS*
PS	*PS*	*Z*	*NS*	*NS*	*Z*
PL	*PS*	*PS*	*PS*	*PS*	*PS*
PL	*PL*	*PS*	*PS*	*PS*	*PL*
NS	*NS*	*NS*	*NL*	*NL*	*NL*

Table 2 Parameters of the robot to be simulated

parameters	value
Acceleration (A_{accel})	1 m/s^2
Braking acceleration (A_{decel})	−2.5 m/s^2
L	2.885 m
Maximum steering angle (δ_{max})	0.5435 rad
Maximum speed	10 m/s

$$\frac{v_{\max}^2 - v_0^2}{2A_{\text{accel}}} + v_{\max} t_{\min} + f\left(v_{\max}\right) < D \tag{32}$$

Otherwise, the value of coast speed is obtained by solving the following equation

$$\frac{v_{\text{coast}}^2 - v_0^2}{2A_{\text{accel}}} + v_{\text{coast}}\, t_{\min} + f\left(v_{\text{coast}}\right) = D \tag{33}$$

It should be noted that the first and second terms in the above relations are, respectively, the distance traveled in the movement with constant acceleration (the first part of the movement) and the movement with constant speed (the second part of the movement) and the function $f(v)$ represents the distance traveled in. The duration of the robot's braking time is defined by the following relationship and according to the friction forces in the environment as follows.

$$\begin{gathered} f(v) = \frac{v^2}{2A_{\text{decel}}} + \alpha_2 v^2 + \alpha_1 v + \alpha_0 \\ \alpha_2 = -0.0252, \alpha_1 = 1.2344, \alpha_0 = -0.5347 \end{gathered} \tag{34}$$

The parameters of the simulated car are presented in Table 2.

5 An example of the implementation of car dynamics

In this part, an example of the block diagram simulation of the control system diagram with unit feedback is shown considering the vehicle dynamics with specific initial conditions for PI and fuzzy controllers. It should be noted that in all the figures, the car is considered as Fig. 6, in this figure, the front wheels have the ability to change the angle relative to the axis of the car.

The values of the initial speed, initial angle and initial position for the robot are according to Table 3.

6 Multi-agent movement strategy

As previously said, each agent in the suggested approach first determines if it can meet its path with agents of higher priority before doing so with agents of lower priority. Whether dealing with agents of higher priority or agents of lower

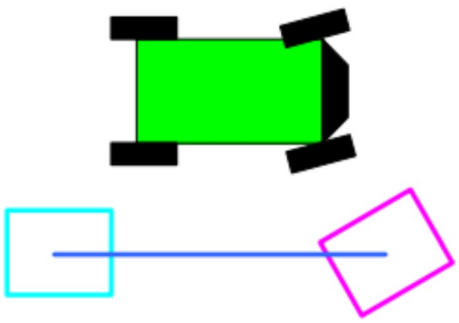


Fig. 6 The figure used in the simulations as a car, the front wheels have the ability to change the angle relative to the vehicle axis

Table 3 Parameters of the robot to be simulated

Parameter	value
Initial position	[10, 0]
The initial angle of the car axis	3 *Rad pi* /
Initial steering angle	*Rad* 0
Initial speed	1 m/s

priority, the decision-making procedure about their movement is the same. Because of this, two additional variables, m and n, have been incorporated into the architecture of the program flowchart. These variables stand for the factors that have a higher priority (lower number) and a lower priority (higher number), respectively.

6.1 Investigating the possibility of interaction between factors

Similar to the circle with a wider radius in the LOS approach, a parameter called step-safety 1 is used to check the risk of collision between agents. The step implies the distance between the two places of the robot according to the sample time is equal to 0.04 s. In this case, the radius is measured in steps, but it might be greater than the robots' sensors' field of vision because every agent always has access to the path that was intended for them. This characteristic is actually similar to a driver's level of awareness, where they can make an appropriate judgment based on their state and other driving cars before they are too close to each other. Hence, the greater this value, the earlier collisions can be identified and agents may be instructed to halt moving, and collisions can be avoided by having the agents change course. Stated differently, we assume a big circle centered at the robot's present position and containing the safety step radius surrounding each agent. Additionally, we take into account a smaller circle with a radius of ($-r$ safety step) that is centered at the robot's current location as shown in the picture, as well as the potential. We investigate where these circles intersect. Robots will check their paths with other robots whether their safety step radius circles do not meet; if not, they will attempt to adjust their paths or even stop, depending on how fast and far they are from the collision location. In order to avoid a collision between them, the lower priority factor should either halt the higher priority factor or the lower priority factor.

A collision ought to occur between the safety step and the -r safety step. In this instance, the agent with lesser priority has two flags activated: the security flag and the angle flag. When these flags are activated in the subsequent iteration, the reduction factor and the standard deviation of the angle for the new node in Gaussian sampling are increased for this factor, in accordance with Gaussian sampling of the mean and standard deviation of the distance. This way, in the event that the path needs to be changed in the subsequent iteration, the branches will be connected to the main node to increase the likelihood of locating a safe alternative path.

As mentioned, in Gaussian sampling, the samples are determined according to the current position and direction from the following relationship:

$$\begin{bmatrix} s_x \\ s_y \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} + r\begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix}$$
$$\begin{cases} r = \sigma_r |n_r| + r_0 \\ \theta = \sigma_\theta n_\theta + \theta_0 \end{cases} \tag{35}$$

where (x_0, y_0) is the current position, θ_0 is the current angle, n_r, n_θ are random variables with standard Gaussian distribution,σ_r, σ_θ is the standard deviation of the angle and length, respectively, and finally r_0 is the offset value that is determined during the program. In simpler terms, r and θ are two random variables with means θ_0, r_0 and standard deviationσ_r, σ_θ, where r determines the distance of the sampled point from the root (current position) and θ determines the angle.

Collision should occur in less than -r safety step. In this case, in addition to the security and angle flags being activated for the agent with lower priority, these two flags are also activated for the agent with higher priority (with different values for each). Also, the current desired route is considered unacceptable for the agent with lower priority, and the agent with lower priority is forced to make an emergency stop. Now, according to the speed and location of the agent with lower priority, we need to obtain the location of this agent in the future node while the emergency stop order 3 has been given to it.

We call this new position the emergency stop position. Now we check the distance of the agent with higher priority in two situations with the emergency stop position of the agent with lower priority.

Suppose the robot continues its path without changing its speed, in a next time step we predict the position of the robot with a higher priority and calculate its distance to the emergency stop position of the lower order agent—if the distance was more than 2 m (according to the dimensions of the robots and their safe distance from each other) there is no collision, so the higher order agent can continue on its way. If the distance is less than 2 m, we must check whether the collision can be prevented by emergency stop of the higher order agent (predicting the position of the agent in the future node similar to the emergency stop of the agent with lower priority) or not, if this is possible, that is, with an emergency stop. This factor means that the distance between the two factors is more than 2 m. The factor with higher priority must also make an emergency stop, otherwise (emergency stop of both factors) a collision is possible and the user must be notified of the collision before the collision.

In Figs. 7 and 8 the flowchart of multi-agent coordination program is shown.

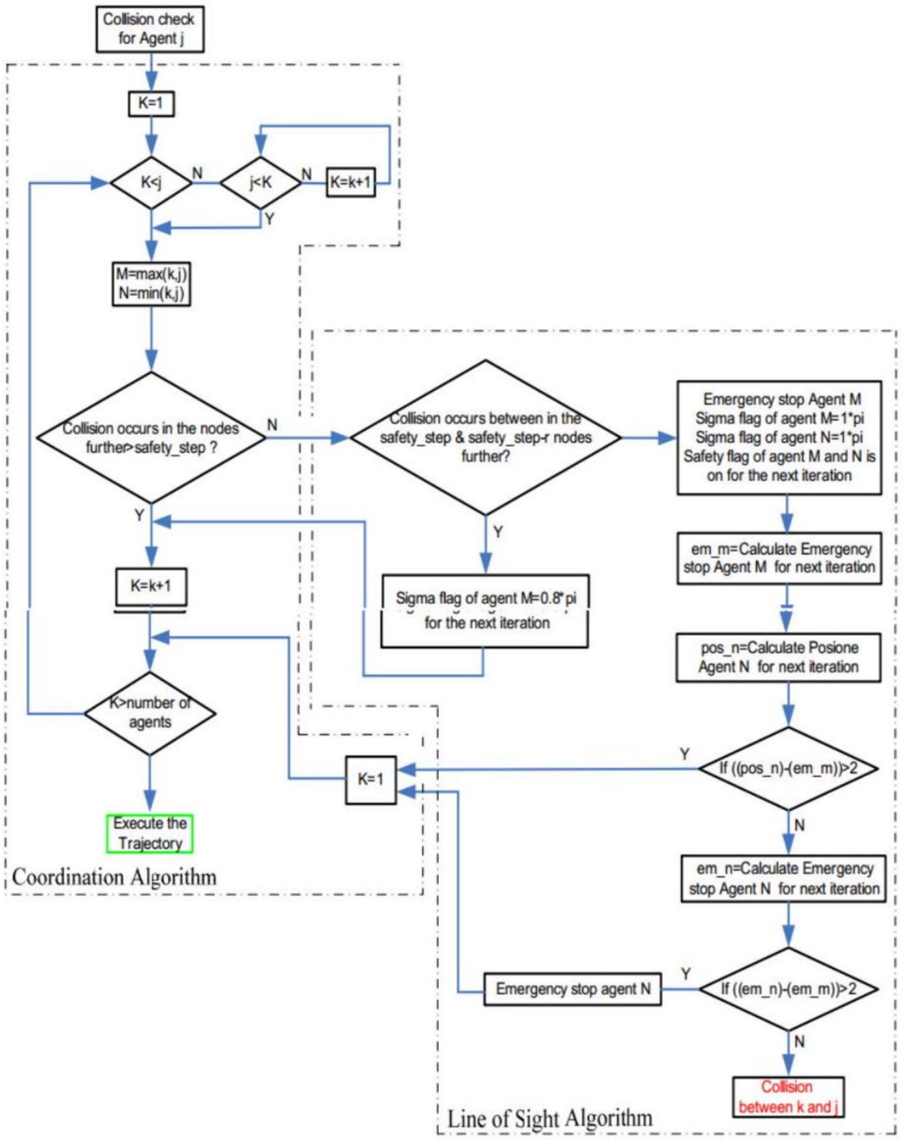
To calculate the new position of the robot (x_2, y_2) according to the current position and angle (x_1, y_1) and considering ΔL as a chord of a right triangle, we act as follows:

$$\begin{aligned} x_2 &= x_1 + \Delta L * \cos(\theta) \\ y_2 &= y_1 + \Delta L * \sin(\theta) \\ t_2 &= t_1 + T \end{aligned} \tag{36}$$

7 Simulation

The calculated routes are shown along with the curve of speed changes and distance from the reference route.

A- The results of simulating the control system with single feedback using the controller PI:



Collision check for Agent j
K=1
K<j
N
j<K
N
K=k+1
Y
M=max(k,j) N=min(k,j)
Collision occurs in the nodes further>safety_step ?
N
Y
K=k+1
K>number of agents
Execute the Trajectory
Coordination Algorithm
Collision occurs between in the safety_step & safety_step-r nodes further?
Y
Sigma flag of agent M=0.8*pi for the next iteration
Emergency stop Agent M Sigma flag of agent M=1*pi Sigma flag of agent N=1*pi Safety flag of agent M and N is on for the next iteration
em_m=Calculate Emergency stop Agent M for next iteration
pos_n=Calculate Posione Agent N for next iteration
Y
If ((pos_n)-(em_m))>2
K=1
N
em_n=Calculate Emergency stop Agent N for next iteration
Emergency stop agent N
Y
If ((em_n)-(em_m))>2
N
Collision between k and j
Line of Sight Algorithm

Fig. 7 Flowchart of coordination strategy program

Fig. 8 The path traveled by the robot using the PI controller (Fig. 8), taking into account the initial conditions according to Table 3

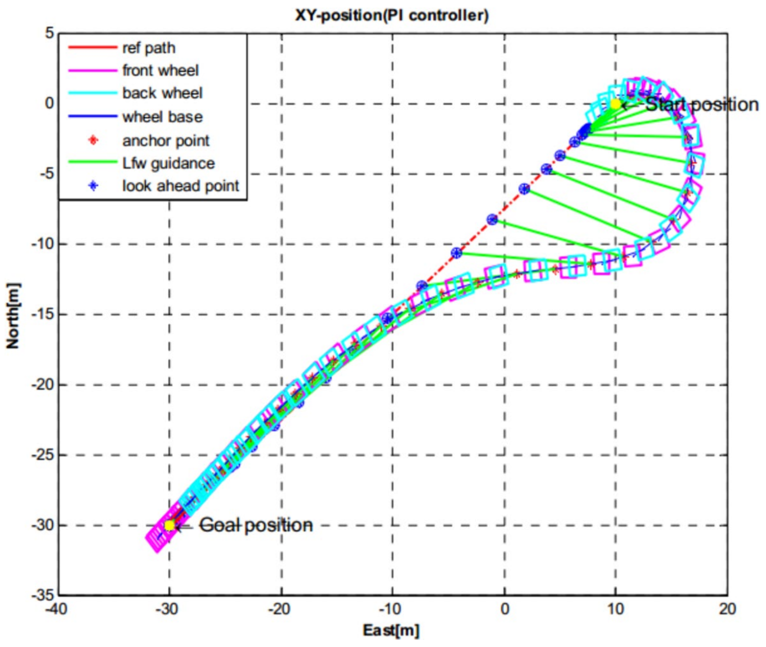
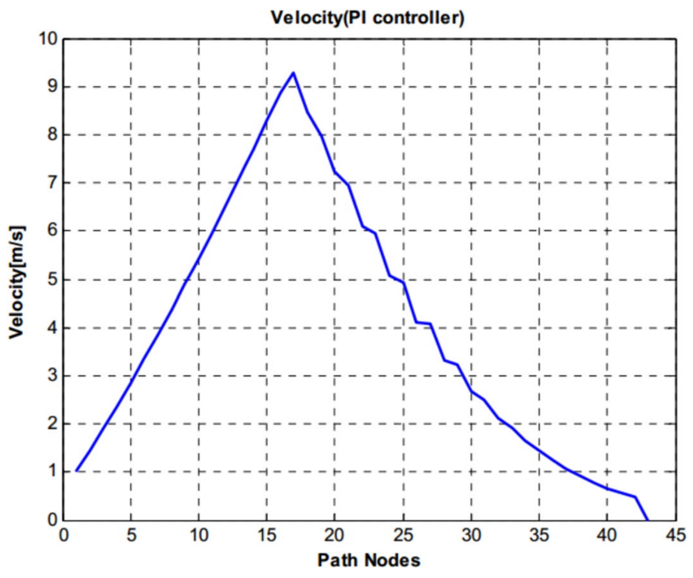


Table 4 Comparison of the results from the use of PI and Fuzz Controller

Average speed during motion (seconds/m)	The highest speed during motion (second/m)	The most distance from the reference route (m)	Similar time (seconds)	Type of controller
5.2784	10.2036	13.9598	1.0265	PI
5.3880	9.8679	13.9517		Fuzzy

As shown in Table 4, in all the car states, it follows the reference path well and reaches the end point. It should be noted that the condition of the repetition of the loop of the dynamic motion of the vehicle of the vehicle is that its distance from the lower point of the reference path is less than a determined value (equal to 1.5 m in the above figures). By comparing the shapes at low speed and high speed, it is observed that although the speed of motion on the reference path of Fig. 9 is more likely to be and the car must take a longer curve to place the path, but this issue is still a problem for the controller. Not built and the car correctly moves to the end point of the reference route.

Fig. 9 Speed of the robot along the path considering the initial conditions according to Table 3



The other point is that, as can be seen, the car speed curve is slightly different from the curved curve (10), which is due to the approximation in calculating the length of the route (D), because the precise length of the vehicle of the vehicle. At first, it is not clear, and therefore, the quantity of path D is approximately calculated.

The other point is that in the above shapes the exact location of the vehicle along the route is as calculated in the dynamic path calculation program. As shown in the shapes, many points are calculated along the route to accurately position the vehicle, resulting in minor change between consecutive points.

8 Preparing a program to overcome the known dynamic barriers for real hardware tests

If the environment around the robot also has differential constraints (dynamic constraints) in addition to algebraic constraints (the restrictions created by barriers), it is necessary to facilitate the simulation of initial and specific assumptions on dynamic limitations (which can be other robots, individuals and so on around the robot).

- The assumptions on the dynamic limitations in this study are as follows:
- Moving barriers have linear motion and their motion and speed are identified, and the barriers to the movement move and the robot is able to measure and detect the speed and direction of these barriers.
- Moving barriers have specific dimensions, and the robot follows its way in the open space without obstacles.

The time it takes to go to each node and the speed at each node must be recorded in the tree in order for the robot to be able to overcome the dynamic barriers (as well as those from other robots) and be a part of the design. It is actively expected that there is a program that generates the factor's dynamic pathway and connects the matching nodes in the same manner. Furthermore, the software yields a path that connects several nodes, with the distance between each pair of measures in the path equal to a pre-established value (Fig. 10, 11).

In the form of (11), the robot tries to take a safe and secure path to its move, considering the position of moving barriers.

8.1 Path design with 4 agents (particles)

With four components, the considered environment is identical to the prior environment. Factors 1 through 4 have initial positions of (0,0), (0,4), (0,8) and (0,12), in that order. The factors are ranked in priority order based on their numbers, with factor 1 having the highest importance and factor 4 having the lowest priority. Figure 12 shows an example of the route design outcomes.

The findings collected indicate that even in such a densely populated area, the agents' route was successfully designed without any collisions. Figure 12 makes it evident that the agents have frequently altered their intended course to stay clear of other agents.

The way the agents pass each other on the trip is shown in Fig. 13. It is believed that the factors have clashed in some circumstances, such as the first section, where the figures have shrunk. For this reason, an attempt has been made to display this part independently in Fig. 14.

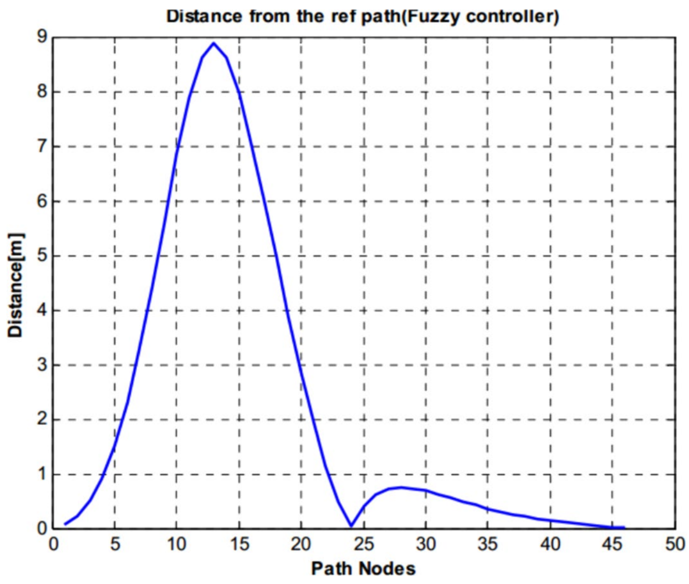


Fig. 10 Robot distance from the reference route along the way with the initial conditions (Fig. 10) in accordance with Table 3

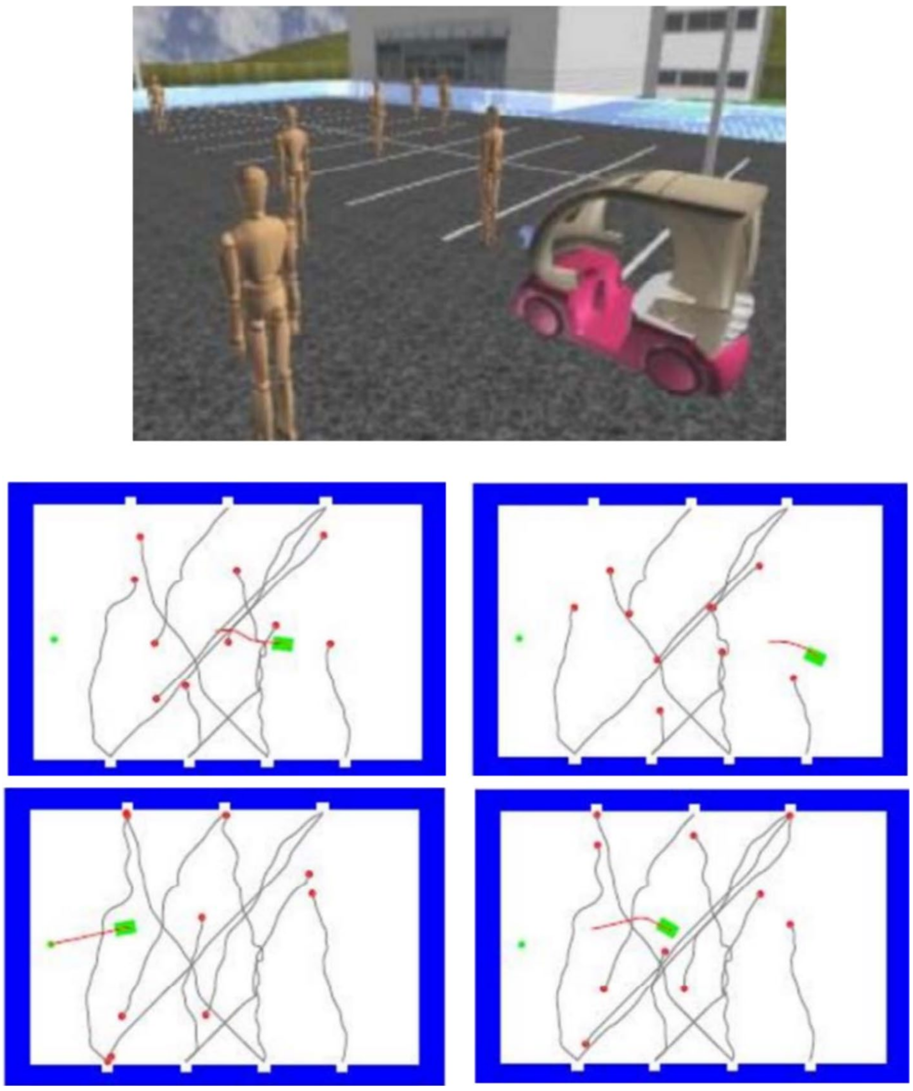


Fig. 11 Of the route design in an environment with moving barriers

It is assumed that the agents leave the environment after reaching the destination. The proper functioning of the factors and the lack of collision between them is well known in the above figures.

8.2 Implementation of CL-RRT considering *car* dynamics

There are two sections to the robot movement program flowchart. In this sense, the path's design for the robot's travel along it is connected to the first section. This path design checks both the robot's path's optimality and its ability to prevent collisions with obstructions. The RRT algorithm selects the first two locations, which are the start and finish of the robot's path, under the specified parameters. Taking into account the robot's dynamics, this ensures that the robot follows a straight course after receiving these two points. Or, based on his dynamics, he decides not to move in a straight path between these two sites, but instead may choose a curved path or any path other than a straight path to reach this temporary goal according to his dynamics. Thus, by giving the robot these two points, we should, in accordance with the CL-RRT algorithm, verify that the robot's dynamic path avoids obstacles, allowing us to choose the optimal dynamic path for the robot's movement. It is important to keep in mind that while designing the dynamic path, the robot must save its output and regulate its speed, input values and other requirements for actual movement at predetermined intervals. This is because the robot moves in two halves. Movement is both practical and simulative, and there is no longer any path simulation in practical movement (Fig. 15).

In order for the robot to travel in the intended path and follow the intended path for its movement as closely as possible, it just needs the values recorded in the simulation of dynamic movements for real movement. Naturally, it is also important to keep in mind that the robot will be able

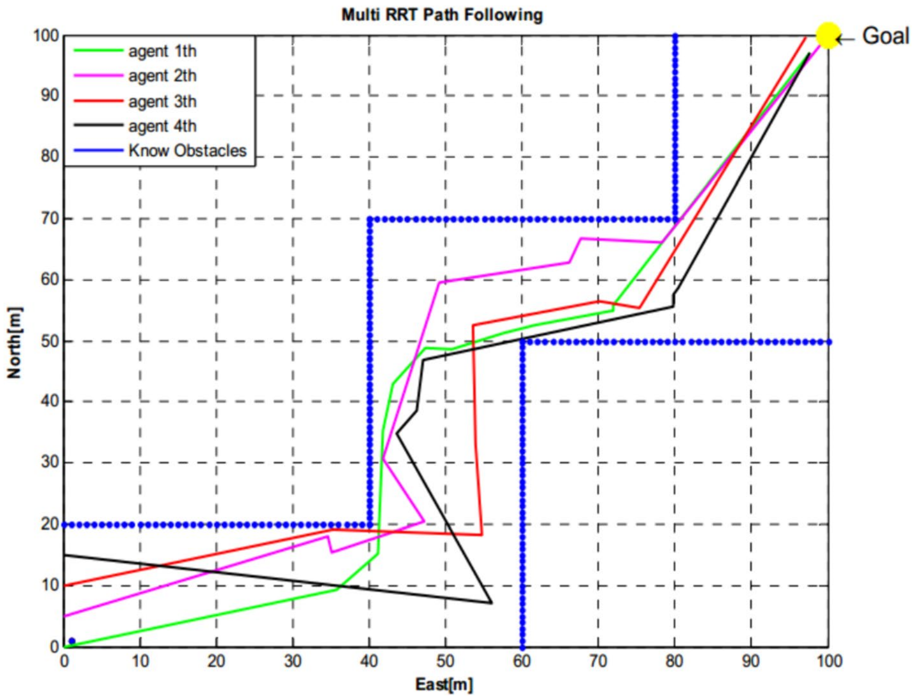


Fig. 12 Designed and traveled routes for 4 factors

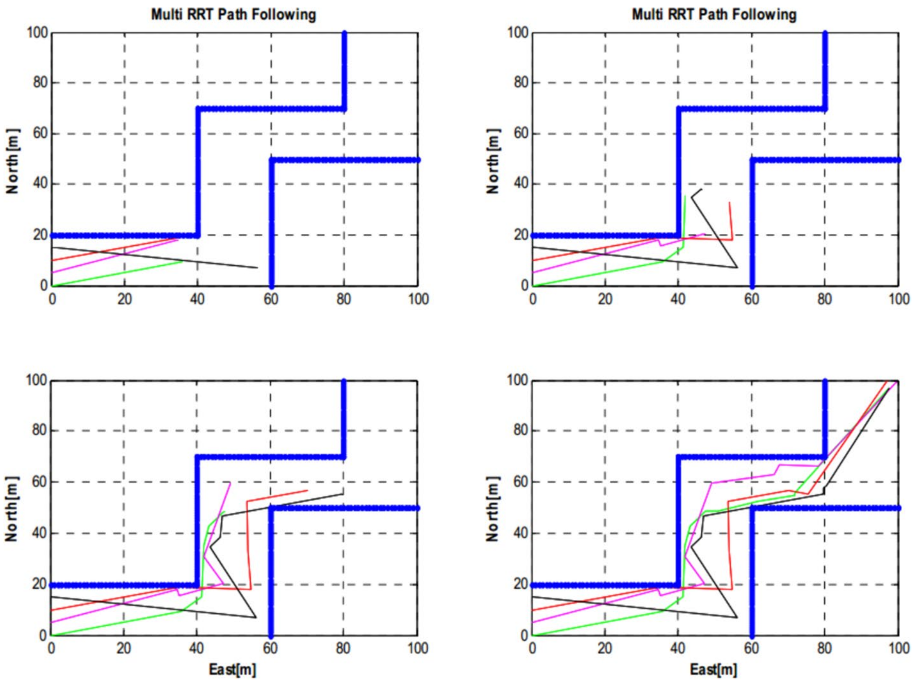


Fig. 13 Breakdown of designed and traveled paths for 4 factors

to follow the intended path more successfully the more intermediate points it takes into account.

It is important to note that since the objective of this project is for the robot to move online, both simulation and real pathfinding are carried out concurrently.

The flowchart related to the movement and design of the robot path is as follows.

9 Route design with 4 factors

The same environment with four elements is taken into consideration in this part. Factors 1 through 4 start at (0, 10, 10, 20), 10, 10, 10, 30 and 10, in that order. The factors are ranked in priority order based on their numbers, with

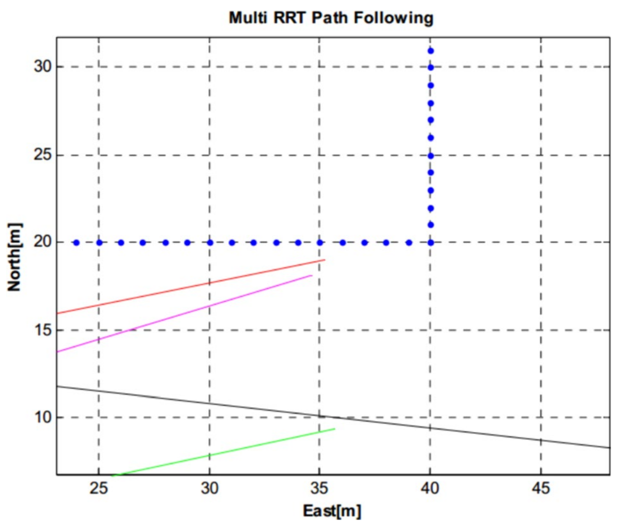


Fig. 14 The initial distance of the robots. As can be seen, the robots (particles) do not collide with each other

factor 1 being the highest importance and factor 4 being the lowest.) The suggested flowchart in (16) is accompanied by an example of how the route design results according to it.

The findings collected indicate that even in this highly dense environment, the agents' path planning was completed successfully and without any collisions. Agents have frequently altered their intended direction (thin lines) to avoid colliding with other agents, as the figure makes evident (Fig. 16).

For the findings displayed in the total (in Tables 5, 6), the agents' path length and time spent traveling through the initial environment are given. Ten meters is the starting distance between the agents at the start of the journey.

It can be seen that the paths obtained were without collision and despite the high density of agents and their proximity to each other, the design of the route was well established for the agents.

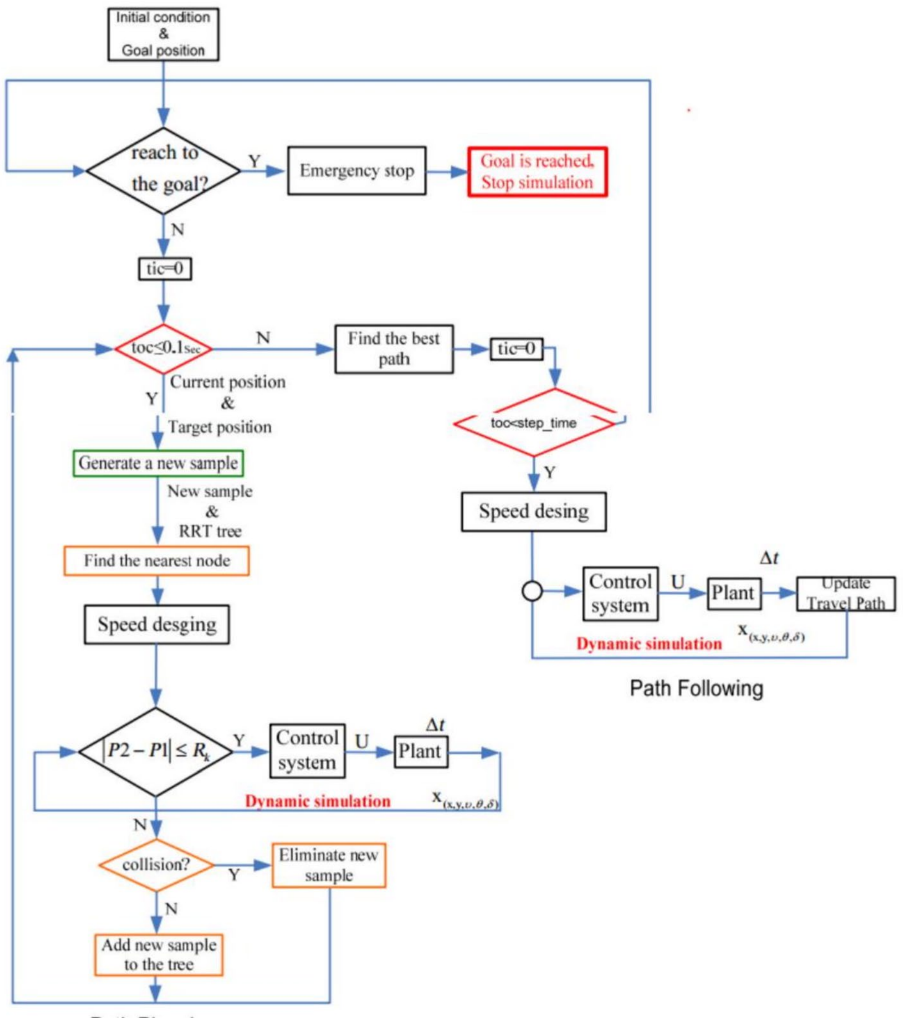


Fig. 15 Block diagram of the algorithm of designing and following the path by the robot using the CL_RR algorithm

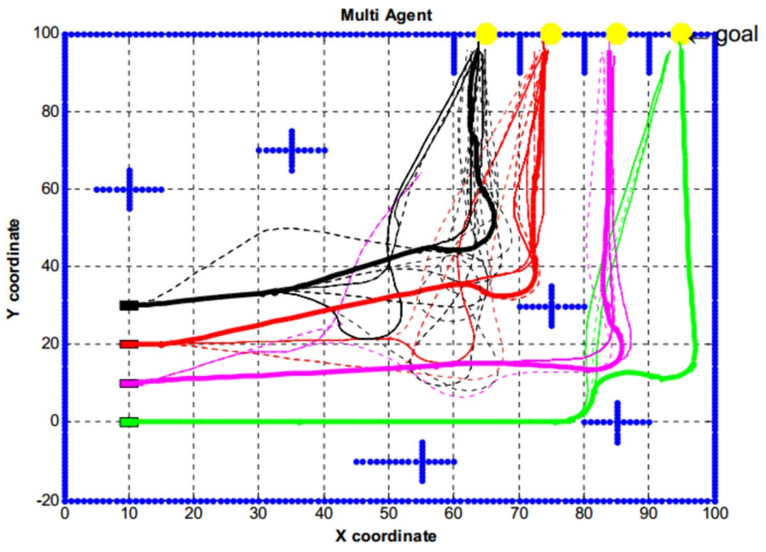


Fig. 16 Designed and traveled routes for 4 factors

Table 5 Length of the route and the time during the route for the obtained results

Path length (m)		Traverse time (s)
Agent 1	177	46.48
Agent 2	157	43.76
Agent 3	127.5	45.88
Agent 4	109.8	52.12

Table 6 Minimum distance between agents along the path in the obtained results

	Agent 2	Agent 3	Agent 4
Agent 1	10	20	29.9
Agent 2	×	9.32	19.78
Agent 3	×	×	7.3

10 Proposed method for multi-agent system routing along with queuing problem

As previously said, the virtual control approach is highly helpful in forming a regular shape. However, as was also mentioned, this method has a significant flaw in the topic of follower agent routing. The virtual control mechanism and the routing program are combined in the suggested approach. Assume that the objective of three agents moving in an environment that corresponds to an isosceles triangle is to coordinate their movements so that the guiding agent, an isosceles right triangle, is formed. It attempts to use routing to get to the target location and is depicted in green at the top of the triangle. Although the goals of the lower priority factors are the same as those of the other heads, their positions have changed simultaneously with

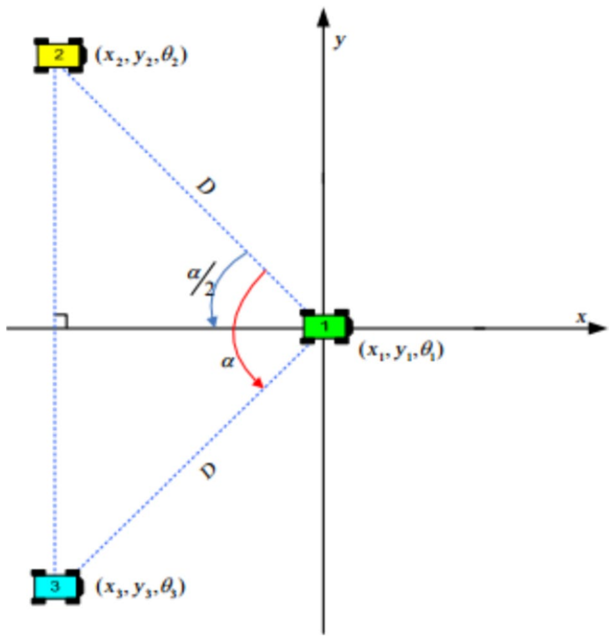


Fig. 17 Here are three robots in a regular forming position following a straight line

the movement of this factor with the highest priority and in accordance with the movement of that desired geometric shape (in this case, an isosceles triangle with a leg length of D meters). Every moment, lower priority factors are computed based on the intended configuration.

At first, assume that according to Fig. 17, the angle of factor #1 is zero with respect to the axes. Therefore, according to the angle and length of the leg of the triangle (D), the position of factors number 2 and 3 can be determined using the following relationships.

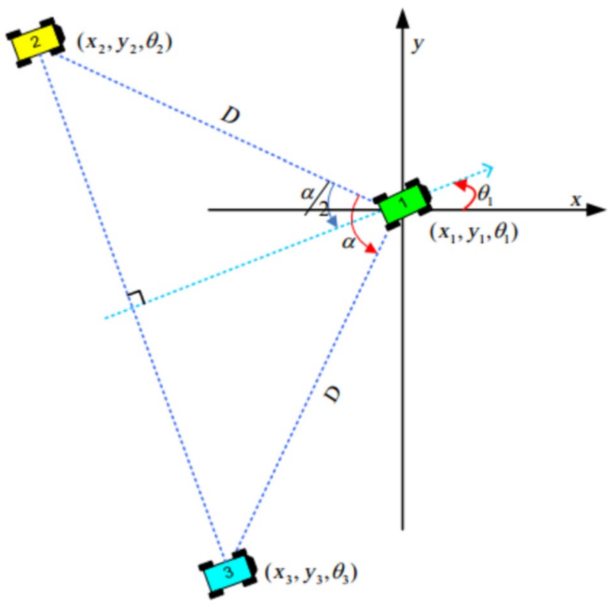


Fig. 18 Initial position of the robots

$$\begin{cases} x_2 = x_1 - D \cdot \cos(\alpha/2) \\ y_2 = y_1 + D \cdot \sin(\alpha/2) \end{cases} \quad \begin{cases} x_3 = x_1 - D \cdot \cos(\alpha/2) \\ y_3 = y_1 - D \cdot \sin(\alpha/2) \end{cases} \tag{37}$$

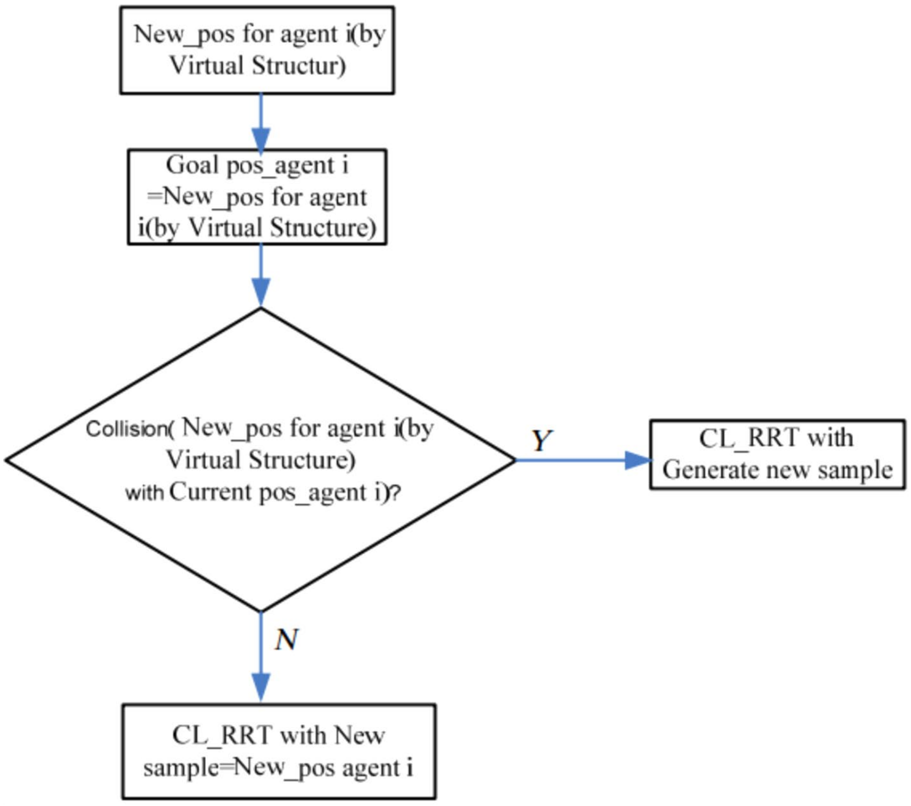
Now let us assume that factor number 1 is the change of angles with the size of 1 in the positive trigonometric direction according to Fig. 19.

In order to maintain the desired shape during the movement, it is sufficient not to include the angle of factor number 1 in specifying the position of other factors as follows in the previous relationships (Fig. 18).

$$\begin{cases} x_2 = x_1 - D \cdot \cos\left(\alpha/2 + \theta_1\right) \\ y_2 = y_1 + D \cdot \sin\left(\alpha/2 + \theta_1\right) \end{cases} \quad \begin{cases} x_3 = x_1 - D \cdot \cos\left(\alpha/2 + \theta_1\right) \\ y_3 = y_1 - D \cdot \sin\left(\alpha/2 + \theta_1\right) \end{cases} \tag{38}$$

Thus, the objectives of the lower order elements are ascertained in accordance with the geometric shape in issue during the movement by applying the aforementioned relations. On the other hand, the RRT routing algorithm uses Gaussian sampling to create new samples every instant based on the intended situation and the existing condition. The following is the suggested procedure for

Fig. 19 Flowchart of the suggested approach to address the routing problem and the agent movement problem concurrently. As can be seen, the only modification made to address the queuing problem is the sample strategy for the i-th agent, which is the next agent in line. (The RRT routing algorithm creates it.)



New_pos for agent i(by Virtual Structur)
Goal pos_agent i =New_pos for agent i(by Virtual Structure)
Collision(New_pos for agent i(by Virtual Structure) with Current pos_agent i)?
Y
CL_RRT with Generate new sample
N
CL_RRT with New sample=New_pos agent i

integrating the RRT algorithm with virtual control for the i-th agent (follower agent):

(1) Using the virtual control method, calculate the position of agent i (New_pos for agent i) by virtual structure and consider this position as the target position of agent *i*.
(2) If the direct path between the current position of the *i*-th agent and the position of this agent on the desired shape (the target of this agent) is without encountering obstacles, instead of using Gaussian sampling, this same position should be selected as a random sample. In other words, in this case as well the random sample is the same as the target position. Otherwise, generate a safe route to the target position using Gaussian sampling and routing.
(3) After generating a new sample in step 2, use the CL_RRT routing program to reach the target position (step 1) and generate the route.

This type of queuing architecture also allows for the control of the robots that follow at varying speeds. Because Fig. 19 serves as the basis for the optimal speed design, the follower robots' speed increases as their distance from the guide robot increases. Conversely, the follower robots' speed decreases as the guide agent's distance from the follower robots decreases.

The simulation example is positioned in Figs. 20 and 21 in accordance with the previously provided explanations. Its objective is to ensure that, in addition to the fact that the arrow's guiding agent has arrived at its intended place, the second agent. Furthermore, thirdly, they should use internet design to create their own movement path, making sure that each one matches the end of an arrow and avoids any barriers in the surroundings.

It is evident that as the second and third agents approach impediments in their path, they individually adjust their route by utilizing the RRT algorithm and attempt to go past

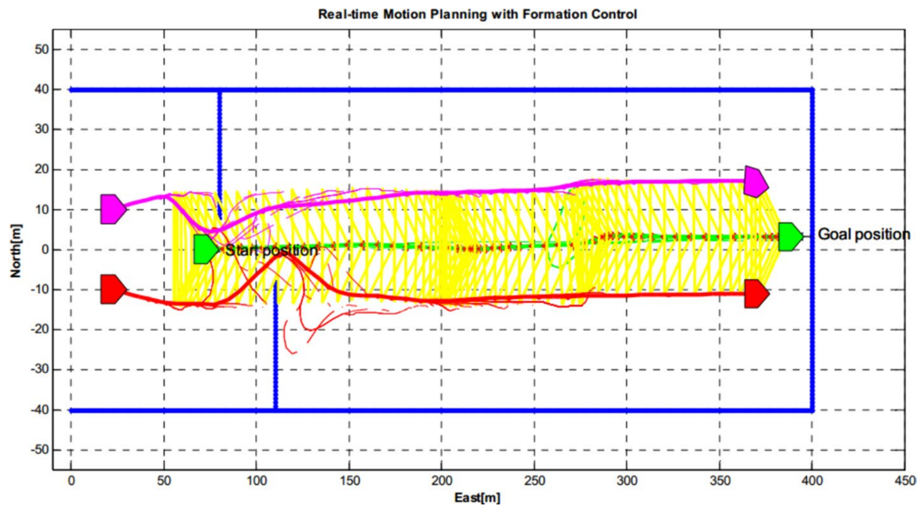


Fig. 20 Simulation of a multi-agent system along with the problem of queuing in an obstacle environment

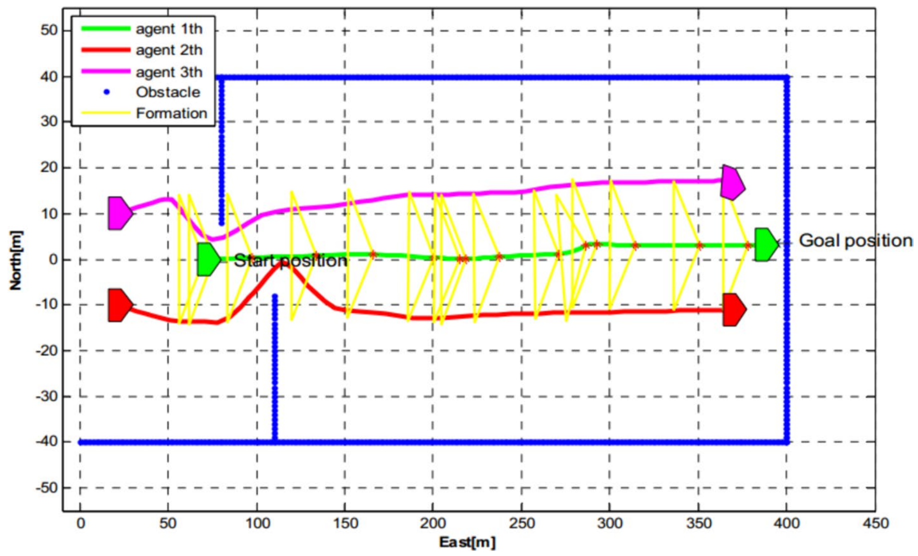


Fig. 21 Simulation of a multi-agent system along with the problem of queuing in an obstacle environment

them. Once more, they choose their course by applying the RRT algorithm to ensure that their movement conforms to the intended geometric shape. Additionally, it is evident that the changes in the first factor's movements are consistent with the movements of the second and third factors.

11 Conclusion

In this paper, after reviewing the existing methods for designing the path from a new method based on two prioritization-based coordination methods and the LOS method for designing multifactor pathways by the closed tree loop method, we first examine and develop an algorithm (CL-RRT) online for the purpose discussed, and changes in how new points in the environment were sampled to improve the performance of agents in the design of multifactor routes. It was designed to control the robot along the path using two PI and fuzzy controllers, and the results were compared with each other, with the fuzzy controller being more flexible during movement.

The following can be concluded from reviewing and comparing existing methods:

Existing methods for designing multiple factors can be divided into centralized and decentralized methods. The main disadvantage of centralized methods is that they increase the volume of computing by increasing the number of factors that make them suitable for use in large systems. On the other hand, calculations in decentralized design are divided between agents, and due to the absence of a central designer in the event of a failure of each of the factors, the rest of the single-way design methods can be divided into continuous and continuous methods. Discrete downturn with great and difficult issues in applying the dynamic feasibility, both in terms of dynamic and dynamic obstacles, is the means of not increasing the accuracy of the environment and consequently the dimensions of the problem, especially in complex and crowded environments.

The following results were compared to compare the selected methods to design the path:

The potential field method for simple environments works in a relatively rapid time, but it is difficult to get rid of the local minimum and find the route to the target. Another problem with this method is that it should be considered along the way, and as the robot approaches the barriers, the dimensions of the barriers become larger due to the proximity of the robot. It becomes very heavy.

The offline route design method for a variety of environments is well able to find the answer and always reaches a single answer for an environment, but the amount of time needed to execute the algorithm is even for simple environments and increases as the complexity of the environment increases. Also, given that the purpose of this thesis was to move multiple robots in the environment at any moment, depending on the cost function, the rating for each cell should be recovered in the direction, which is a lot of time and memory. It consumes, which is almost impossible in practice.

As a result, random search algorithms and RRT algorithms were used to move the robots online in the environment to allow the optimal direction to move in less time and with less memory volume. Subsequently, given that the tree’s expansion in this algorithm was based on the line between the starting point and the temporary goal, and according to the robot's dynamics, it could not be accurately moved on the right line and its move to achieve the goal set in the length of the route was curved, so the CL-RRT algorithm family, which was the expansion of the tree based on the device's dynamics, was used to reach zero on the problem of the robots' possible impact.

It should be noted, however, that the coordination method for the movement of the robots as a team that combines the two methods of prioritization and LOS, as well as the robot team's lineup method during movement, is due to the simultaneous routing and maintenance of the whole team. The first movement was used in this article.

The design algorithm of the multifactor route design for a variety of environments, including the large number of barriers and environments with a high number of agents, with consideration of the dynamics of the agents, is well able to design the path and find the appropriate paths without collision for the agents.

The car follows the reference path and reaches its end point, with a distance of less than a specified value. The controller correctly reaches its reference point despite traveling a longer curve. The car velocity curve has an informative curve (16), due to approximation in calculating the route length (D). The precise location of the vehicle along the route is calculated in the dynamic plan calculation program, resulting in minor position changes between consecutive points.

References

1. Khalaji AK, Jalalnezhad M (2021) Robust forward\backward control of wheeled mobile robots. ISA transactions 115:32–45
2. Khalaji AK, Mostafa J (2017) Modeling and backstepping control of a wheeled robot. 2017 IEEE 4th International Conference on Knowledge-Based Engineering and Innovation (KBEI). IEEE
3. Khalaji AK, Jalalnezhad M (2021) Stabilization of a tractor with n trailers in the presence of wheel slip Effects. Robotica 39(5):787–797
4. Ayawli BBK et al (2019) Optimized RRT-A* path planning method for mobile robots in partially known environment. Inf technol control 48(2):179–194
5. Liu L, et al. (2023) Path planning techniques for mobile robots: review and prospect. Expert Syst Appl:120254.
6. Qin H et al (2023) Review of autonomous path planning algorithms for mobile robots. Drones 7(3):211

7. Cao L et al (2022) 3D trajectory planning based on the rapidly-exploring random tree-connect and artificial potential fields method for unmanned aerial vehicles. Int J Adv Robotic Syst 19(5):17298806221118868
8. Sundarraj, Subaselvi, et al. (2023) Route planning for an autonomous robotic vehicle employing a weight-controlled particle Swarm-optimized dijkstra algorithm. IEEE Access.
9. Julius FS, Sitharthan R (2023) Improved RRT* algorithm-based path planning for unmanned aerial vehicle in a 3D metropolitan environment Unmanned Syst: 1–17.
10. Wang, W, et al. (2024) Towards optimization of path planning: An RRT*-ACO algorithm. IEEE Access
11. Dong L et al (2023) A review of mobile robot motion planning methods: from classical motion planning workflows to reinforcement learning-based architectures. J Syst Eng Electron 34(2):439–459
12. Alatise MB, Hancke GP (2020) A review on challenges of autonomous mobile robot and sensor fusion methods. IEEE Access 8:39830–39846
13. Hoy M, Matveev AS, Savkin AV (2015) Algorithms for collision-free navigation of mobile robots in complex cluttered environments: a survey. Robotica 33(3):463–497
14. Yang F et al (2022) Obstacle avoidance path planning for UAV based on improved RRT algorithm. Discrete Dyn Nat Soc 2022:1–9
15. Nasir J et al (2013) RRT*-SMART: a rapid convergence implementation of RRT. Int J Adv Robotic Syst 10(7):299
16. Elbanhawi M, Simic M (2014) Sampling-based robot motion planning: a review. Ieee access 2:56–77
17. Zong C et al (2021) Research on local path planning based on improved RRT algorithm. Proc Inst Mechanical Eng, Part D: J Automobile Eng 235(8):2086–2100
18. Wong C (2020)Adaptive task planning and motion planning for robots in dynamic environments
19. Liu Y et al (2019) Intelligent multi-task allocation and planning for multiple unmanned surface vehicles (USVs) using self-organising maps and fast marching method. Inf Sci 496:180–197
20. Seif R, Oskoei MA (2015) Mobile robot path planning by RRT* in dynamic environments. Int j intell syst appl 7(5):24
21. Hao K et al (2023) CERRT: a mobile robot path planning algorithm based on RRT in complex environments. Appl Sci 13(17):9666
22. Chen Long et al (2018) A fast and efficient double-tree RRT $^* $-like sampling-based planner applying on mobile robotic systems. IEEE/ASME trans mechatron 23(6):2568–2578
23. Zhao P et al (2023) Dynamic RRT: fast feasible path planning in randomly distributed obstacle environments. J Intell Robot Syst 107(4):48
24. Kiani F et al (2021) Adapted-RRT: novel hybrid method to solve three-dimensional path planning problem using sampling and metaheuristic-based algorithms. Neural Computing Appl 33(22):15569–15599
25. Yang K, Gan SK, Sukkarieh S (2013) A Gaussian process-based RRT planner for the exploration of an unknown and cluttered environment with a UAV. Advanced Robotics 27(6):431–443
26. Wang L et al. (2023) A Path Planning Framework Based on an Improved Weighted Heuristic RRT and Optimization Strategy. IEEE Transactions on Intelligent Vehicles
27. Moon C-B, Chung W (2014) Kinodynamic planner dual-tree RRT (DT-RRT) for two-wheeled mobile robots using the rapidly exploring random tree. IEEE Trans Industr Electron 62(2):1080–1090
28. Kumar A et al. (2021) Performance analysis of complex manufacturing system using Petri nets modeling method. J Phys: Conference Series. 1950(1). IOP Publishing
29. Rani S et al. (2021) RPL based routing protocols for load balancing in IoT network. J Phys: Conference Series 1950(1) IOP Publishing
30. Kumar A et al (2022) Stochastic Petri nets modelling for performance assessment of a manufacturing unit. Mater Today: Proc 56:215–219
31. Yadav AS, et al. (2023) Optimization of an inventory model for deteriorating items with both selling price and time-sensitive demand and carbon emission under green technology investment. Int J Interact Des Manuf (IJIDeM) 1–17
32. Chen L et al. (2020) Fuzzy kinodynamic RRT: a dynamic path planning and obstacle avoidance method. 2020 international conference on unmanned aircraft systems (ICUAS). IEEE
33. Kuwata Y et al. (2008) Motion planning in complex environments using closed-loop prediction. AIAA Guidance, Navigation and Control Conference and Exhibit.
34. Frazzoli E, Dahleh MA, Feron E (2002) Real-time motion planning for agile autonomous vehicles. J Guid Control Dyn 25(1):116–129

Authors and Affiliations

Hsin-Yin Hsieh[1] · Kuan-Hung Chen[2] · Chich-Jen Shieh[3] · Shavan Askar[4] · Mostafa Jalalnezhad[5]

✉ Kuan-Hung Chen
guanhong0224@sgu.edu.cn

Hsin-Yin Hsieh
P38094036@ncku.edu.tw

Chich-Jen Shieh
charleshieh@hqu.edu.cn

Shavan Askar
Shavan.askar@epu.edu.iq

Mostafa Jalalnezhad
mostafajalalneghad@yahoo.com; mostafajalalnezh@khu.ac.ir

[1] Department of Industrial Design, National Cheng Kung University, Tainan, Taiwan

[2] College of Tourism and Geography, Shaoguan University, Shaoguan, China

[3] Minjiang Scholar 2014, College of Economics and Finance, Huaqiao University, Fujian, China

[4] Information System Engineering Department, Erbil Technical Engineering College, Erbil Polytechnic University, Erbil, Iraq

[5] Department of Mechanics, Mechaical Enginearing, Kharazmi University, Tehran, Iran