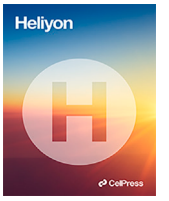
Heliyon 10 (2024) e26403

Contents lists available at ScienceDirect

Heliyon

journal homepage: www.cell.com/heliyon



Research article

BI-RRT*: An improved path planning algorithm for secure and trustworthy mobile robots systems

Honghui Fan [a], Jiahe Huang [b], Xianzhen Huang [b], Hongjin Zhu [a], Huachang Su [c,*]

[a] School of Computer Engineering, Jiangsu University of Technology, ChangZhou, JiangSu, China
[b] School of Mechanical Engineering, Jiangsu University of Technology, ChangZhou, JiangSu, China
[c] School of Computer Science, Nanjing University of Information Science and Technology, NanJing, JiangSu, China

ARTICLE INFO

Keywords:
Initial solution
Bidirectional search
Obstacle expansion
Informed sampling
Smooth path

ABSTRACT

The optimal RRT in elliptic space sampling (Informed-RRT*) is an extension of RRT that provides asymptotic optimality, however, it experiences gradual progress and close to obstacles. In the paper, we propose a novel path planning algorithm guided bidirectional Informed-RRT* (BI-RRT*), that introduces extension range, dual-direction exploration, and refinement in trajectory design. The growth range refers to maintaining an additional area from the obstacle to enhance the dependability of the path through preventing impacts. Bidirectional search is a search strategy using both start and target points for a initial solution. Smoothing improves path robustness by using cubic spline. Furthermore, simulation tests for the BI-RRT* algorithm are executed, and the efficacy of the suggested algorithm is confirmed through its application in a robot operating system (ROS). Simulations and experimental tests verify that the proposed algorithm improves the path planning capability. We emphasize the importance of safety, privacy, and reliability in the deployment of AI systems. Our algorithm ensures that the planned paths maintain a safe distance from obstacles, reducing the risk of collisions. Additionally, we prioritize privacy by adhering to data protection regulations and implementing secure communication protocols within the AI system. Moreover, we have applied rigorous testing and validation processes to enhance the reliability of our algorithm, ensuring consistent and accurate path planning outcomes.

1. Introduction

Path planning can be thought of as a way for a mobile agent to discover a route in the state space that enables it to move from a starting point to a destination point. In engineering applications, the position pose of the mobile agent is considered and its volume must be considered. We refer to the state space where the position pose as well as the volume of the mobile agent needs to be considered as the workspace.

Path planning in state space should first have completeness, when a path exists in state space, it can be found in finite time. Path planning methods that satisfy completeness are generally classified into resolution-complete path planning methods and probability-complete path planning methods.

Resolution-complete planning solves for the configuration space in the state space, and if a solution exists then the corresponding solution must be found in the resulting representation. Resolution-complete planning guarantees completeness and provides bounds

* Corresponding author.
E-mail address: shc_0804@163.com (H. Su).

https://doi.org/10.1016/j.heliyon.2024.e26403
Received 7 November 2023; Received in revised form 1 February 2024; Accepted 13 February 2024

Available online 20 February 2024

on the suboptimality of solutions, but can be computationally expensive, especially in high-dimensional spaces. As in A* [1] and Dijkstra algorithm [2], the configuration space is first discretized and feasible paths are found in the space. For the existence of a feasible solution then an executable path is returned, if a route is not available then the solution fails, and no path is returned. The efficiency of the computation relies on the intricacy of the environment.

For sampling-based planning, the space is continuous and more benign. The sampling approach generates a sparse representation of the configuration space on which to search in the hope of finding a solution and can only provide probabilistic completeness guarantees. Probabilistic completeness means that if a resolution is available, the likelihood of identifying a solution converges to 1 as long as the quantity of data points approaches infinity, and the solution method is well suited to high-dimensional planning. As in Probabilistic Roadmaps (PRM), Rapid-exploration Random Tree (RRT) use random sampling to solve for feasible paths. Instead of rasterization in the configuration space, random sampling is discretized in the feasible configuration space based on probabilities, a feature that makes probabilistic completeness-based path planning methods more responsive to solving feasible paths in complex environments. A specific path is returned if the test sample is large enough, but no failure result is returned if no feasible path exists.

In the random sampling approach, multi-objective as well as single-objective decisions can be made, where PRM can be used for multi-objective route formulation and the RRT algorithm for single-objective path planning problems. RRT algorithm has significant performance improvements for single-objective route formulation problems. The RRT algorithm is a progressive sampling exploration technique that doesn't necessitate parameter settings rectification and has good performance in use. An advanced version of RRT, RRT-Connect, uses dual trees to more rapidly seek viable solutions, especially if the target point is hard to reach. RRT-connect uses an incremental approach to search simultaneously using two trees, one from the starting position and the other from the target point. The connection is attempted from the target point. Once the connection is successful, the search is stopped.

Although RRT and RRT-Connect can quickly plan feasible paths, owing to the absence of refinement of RRT and RRT-connect through stochastic sampling in the state space, to solve this problem. Karaman introduced RRT* [3], which rewires the state space and cuts branches to optimize the search tree, resulting in a near-optimal solution. This type of planning is called asymptotic optimality. Because of the rewiring required, the number of samples needs to be increased to return a near-optimal solution. RRT*-Connect [4] combines RRT-Connect with RRT* and has a bidirectional approach that yields a nearly optimal solution. Like RRT*, RRT*-Connect continuously searches the entire region with the aim of providing a solution that is better than the current solution.

Although RRT* and RRT*-Connect yield nearly optimal results, they look at all states to optimize their paths. However, this is not an efficient way to reduce path costs. Gammell et al. [5] demonstrate that the effectiveness of existing methods decreases exponentially with the dimensionality of the configuration space. In response they proposed informed RRT*, and after obtaining a first solution, used informed sampling for RRT*. Informed sampling in a portion of the state space allows for faster optimization of the sampling space and better results.

Q-RRT* (Jeong et al. [6]) extends the scope of RRT* to reselect the progenitor node and establishes the 'level' of the progenitor node choice to beyond the circumference so that the expense of the complete route is less than RRT*. The exploration duration is extended due to the progenitor node encompasses a broader range. Gammell et al. [7] propose batch sampling, using multiple batches of multiple samples, using informed sampling, where the generated sample points are implicitly connected to the surrounding samples to form a random geometric graph (RRG) using the constructed dense RRG for incremental search. Strub et al. [8]. Proposed the ABIT* algorithm using advanced graph search, using expansion and truncation factors, to find an initial path faster and optimize it in the remaining time. These improved versions are since a portion of the state space to be acquired, the ellipsoidal space, needs to be determined by the initial solution. That is, before finding the initial path, they require examining the whole state space similarly as the RRT* [9], and only after the first initial solution is obtained do they start to perform informed sampling in a subset of the state space, and they will be slower in obtaining the initial solution than the unoptimized RRT. In RRT planning methods, the one-way search tree is typically replaced by a bidirectional search RRT-connect algorithm to enhance the effectiveness of initial solution finding.

In this paper we use an asymptotically optimal route formulation technique, inspired by the fusion algorithm described above we propose a single-objective decision path planning method based on informed sampling, using bi-directional search to overcome slow convergence, using expansion distance to avoid close to obstacles. The contribution can be clearly illustrated as follows.

Build initial solutions using a bidirectional search method to find possible path solutions in the state space in as short a time as possible. Lowers the time expense for determining the initial route.

The initial solution created is inflated and truncated to optimize the initial path. The obstacle space is inflated and the sampled points on the initial path are tested for collisions and if no obstacle exists between two points, the point is merged to the next sampled point to optimize the path length.

Combined with existing informed sampling. The set of state space ellipsoidal space subsets obtained after the initial path optimization is further reduced, thus reducing the number of iterations.

A path smoothing operation is performed on the resulting paths. Optimization of the stability and smoothness of the paths using the Cubic Spline interpolation method, followed by a final test on real scenarios. The advantage of the BI-RRT* algorithm with a bidirectional search strategy over other algorithms is that it is able to plan less costly and smoother paths in a shorter period of time, which better satisfies the mobility requirements of mobile agents.

The rest of the paper is organized as follows. The second section describes the algorithms relevant to this paper and the strategies for optimizing and improving them. The third section presents the algorithm optimization strategy and the associated pseudo-code proposed in this paper [10]. Section 4 gives a comparison between the simulation tests and the improved algorithm. Part V tests the improved algorithm in a robotic system and verifies the effectiveness of the improved algorithm. The final conclusions are presented in Part VI.

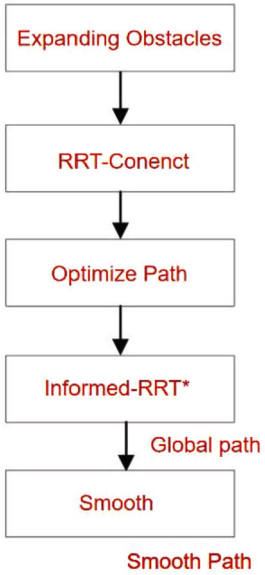


Fig. 1. Path planning flow chart.

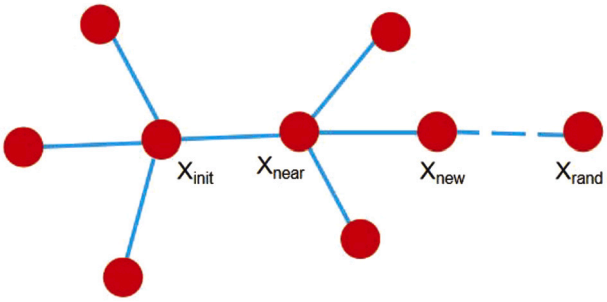


Fig. 2. RRT node extension.

2. Background

2.1. Problem definition

We formulate the optimal planning problem in a similar fashion to [11]. The primary contribution of this paper is the presentation of novel algorithms, namely PRM and RRT, which are shown to be asymptotically optimal, i.e., the cost of the returned solution almost certainly converges to the optimum. $x \subseteq \mathbb{R}^n$ is the bounded workspace of the path planning problem (n⩾ 2) where n denotes the dimension of the workspace. $x_{obs} \subsetneqq x$ denotes the obstacle region in the bounded space and the free region is denoted as x_{free}, which satisfying $x_{free} = x \setminus x_{obs}$. x_{free} contains all states in the workspace set. Given a path starting point x_{init} and an end point x_{goal} use the continuous function $x : [0,1] \geq Rn$ to denote its robot path. If for all $a \in [0,1]$, there is $s(a) \in x_{free}$, satisfying $s(0) = x_{init}$, $s(1) = x_{goal}$, which means that a continuous path devoid of obstacles from the initial point to the goal point satisfying the robot's own constraints can be planned as the solution of the problem. The solution merges the benefits of RRT-Connect and RRT* and finds solutions faster and converges to the theoretical optimum [12]. The optimal planning problem can be described as the task of locating a route s* [13], which makes a specified cost function $c : \sum \mapsto R_{n'}$, and satisfies the conditions $s(0)$, $s(1)$. Then there is $s^* argmin = c(s)|s(0) = x_{start}$, $s(0) = x_{start}$, $\forall a \in [0,1]$, $s(a) \in x_{free}$. The flowchart of the model for path planning in this paper is based on the traditional informed RRT* algorithm for improvement. The algorithm flow is shown in Fig. 1.

2.2. RRT algorithm

The basic principle is introduced: as shown in Fig. 2, the search process of RRT algorithm is like the growth process of a tree, which keeps spreading in all directions. The algorithm uses X_{init} as the initial point of the route and as the origin node of the stochastic tree T. It creates a random X_{rand} point in the robot configuration space X_{free}, finding the nearest point X_{near} in the tree, growing in the direction of with step u. Adding the generated branches and endpoints X_{new} to the tree if no obstacle is encountered.

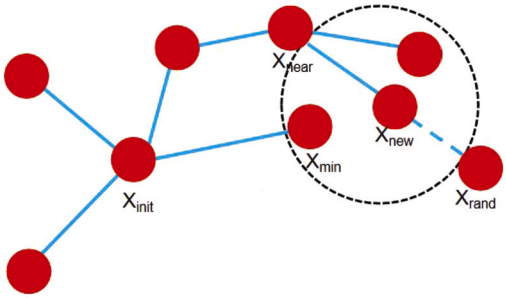


Fig. 3. Choose the least cost path.

Random points are generally uniformly distributed, when there are no obstacles, the tree will growing approximately uniformly in all directions, so that the space can be explored quickly [14].

The RRT algorithm is efficient and adaptive. By means of random sampling and fast expansion of the tree structure, the RRT algorithm has the capability to discover viable paths quickly, especially in computationally complex environments where the performance advantages are obvious. Compared to traditional route formulation techniques, like A* and Dijkstra algorithms, the RRT algorithm is more applicable to a wide range of problems and environments. It can be utilized in both 2D and 3D environments and can handle environments with complex geometries and dynamic obstacles.

The RRT algorithm is extensively employed in the domain of robotic route planning. It provides effective path planning solutions for diverse categories of robotic systems, including mobile robots, industrial robots, and UAVs, by searching large configuration spaces and generating feasible paths.

In the realm of autonomous vehicles, path planning is a key issue. RRT algorithms can be used for route planning in autonomous vehicle navigation to generate safe and efficient driving paths considering factors such as traffic rules and obstacles [15]. This is of great significance for realizing autonomous driving technology.

In addition, RRT algorithms also play an important role in virtual character motion planning. In video games and virtual reality, autonomous character motion planning is essential, and RRT algorithms can be used to generate intelligent paths for virtual characters to steer clear of obstacles and reach the desired destination.

2.3. RRT* algorithm

Although the RRT algorithm is a relatively efficient approach that can adeptly address route formulation challenges with partial constraints and offers numerous benefits. The RRT algorithm yields relatively poor feasible paths and the RRT* algorithm is founded on this improvement [16].

The primary characteristic of the RRT* algorithm is its ability to rapidly determine the initial route. As the quantity of sampled points rises, it keeps optimizing until it locates the destination point or reaches a predefined maximum cycle count. The RRT* algorithm optimizes asymptotically, as the number of cycles grows, the resulting path is more and more optimal, and achieving an optimal route in a finite time frame is unattainable. In simpler terms, a certain duration of computational time is necessary to obtain a reasonably satisfactory optimized route. The convergence time of the RRT* algorithm is a fairly significant area of research. However, it is indisputable that the expense of the route computed by the RRT* algorithm is considerably reduced compared to that of the RRT. The distinction between the RRT* algorithm and the RRT algorithm primarily resides in two recalculation procedures for the new node X_{new}, which the process of rechoosing the parent node for X_{new} and the process of rewiring [17].

The procedure of re-picking the parent node: the closest neighbors are located within a specified radius around the recently generated node X_{new} as a substitute for replacing X'_{new}s parent node. The path cost of the “near neighbor” node back to the initial position is calculated in turn, adding the path cost of X_{new} to each “near neighbor”. The shortest path X_{min} is selected, as shown in Fig. 3. Rewiring process. The resulting X_{new} is connected to the least costly parent node and the branching operation is performed. The cost function is further reduced. As shown in Fig. 4.

2.4. Informed RRT*

The informed RRT* algorithm was proposed to overcome the slow convergence of the RRT* algorithm. Informed RRT* is an improvement of the RRT* algorithm, which uses an elliptic sampling method instead of global uniform sampling. The elliptic region is represented as follows: the elliptic equation is $\frac{x^2}{a^2}+\frac{y^2}{b^2}=1$, the coordinates of the focal point are $(\pm c,0)$, the long axis measures a in length, while the short axis measures b, and the sum of the focal distance between them satisfies the distance from any the distance from the points on the ellipse to the two points totals 2a; We can get: $a^2=b^2+c^2$ in the informed RRT *. In the algorithm, we take the starting point xstart and the target point X_{goal} as the foci of the ellipse, and let half of the initial path be the long axis a, that is $a=\frac{c_{best}}{2}$. We have $c=\frac{c_{min}}{2}$, $b=\frac{\sqrt{a^2+b^2}}{2}$. So we get all the variables in the elliptic equation. The following Fig. 5:

In subsequent iterations, whenever a shorter route is discovered, the length of this shorter route is employed as the new c_{best}, the sampled ellipse updated.

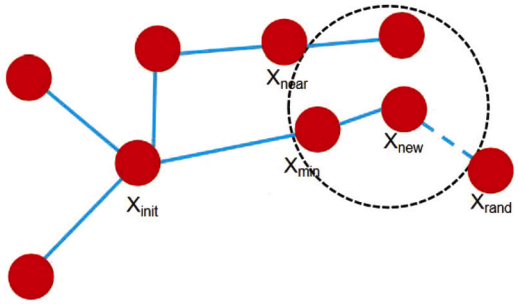


Fig. 4. Rewiring process.

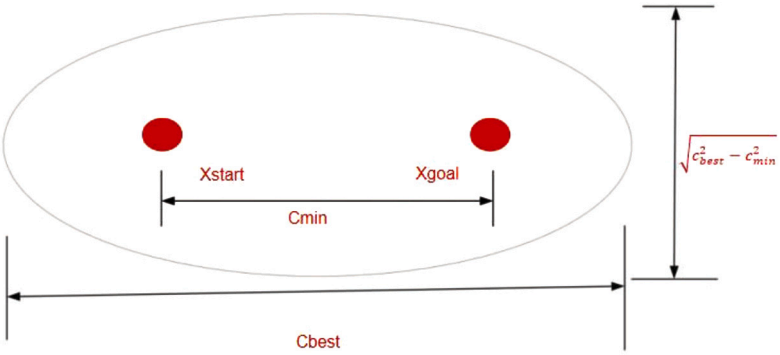


Fig. 5. Informed RRT* elliptical region.

Then the sampling takes place within the sampling region shaped like an ellipse. The sampled points are first sampled in the standard equation and rotated and translated to the actual sampling region. Two matrices are needed in this transformation process: the translation vector, and the rotation matrix. These two parameters only require calculation during the initialization phase [18].

The transformed coordinates are $\begin{bmatrix} x' \\ y' \end{bmatrix} = R \begin{bmatrix} x \\ y \end{bmatrix} + T$, where $R = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$ is the rotation matrix, θ represents the angle between the line that connects the starting point x_{start} and the target point x_{goal} and the x-axis. $\begin{bmatrix} x_{center} \\ y_{center} \end{bmatrix}$ is the translation vector, which can be expressed as the midpoint x_{start} and the target point x_{goal}. The process of Informed RRT* is the same as RRT* except for the sampling process.

2.5. Comparative analysis of related work

Although the Informed RRT* algorithm has greatly enhanced the velocity in comparison to the RRT* algorithm, further efforts are needed to enhance algorithm efficiency. In this paper, three optimization ideas are suggested for the informed RRT* algorithm: the expansion of obstacles, the establishment of the initial path using a bidirectional search strategy, the pruning operation of the initial route to acquire the optimal long axis, and the optimization of the route using elliptic space.

2.5.1. Expansion of environment model

During the mobile robot route planning procedure, robot obstacle avoidance is an important indicator of path feasibility. The robot is generally treated as a mass in simulation experiments, but in practice, the size of the robot is a factor that cannot be ignored. To guarantee the robot's safety, to make the simulation more practical, and to make it more feasible and convincing to treat the robot as a mass point, the obstacle area is expanded to half the width of the mobile robot in real operation [19].

2.5.2. Bidirectional search

The traditional RRT search method uses a tree T to search between the start and target points. RRT-Connect searches by using a bidirectional search strategy. A secondary tree is constructed within the vicinity of the target point for expansion, and each iteration is expanded by random sampling as in the original RRT algorithm, but the bidirectional expansion is improved in the direction of random sampling, where the first tree T_1 built at the start point has a direction to search from the starting point to the destination, and the tree T_2 built at the target point has a direction to search from the destination to the starting point until these two trees converge, which means the path search is successful. The whole algorithm ends. Each iteration needs to consider the balance of the two trees, i.e., the number of nodes of the two trees, exchange the measurement order, and choose the tree with fewer nodes to continue the expansion. This bidirectional search RRT technique possesses favorable search qualities and has substantially enhanced the speed

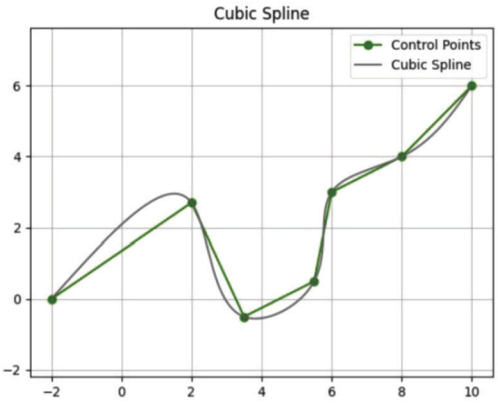


Fig. 6. Rewiring process.

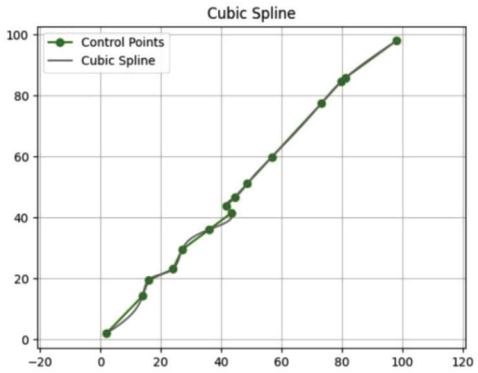


Fig. 7. Rewiring process.

and efficiency of the search over the original RRT algorithm. This expansion driven by inspiration renders the tree expansion more avaricious and unambiguous, making the two-way search RRT algorithm more efficient than the single-tree algorithm. The initial path can be obtained faster.

2.5.3. Path smoothing strategy

Cubic Spline interpolation is a segmentation function which has the following characteristics: the error on the nodes is kept within a small range [20]; the inherent high stability and the overall high smoothness make this interpolation function suitable for doing trajectory line algorithms [21]. The segmentation function means that $[a, b]$ is divided into n intervals $\left[\left(x_0, x_1\right), \left(x_2, x_3\right), ..., \left(x_{n-1}, x_n\right)\right]$, with a total of $n+1$ points, where two endpoints $x_0 = a, x_n = b$. The cubic spline implies that the curve within each small interval is a cubic equation and adheres to the following conditions: $S(x) = s_i(x)$ is a cubic equation on $[x_i, x_{i+1}]$ in each small segment interval; the interpolation condition is satisfied, that is $S(x_i) = y_i (i = 0, 1..., n)$; the curve is smooth, i.e., $S(x), S'(x), S''(x)$ is continuous. Then the constructive equation of the cubic equation is as follows: $y = a_i + b_i x + c_i x^2 + d_i x^3$. We transform this equation into the cubic spline function $S_i(x)$. From $S_i(x)$ we can see that each small interval has four unknown (a_i, b_i, c_i, d_i). There are $4n$ unknowns between n small intervals. To solve these unknowns, we need to solve the $4N$ equation. Since the $n-1$ interior points are all consecutive, i.e., the end point in the $i+1st$ interval and the beginning point in the $i+1st$ interval are the same point. Their first-order derivatives should also be the same, i.e., $S_i'(x_{i+1}) = S_{i+1}'(x_{i+1})$ hen there are $n-1$ equations; The second order derivatives of the interior points have to be continuous, i.e. $S_i''(x_{i+1}) = S_{i+1}''(x_{i+1})$, which also has $n-1$ equations, and all points must meet the interpolation requirement $S(x_i) = y_i (i = 0, 1, ..., n)$. All interior points except the endpoints satisfy $S_i(x_{i+1}) = y_{i+1}, S_{i+1}(x_{i+1}) = y_{i+1}$. There are $2n$ equations and there are a total of $4n-2$ equations. The lack of two equations can be solved for all unknown variables, with these two equations obtained using boundary conditions. There are three kinds of boundary conditions: natural boundary, fixed boundary, and non-nodal boundary. Natural boundary: the second-order derivatives at the endpoints are 0; Fixed boundary: the first-order derivatives at the designated endpoints are A and B; non-nodal boundary: the third order derivative of the first interpolation point is forced to be equal to the third order derivative value of the second point, and the third order derivative of the last point is identical to the third bounded inverse value of the penultimate point. From this we can create the equation $g_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$ in each subinterval $x_i \le x \le x_{i+1}$. After smoothing, the path is shown in the Fig. 6, you can clearly see from the figure that the path has become smooth. Fig. 7 shows the path after smoothing.

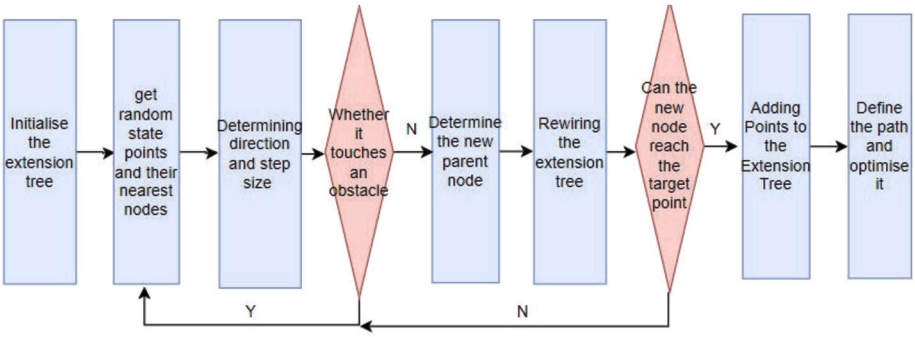


Fig. 8. BI-RRT*structure plan.

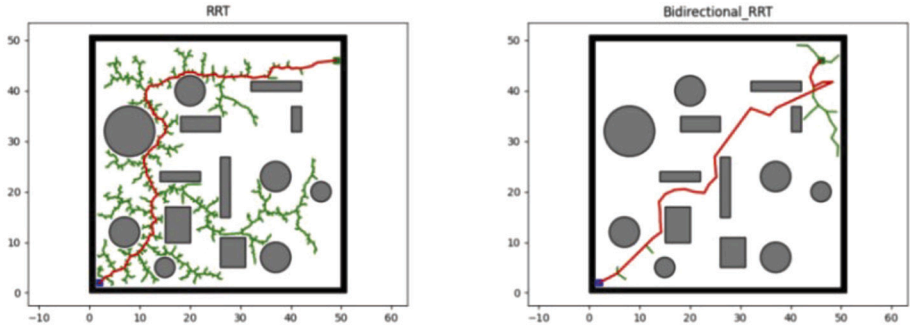


Fig. 9. Comparison between RRT planning and RRT-Connect planning.

2.5.4. Optimization steps

Utilize a bidirectional search approach to swiftly explore potential path solutions within the state space. This minimizes the time required for solving the initial path. The initially generated solution is adjusted by expanding and truncating it to optimize the path. By inflating the obstacle space and testing for collisions between sampled points on the initial path [22], points lacking obstacles in-between are merged with the subsequent sampled point, thereby improving the path's length optimization.

In conjunction with existing informed sampling techniques, a subset of ellipsoidal spaces within the state space is obtained after optimizing the initial path, resulting in fewer iterations.

Subsequently, a path smoothing operation employing the Cubic Spline interpolation method is performed on the obtained paths to enhance their stability and smoothness. Finally, the paths undergo testing in real-world scenarios [23]. BI-RRT* structure plan as shown in Fig. 8.

3. Optimization strategies and related pseudo-algorithms

As shown, Fig. 9(a) shows the way of path planning shows the way of path planning using RRT algorithm, and Fig. 9(b) shows the way of path planning using RRT-Connect search [24]. Elliptic space search strategy: to enhance the convergence rate of RRT* to reach the asymptotic optimal path planning a bounded elliptic space $X_e \subseteq X$ is proposed so that X_{rand} the sampling range falls in X_e. For the issue of locating the optimal route length in R^n space, we use the Euclidean distance as a heuristic for this algorithm: this state subset X_f can be depicted as $X_f = X_e \in X | \parallel x - x_{start} \parallel^2 + \parallel x - x_{goal} \parallel \subseteq c_{best}$, for representation. As shown in Fig. 5 X_f is the ellipse sampling domain for finding the minimum path cost in R^2 path sampling space, where X_{start} is the start point, X_{goal} is the target point, and the initial shape of the ellipse depends on the positions of the initial and destination points and the dimension of c_{best}. The algorithm deviates from RRT* in that once a solution is discovered, it directs its search towards enhancing the planning problem aspect of the solution. We add a bidirectional search strategy to the search strategy in elliptic space to rapidly locate the initial solution, enabling the search to be quickly focused on improving the path planning.

3.1. Pseudo-algorithm

First for the obstacle to r-length expansion, for whether there exists a route from the starting point to the destination point using IsFeasibleSolution() function to determine, if there is a return path τ, if the path planning is not successful then return Failure. If the

path planning is successful, we use PrunningOperation() to prune the existing path to obtain the minimum cost of cbest, and then bring cbest into the function OptimizeTree() to obtain the optimal initial ellipse space for path planning.

Algorithm 1 Advanced Informed RRT*.

$V_1 \leftarrow x_{start}, E_1 \leftarrow \emptyset$; $V_2 \leftarrow x_{goal}, E_2 \leftarrow \emptyset$;
$\tau_1 \leftarrow (V_1, E_1); \tau_{\leftarrow}(V_2, E_2)$;
$V \leftarrow x_{start}, E \leftarrow \emptyset$;
$c_{best} \leftarrow \infty; \tau \leftarrow (V, E)$;
$ExpansionDistance(r)$;
$\tau \leftarrow IsFeasibleSolution(\tau_1, \tau_2)$;
if $status \neq Failure$ **then**
    $c_{best} = PruningOperation(\tau)$
    $\tau \leftarrow OptimizeTree(x_{start}, x_{goal}, c_{best}$
end if
return τ

The algorithm uses a Bidirectional search RRT to determine if a feasible path exists and returns the path if it does. In k iterations, random sampling is conducted within the unoccupied space, and if the extended point is not in the obstacle space, it determines if it intersects with another tree, and returns the entire path if it produces an intersection. If it does not intersect with another tree at the end of the iteration count, the pathfinding fails and returns Failure.

Pruning of the obtained path. To put it simply, if no obstacle exists between two path nodes then extend the next path node until an obstacle exists at both path nodes, preserving the initial point and the prior path node where an obstacle node exists to attain the minimum path cost.

Algorithm 2 IsFeasibleSolution().

Input: τ_1, τ_2;
Output: $pathmaporFailure$;
$V_1 \leftarrow x_{start}, E_1 \leftarrow \emptyset\ V_2 \leftarrow x_{goal}, E_2 \leftarrow \emptyset$
$\tau_1 \leftarrow (V_1, E_1); \tau_2 \leftarrow (V_2, E_2)$;
for $k = 1$ to k **do**
    $x_{rand} \leftarrow RandomSample(k); k \leftarrow k + 1$;
    if $NotCextend(\tau_1, x_{rand}) \leftarrow Trapped$ **then**
        if $connect(\tau_2, x_{new}) \leftarrow Reached$ **then**
            return $Path(\tau_1, \tau_2)$;
        end if
    end if
    return $Failure$;
end for

In i iterations, firstly, the elliptic region of c_{best} is sampled for a given long axis. If the extended point is not in the obstacle space, judging whether the generated point reaches the target point. Saving the generated path node and finding its path cost if it does. If the new route cost is less than the route cost from the previous step, setting the new route cost to cbest to obtain the best search space and returning the optimized path after the number of iterations [25].

3.2. Analysis

For finding the minimum path in $x \subseteq \mathbb{R}^n$ space, the Euclidean distance is a heuristic that can be accepted. This use of bidirectional search in a portion of the state space can improve existing solutions, and then this subset of states x_f, $X_{\hat{f}} \supseteq X_f$ can be expressed in terms of the cbest closure of the current solution as: $X_{\hat{f}} = x \in X | \|x - x_{start}\|^2 + \|x - x_{goal}\|^2 \leq c_{best}$.

Algorithm 3 PruningOperation().

Input: τ
Output: c_{best}
$left \leftarrow 0, right \leftarrow 1$
while $left < len(\tau)$ **do**
    while $right < len(\tau) and\ freecollision(\tau[left])$ **do**
        $right + 1$;
    end while
    $left \leftarrow right$;
    $path.add(\tau[left])$;
end while
$c_{best} \leftarrow getDistance(path)$;
return c_{best}

For a positive cost function, then the cost of the optimal route from x_{start} to x_{goal} is bounded by the expense of the optimal route through $x \in X, h(x)$ denoted as the expense of the optimal route from x_{start} to x_{goal}, where $f(x)$ denotes the optimal cost from x_{start}

to x and $g(x)$ denotes the optimal cost from x to x_{goal}. Because the algorithm uses RRT* to converge asymptotically to the optimal route for each state, the function $\hat{f}(.)$, which evaluates the optimal cost, must estimate sufficient conditions for acceptability $\hat{g}(.)$ and $\hat{h}(.)$. These two parts are separate admissibility heuristics for $g()$, $h()$ respectively.

Also because the admissibility of $\hat{f}(.)$ makes the addition of a state to $X_{\hat{f}}$ required to enhance the solution. Due to the capacity to fill space of RRT's sampling, we learn that adding a state (which can be added to $X_{\hat{f}}$ with a sample other than a portion until it's filled within the RRT growth constraint parameter η of its boundaries.) becomes the likelihood of sampling a state like that. Thus, in any iteration, the probability of enhancing the solution by uniformly sampling a more extensive subset, $x^{i+1} \quad U(X_s), X_s \supseteq X_{\hat{f}}$, is no greater than or equal to the established metric $\lambda(.)$.

$$P(C_{best}^{i+1} < C_{best}^{i}) \leq P(x^{i+1} \in X_f) \leq P(x^{i+1} \in X_f) = \frac{\lambda(X_f)}{\lambda(X_s)}. \tag{1}$$

Algorithm 4 OptimizeTree().

Input: $x_{start}, x_{goal}, c_{best}$
Output: τ

```
for i = 1 to i do
    x_rand InformedSample(x_start, x_goal, c_best)
    if (τ, x_rand ← trapped) then
        if x_new ∈ x_goal then
            x_soln ← x_soln ∪ x_new
        end if
    end if
    newc_best ← minCost(X_soln)
    if newc_best < c_best then
        c_best = newc_best
    end if
end for
return τ
```

Using the elliptic volume sampled in $\mathbb{R}^n$ then we have $P(C_{best}^{i+1} < C_{best}^{i}) \leq \frac{c_{best}^{i}(c_{best}^{i2} - c_{min}^{2})^{\frac{n-1}{2}}}{\delta_n}$ where δ_n is the volume of the n-dimensional unit sphere and X_s be a hyperrectangle that snugly encloses the informed subset [26]. The sample probability of sampling uniformly from $x\hat{f}$ in the above equation (1) is $\frac{\delta_n}{2^n}$, when n = 3. Irrespective of the solution, algorithm parameters, etc., the likelihood of enhancing the solution through rejection sampling in each iteration is at most 32%. The expense of the optimal solution, c_{best}, will then linearly approach a theoretical minimum, c_{min}, if the subset is sampled uniformly without barriers [27]. In the scenario of uniform sampling, the expected value is

$$E[\hat{f}(x)] = \frac{nc_{best}^{2} + c_{min}^{2}}{(n+1)c_{best}} \tag{2}$$

We assume that η is larger greater than the planning problem's diameter by assuming that the RRT* reconnection parameter is larger than the radius of the informed sampling space, similar to the proof of RRT* asymptotic optimality. Then it follows from the above equation (2) the convergence rate of the solution cost follows a linear pattern with a rate μ that is solely determined by the state dimension:

$$\eta = \frac{\partial E[c_{best}^{i}]}{\partial c_{best}^{i-1}}|_{c_{best}^{i-1} = c_{min}} = \frac{n-1}{n+1} \tag{3}$$

For an unobstructed space, then the optimal solution path is conditioned on the line between x_{start} and x_{goal}, i.e. $x \in X_{\hat{f}, c(s)=\sqrt{x_{goal}^{2} - x_{start}^{2}}}$, which gives $s^* - c(s) = 0$; since RRT* equation (3) is itself an asymptotically optimal planning algorithm, the process of suboptimal solution is a continuous search for the smallest c_{best} in a finite time subset of state space sampling points, in order to obtain a smaller informed space.

4. Simulation experiment

This paper uses Python for code implementation. The experimental environment is configured as follows: processor is Intel(R) Core(TM) i7-4790 CPU@3.60 GHz, 16G RAM device, Win10 Professional 64-bit. To assess the effectiveness of the BI-RRT* algorithm put forward [28] in this paper, a comparison is made between Informed-RRT* and the improved algorithm in this paper; The pink boxes are inflated for obstacles and the original map has a resolution of 50*50; The fixed starting point (blue solid point in the bottom-left corner) and the ending point (red solid point in the upper right corner), with coordinates $[2, 2]$ and $[46, 46]$ respectively. In the diagram we use grey rectangles and circles to represent the obstacles, which are randomly generated in the diagram. Where blue indicates the starting point, red indicates the target point and the red line is the planned path. The experiment is divided into two parts: environment A Fig. 10(A) with a resolution of 50*50, a complex environment B Fig. 10(B) with a resolution of 100*100 and a simple environment C Fig. 10(C) with a resolution of 30*30, where environment A, B and C are shown in the Fig. 10 below.

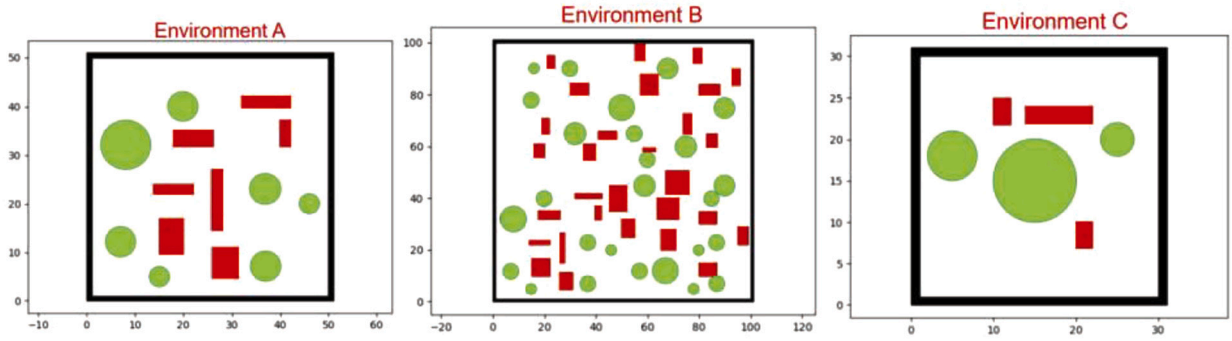


Fig. 10. Comparison between Environment A, Environment B and Environment C.

Table 1
Infomed-RRT*.

Sampling rate / Sampling size	5% Time used	5% Path length	10% Time used	10% Path length	10% Time used	10% Path length	Mean Time used	Mean Path length
1%	124.5	64.7	88	64.7	67.5	64	93.33	64.47
2%	54.3	64.5	37.9	69	32.5	69.3	41.57	67.6
4%	29.5	66.2	17.4	68.2	11.35	67.75	19.41	67.3
6%	38.4	66	9.65	69	14.15	69	20.73	68
8%	19.06	68.6	10.2	67.7	10.1	68.2	12.31	68.17
10%	16.63	67.5	11.3	69.4	9.4	72.4	12.44	69.77
20%	5.4	72.1	3.9	79.6	7.6	72.7	5.6	74.8
Mean	41.11	67.73	25.48	69.64	21.8	69.05		

Table 2
BI-RRT*.

Sampling rate / Sampling size	5% Time used	5% Path length	10% Time used	10% Path length	10% Time used	10% Path length	Mean Time used	Mean Path length
1%	125	64.6	90	65	67.3	64	94.1	64.53
2%	55.4	65.14	33.33	68.76	27.04	69.33	38.59	67.76
4%	13.4	65.9	7.07	68.8	7.0	68	9.16	67.57
6%	28.33	66.2	4.71	68.8	12.7	68.1	15.25	67.7
8%	14.62	69.2	7.2	69.7	4.55	68.5	8.79	69.13
10%	7.3	68.6	8.65	70.4	7.1	68.6	8.79	69.2
20%	3.1	72.1	3	73.3	3.3	72.15	3.13	72.52
Mean	35.3	67.39	23.23	69.26	18.42	68.37		

4.1. Selecting parameters

We draw on the work of kiril soloveg* et al. [29]. To improve the sampling success as well as the efficiency of sampling by the ordered radius induced in $r_n = \gamma(\frac{log(n)}{n})^{\frac{1}{d}}$, where $\frac{log(n)}{n}$ is a function of sampling dispersion and decreases with the increase in the number of sampled points, d is the dimension of the configuration space, and γ is an environment-dependent parameter, and n is the quantity of samples in the sampling region.

Simulations were carried out in environment A shown in Fig. 10 to select the appropriate sampling step size as well as the target sampling rate for BI-RRT* before comparing it with other algorithms. Setting the appropriate sampling step size and target sampling probability is important because while a higher sampling step size can improve the time that BI-RRT* can take to solve, it can also increase the length of the path, resulting in non-optimal path selection. We have tested Informed-RRT* against BI-RRT* at different sampling step sizes and target point sampling rates to obtain Table 1 and 2. We normalize the sampling step lengths in the sampling space using unit sampling steps of 1%, 2%, 4%, 6%, 8%, 10%, and 20%. The solutions generated using the BI-RRT* algorithm have on average a 14.08% decrease in planning duration when compared to the Informed-RRT*, faster acquisition of the initial solution, and faster convergence due to the use of a bidirectional search approach. The performance of robot navigation is typically impacted by the initial solution, as the robot will follow the initial solution in an environment where there may be two or more possible paths with different costs to reach the target. It is therefore essential to discover the initial solution at a reduced cost, and in the simulated environment, we found that when sampling at larger sampling steps, the final path length is obtained in a slightly longer, although the solution can be obtained in a shorter time. When analyzing the initial solution generated by BI-RRT*, we found that using a target sampling rate of 20% and a sampling step size of 4% took, on average, 55.28% less time than Informed-RRT* to get closer to the optimal solution, for essentially the same initial path length. In addition, the initial solution was found in only 48% of the total time. The value of the radius r_n also satisfies this theoretical range of values. As the Table 2 shows, it is recommended that the

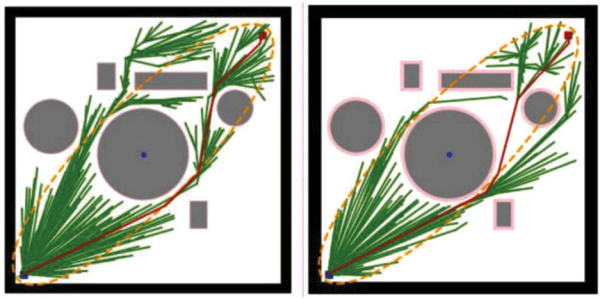


Fig. 11. Simulation in environment C.

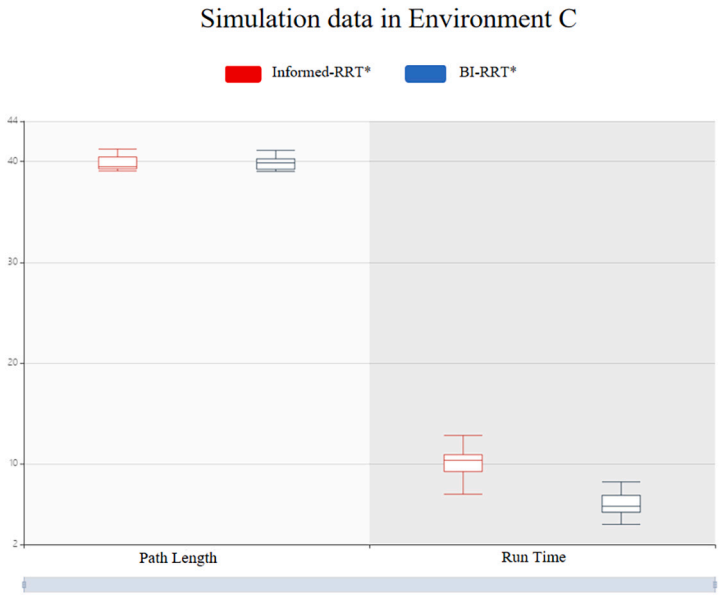


Fig. 12. Simulation Data in Environment C.

Table 3
Average Calculating costs in environment C.

Algorithm	Average runtime	Average Pathlength
Informed-RRT*	10.12	39.86
BI-RRT*	6.05	39.82

sampling step size of 4% and the target sampling rate of 20% be used for practical applications of more complex problems. Based on this conclusion [30], an analytical comparison with other algorithms was carried out using the above parameters.

4.2. Environment C

In the more basic simulated setting, the simulated path outcomes are displayed in the Fig. 11(a). The simulation data used to calculate the initial solution for each algorithm includes the mean execution time and mean route length for the 30 experiments Fig. 11(b). In the Table 3, the average length of the proposed 30 experiments is 10.12 and the average run time is 6.05. Compared to the informed-RRT* algorithm, the generated trajectories are on average 0.1% smaller and the average run time is 40.22% smaller. It is clear from the Fig. 12 that although the route length of the algorithm in this paper is essentially the same as that of informed-RRT, the time is reduced. The outcomes indicate that the algorithm reduces the cost of planning paths in a simple environment and reduces the planning time by 40.22%.

Table 4
Average Calculating costs in environment B.

Algorithm	Average runtime	Average Pathlength
Informed-RRT*	32.31	150.45
BI-RRT*	27.07	147.31

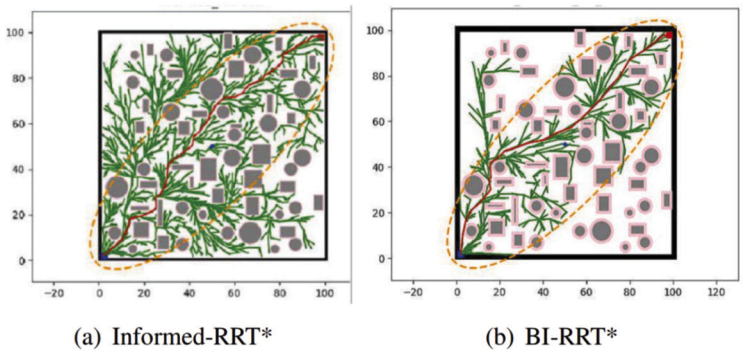


Fig. 13. Simulation in environment B.

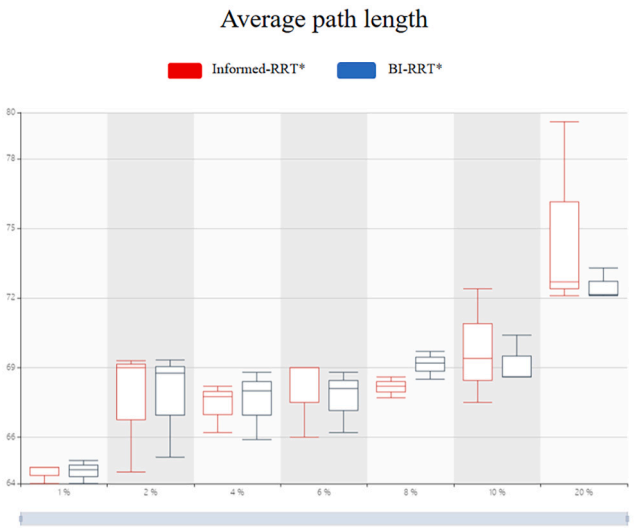


Fig. 14. Simulation Data in Environment B.

4.3. Environment B

Compared to environments A and C, environment B has more obstacles of different sizes and is a more complex environment, making it more difficult to plan path trajectories. However, the algorithm presented in this paper has the capability to still plan paths in complex environments more efficiently and with significant advantages Fig. 13(a). The simulation is shown in Fig. 13(b).

The suggested approach transmits fewer repetitive points, shorter paths and shorter planning times. Analyzing the Table 4, the average path length of the algorithm in this paper is 147.31 and the average running time is 27.07. After testing the initial solution 30 times in environment B, the average path is reduced by 2.10%, and the average running time is reduced by 16.21%. By comparing Fig. 14, the algorithm in this paper does not increase the running time as a result, although it uses the initial solution for ellipse sampling, and these data also indicate an improvement in the performance metrics of the algorithm. Therefore the path designed by the algorithm in this paper is superior.

4.4. Discussion

In this paper we propose an informed-RRT* based algorithm. Using a bi-directional search method, trajectory planning is performed for the mobile agent. The algorithm quickly builds the initial solution by using a bidirectional search tree, using the initial solution we build an elliptical sampling space, we inflate the obstacle space in order to avoid viewing the mobile agent as a prime,

Fig. 15. Overview of the car.

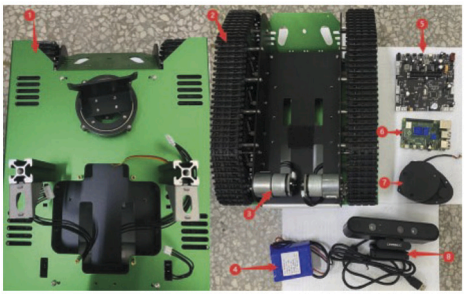


Fig. 16. Overview of the car.

Table 5
Hardware Description.

Sequences	Description
1	Tank frame
2	Tank tracks
3	ASLONG-JGB37-520 Miniature
	DC Geared Motor 12 V 1280rpm
4	4400 mAh Li-ion battery pack
5	Bottom control yahboom's expansion board
6	Raspberry Pi 4b+
7	Slam A1 Lidar
8	Astra Pro depth camera

and finally the planned path is smoothed using the cubic spline method. When compared to other algorithms, the benefit of the BI-RRT* algorithm employing a two-way search approach is its ability to plan a less expensive and smoother path in less time, which can better meet the mobile agent's movement requirements. Simultaneously, the algorithm presented in this paper has certain constraints. It only investigates the problem of trajectory planning for a single mobile agent in a 2D environment, and does not consider path planning in dynamic obstacle space. For the problem of single mobile agent and multi-body mobile agent cooperation in 3D environment, it will be our next research topic.

5. Real world case

The hardware platform utilized in this experiment is the Yahboom-Robot mobile robot. The overall body structure is shown in Fig. 15. The hardware composition is shown in Fig. 16.

See Table 5 for detailed description. In this paper, the improved algorithm is ported to the Yahboom-Robot mobile robot hardware platform. To implement the algorithm porting, the original navigation algorithm of the ROS function package was rewritten. In addition, the AES algorithm has been incorporated to encrypt sensitive data, ensuring confidentiality during data transmission and storage. This approach allows for the benefits of swift advancement and closer integration with other elements of the algorithm. The initial and destination nodes are established in the physical environment. The robot uses the improved informed RRT* algorithm to plan a collision-free path to the target location. This experimental procedure consists of three steps:

1) improved informed RRT* algorithm porting;
2) simultaneous localization and mapping (SLAM) testing;

Fig. 17. Environment Introduction.

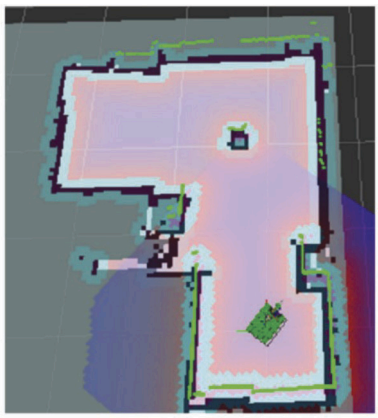


Fig. 18. Overview of the car.

3) autonomous robot navigation testing.

Algorithm porting writes the Informed RRT* algorithm into the ROS robot. SLAM uses the radar on the robot to construct a test map for real-world movement and navigation. Autonomous navigation tests verify the efficiency of the algorithm in real-world scenarios. This section describes the implementation of an autonomous robot navigation experiment to validate the algorithm's efficacy in practical situations by means of algorithm adaptation and map creation. The real-world testing environment Fig. 17(a) and the map created by SLAM are shown in Fig. 17(b). The feasible space after the expansion of the barrier is shown in Fig. 18. The cyan part is the result of the expansion of the obstacles according to the body radius, and the pink part is the feasible domain of the mobile robot. As shown in Fig. 19, the cart plots a smooth route from the initial location to the destination point.

The experimental results show that the robot can autonomously plan a dependable and seamless route to achieve self-guided navigation from the initial node to the destination node. This experiment confirms the applicability of the enhanced algorithm for mobile robots.

6. Conclusion

A path planning algorithm for mobile robots with BI-RRT* is proposed. The algorithm uses three strategies, namely extended distance, bidirectional search and smoothing, to overcome the limitations of traditional sampling algorithms in regards to path resilience and route planning effectiveness. Firstly, extended distances are configured for obstacles in the map to guarantee an error-tolerant distance between the path and the obstacle. Secondly, the initial solution is quickly established using a bidirectional search method, and the sampling space is constrained using the initial solution. Thirdly, the paths are smoothed and optimized.

The effectiveness and optimality of the algorithm are confirmed in two distinct environments and compared against alternative algorithms. The simulation outcomes demonstrate that the algorithm still presents notable benefits in intricate environments. By limiting the sampling space of random points, the algorithm greatly reduces CPU running time. Additionally, it generates high-quality new nodes, enabling them to converge towards the target point faster than Informed-RRT*. The effectiveness of the proposed algorithm, BI-RRT*, is verified through a comparison with other algorithms in numerical simulations. The algorithm can be applied to path planning for mobile robots, especially in scenarios where obstacles need to be avoided, safe distances maintained and efficient

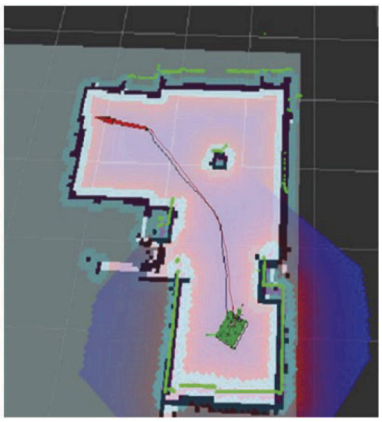


Fig. 19. Overview of the car.

paths planned. For example, areas such as self-guided vehicles in industrial automation, path planning for drones, and path planning for warehouse robots can benefit from the algorithm. In addition, the algorithm can be applied to path planning in intelligent transportation systems and in scenarios that require robots to navigate autonomously in complex environments.

While the BI-RRT* algorithm enables a more cost-effective initial solution and quicker convergence compared to existing sampling-based planning algorithms, no further consideration is given to the dynamic obstacle space, which will be the direction of our future topic. In practical applications, we will also pay further attention to the security, privacy, and reliability of trusted AI systems, ensuring the stability and feasibility of the algorithm in compliance with stringent safety standards and regulations.

CRediT authorship contribution statement

Honghui Fan: Methodology, Funding acquisition, Formal analysis, Conceptualization. **Jiahe Huang:** Resources, Methodology, Funding acquisition. **Xianzhen Huang:** Validation, Supervision, Software. **Hongjin Zhu:** Writing – original draft, Visualization, Validation, Supervision. **Huachang Su:** Writing – review & editing, Investigation, Data curation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Edsger W. Dijkstra, A note on two problems in connexion with graphs, in: Edsger Wybe Dijkstra: His Life, Work, and Legacy, 2022, pp. 287–290.
[2] Jonathan D. Gammell, Siddhartha S. Srinivasa, Timothy D. Barfoot, Batch informed trees (bit*): sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs, in: 2015 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2015, pp. 3067–3074.
[3] Peter E. Hart, Nils J. Nilsson, Bertram Raphael, A formal basis for the heuristic determination of minimum cost paths, IEEE Trans. Syst. Sci. Cybern. 4 (2) (1968) 100–107.
[4] Yongjun Ren, Ding Huang, Wenhai Wang, Xiaofeng Yu, Bsmd: a blockchain-based secure storage mechanism for big spatio-temporal data, Future Gener. Comput. Syst. 138 (2023) 328–338.
[5] Sertac Karaman, Matthew R. Walter, Alejandro Perez, Emilio Frazzoli, Seth Teller, Anytime motion planning using the rrt, in: 2011 IEEE International Conference on Robotics and Automation, IEEE, 2011, pp. 1478–1483.
[6] Sebastian Klemm, Jan Oberländer, Andreas Hermann, Arne Roennau, Thomas Schamm, J. Marius Zollner, Rüdiger Dillmann, Rrt*-connect: faster, asymptotically optimal motion planning, in: 2015 IEEE International Conference on Robotics and Biomimetics (ROBIO), IEEE, 2015, pp. 1670–1677.
[7] In-Bae Jeong, Seung-Jae Lee, Jong-Hwan Kim, Quick-rrt*: triangular inequality-based implementation of rrt* with improved initial solution and convergence rate, Expert Syst. Appl. 123 (2019) 82–90.
[8] Marlin P. Strub, Jonathan D. Gammell, Advanced bit (abit): sampling-based planning with advanced graph-search techniques, in: 2020 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2020, pp. 130–136.
[9] Louis Petit, Alexis Lussier Desbiens, Rrt-rope: a deterministic shortening approach for fast near-optimal path planning in large-scale uncluttered 3d environments, in: 2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC), IEEE, 2021, pp. 1111–1118.
[10] Bin Liao, Fangyi Wan, Yi Hua, Ruirui Ma, Shenrui Zhu, Xinlin Qing, F-rrt*: an improved path planning algorithm with improved initial solution and convergence rate, Expert Syst. Appl. 184 (2021) 115457.
[11] Sertac Karaman, Emilio Frazzoli, Sampling-based algorithms for optimal motion planning, Int. J. Robot. Res. 30 (7) (2011) 846–894.
[12] Yongjun Ren, Fujian Zhu, Jin Wang, Pradip Kumar Sharma, Uttam Ghosh, Novel vote scheme for decision-making feedback based on blockchain in Internet of vehicles, IEEE Trans. Intell. Transp. Syst. 23 (2) (2021) 1639–1648.
[13] Steven LaValle, Rapidly-exploring random trees: a new tool for path planning, Research Report 9811, 1998.
[14] Kiril Solovey, Michal Kleinbort, The critical radius in sampling-based motion planning, Int. J. Robot. Res. 39 (2–3) (2020) 266–285.
[15] Jiabo Feng, Weijun Zhang, An efficient rrt algorithm for motion planning of live-line maintenance robots, Appl. Sci. 11 (22) (2021) 10773.
[16] Jonathan D. Gammell, Siddhartha S. Srinivasa, Timothy D. Barfoot, On recursive random prolate hyperspheroids, arXiv preprint arXiv:1403.7664, 2014.
[17] Zhuping Wang, Yunsong Li, Hao Zhang, Chun Liu, Qijun Chen, Sampling-based optimal motion planning with smart exploration and exploitation, IEEE/ASME Trans. Mechatron. 25 (5) (2020) 2376–2386.

[18] Reza Mashayekhi, Mohd Yamani Idna Idris, Mohammad Hossein Anisi, Ismail Ahmedy, Ihsan Ali, Informed rrt*-connect: an asymptotically optimal single-query path planning method, IEEE Access 8 (2020) 19842–19852.
[19] Deepesh Toshniwal, Hendrik Speleers, Thomas J.R. Hughes, Smooth cubic spline spaces on unstructured quadrilateral meshes with particular emphasis on extraordinary points: geometric design and isogeometric analysis considerations, Comput. Methods Appl. Mech. Eng. 327 (2017) 411–458.
[20] Jian-ao Lian, Fast algorithms for generating parametric cubic spline interpolation curves, in: 2022 5th International Conference on Information and Computer Technologies (ICICT), IEEE, 2022, pp. 100–109.
[21] Yongjun Ren, Jian Qi, Yepeng Liu, Jin Wang, Gwang-Jun Kim, Integrity verification mechanism of sensor data based on bilinear map accumulator, ACM Trans. Internet Technol. 21 (1) (2021) 1–19.
[22] Qiang Wu, Zhaoyang Han, Ghulam Mohiuddin, Yongjun Ren, Distributed timestamp mechanism based on verifiable delay functions, Comput. Syst. Sci. Eng. 44 (2) (2023).
[23] Jonathan D. Gammell, Siddhartha S. Srinivasa, Timothy D. Barfoot, Informed rrt*: optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic, in: 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2014, pp. 2997–3004.
[24] Huanwei Wang, Shangjie Lou, Jing Jing, Yisen Wang, Wei Liu, Tieming Liu, The ebs-a* algorithm: an improved a* algorithm for path planning, PLoS ONE 17 (2) (2022) e0263841.
[25] Hewen Tao, Zhang Yi, Zhao Xiang, Research on Path Planning of Mobile Robot Based on Improved rrt* Algorithm, 2022 IEEE 6th Information Technology and Mechatronics Engineering Conference (ITOEC), vol. 6, IEEE, 2022, pp. 666–670.
[26] Michael Otte, Nikolaus Correll, C-forest: parallel shortest path planning with superlinear speedup, IEEE Trans. Robot. 29 (3) (2013) 798–806.
[27] Yongjun Ren, Yan Leng, Yaping Cheng, Jin Wang, Secure data storage based on blockchain and coding in edge computing, Math. Biosci. Eng. 16 (4) (2019) 1874–1892.
[28] Jiankun Wang, Baopu Li, Max Q.-H. Meng, Kinematic constrained bi-directional rrt with efficient branch pruning for robot path planning, Expert Syst. Appl. 170 (2021) 114541.
[29] Jiale Hou, Zhitao Liu, Hongye Su, Obstacle based fast marching tree for global motion planning, in: IECON 2022–48th Annual Conference of the IEEE Industrial Electronics Society, IEEE, 2022, pp. 1–6.
[30] Yongjun Ren, Yan Leng, Jian Qi, Pradip Kumar Sharma, Jin Wang, Zafer Almakhadmeh, Amr Tolba, Multiple cloud storage mechanism based on blockchain in smart homes, Future Gener. Comput. Syst. 115 (2021) 304–313.