IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 30 March 2024, accepted 16 April 2024, date of publication 24 April 2024, date of current version 1 May 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3392926

RESEARCH ARTICLE

Agile-RRT*: A Faster and More Robust Path Planner With Enhanced Initial Solution and Convergence Rate in Complex Environments

CHUANRONG HUANG, BINGJIE TANG, ZHIYANG GUO, QI SU, AND JINGYAO GAI
College of Mechanical Engineering, Guangxi University, Nanning 530004, China

Corresponding author: Jingyao Gai (jygai@gxu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Award U23A20330; in part by the Specific Research Project of Guangxi for Research Bases and Talents under Award AD22035919; in part by the Innovation Project of Guangxi Graduate Education under Award YCSW2023053; and in part by the Modern Industry School of Subtropical Intelligent Agricultural Machinery and Equipment, Guangxi University, China, under Project T3010097930.

ABSTRACT Path planning is a critical process in mobile robot navigation. Sampling-based path planning algorithms represented by Rapidly Exploring Random Tree star (RRT*) have gained widespread adoption due to their asymptotic optimality and proven efficiency. However, when applied to intricate environments, characterized by narrow passages and cluttered obstacles, these algorithms encounter challenges in both the initial solution generation and the convergence towards the optimal path, mainly caused by the inefficient sampling strategy, thereby impeding its overall effectiveness. To address these limitations, we introduced Agile-RRT* (A-RRT*), an advancement of RRT* algorithm. Our contributions are twofold: firstly, we introduce an adaptive goal-biased sampling strategy, which employs an adaptive principle for determining the step size on the basis of the goal-biased strategy. This avoids getting trapped in local minima and enhances the efficiency of the initial solution generation. Secondly, we introduce a path optimization approach using a secondary tree and subset-informed sampling, to accelerate the convergence toward the optimal path. It optimizes the path by gradually shrinking the designed elliptical planning space around local states, which effectively narrows down the search space. The experimental results demonstrated that the proposed A-RRT* diminishes the initial solution search time by 71.00% and the sub-optimal solution search time by 82.86% in comparison to RRT*. The A-RRT* exhibits superior performance over RRT*, Informed-RRT*, P-RRT* and Quick-RRT* in terms of soundness and efficiency in narrow and intricate environments. This method could expedite efficient motion planning for drones and mobile robots in complex environments.

INDEX TERMS Path planning, RRT*, goal-biased strategy, informed sampling, complex environment.

I. INTRODUCTION

Unmanned robots have gained substantial attention owing to their high efficiency and robust maneuverability. Path planning is one of the elementary technologies of unmanned robots, whose goal is to search for a collision-free path within a map given an initial state and a goal state. It has widespread applications not only in robot navigation, but also in medical and graphical animation applications [1]. For robots operating in narrow and intricate environments, planning a collision-free path can greatly reduce unmanned robots' energy consumption and hardware burden, but remains a long-standing challenge [2], [3].

Path planning algorithms are generally divided into three groups: artificial potential field (APF)-based, grid search-based and sampling-based algorithms. Among these algorithms, sampling-based algorithms are the most popular due to their probabilistic completeness, meaning an algorithm

The associate editor coordinating the review of this manuscript and approving it for publication was Zheng Chen.

will eventually find a feasible path unless no feasible path exists. And they avoid the challenge of constructing a discretized configuration space which indicates less memory cost [4]. Many algorithms were developed including Expansive Space Trees (EST) [5], Probabilistic Roadmap (PRM) [6] and Rapidly-exploring Random Trees (RRT) [7], etc. Among these, an advancement of the RRT algorithm, called Rapidly-exploring Random Trees star (RRT*) [8], was most adopted due to its asymptotic optimality and efficiency in path planning in regular environments. But for navigation in narrow and cluttered environments, the sampling-based algorithms generally suffer from low efficiency, appear as the slow initial solution generation and the slow convergence towards the optimal path.

For initial solution generation, one of the key factors affecting the performance is the random sampling strategy, since many sampled states are eventually discarded with random sampling. The goal-biased sampling strategy is more effective than random sampling. There are relative researches using goal-biased strategy in path planning such as heuristically-guided RRT (hRRT) [9], Potential Function Based-RRT* (P-RRT*) [10], Improved Potential Function Based-RRT* (PF-RRT*) [11] and Bi-directional APF-RRT* [12]. As a representative variant of the RRT* algorithm, P-RRT* iteratively shifts the randomly sampled states with an attractive potential field which improves the goal-reaching probability and avoids collisions. Nevertheless, most algorithms with goal-biased sampling strategy may easily get trapped during path planning in narrow and cluttered environments unless the fixed step size λ and iteration numbers are selected carefully [1].

For the convergence toward the optimal path, different path optimization approaches have been studied to improve the performance. Representative algorithms including RRT*-Smart [13], Triangular Geometerized-RRT* (TG-RRT*) [14], Quick RRT* [4], F-RRT* [15] and sampling-SEE RRT* [16]. To find cheaper paths, nodes are either generated around the initial path or removed to adjust the connection relationships between them. Methods like F-RRT* [15] use dichotomy to create a new node, while others such as RRT*-Smart [16], sampling-SEE RRT* [16] and TG-RRT* [14] generate new nodes through heuristic sampling. Reducing nodes primarily uses the triangle inequality, such as RRT*-Smart [13] and Quick RRT* [4]. During path optimization, the informed sampling strategy [17], [18], [19]has garnered significant attention for generating new nodes around the initial path. It samples directly within a hyperspheriod area which greatly reduces the exploring spaces. However, when the optimal solution has a high cost, the size of the region's area is affected and becomes larger, resulting in the algorithm's convergence speed being slow. Moreover, it fails to plan a path when the hyperspheriod area is larger than the configuration space [20].

In this paper, a robust sampling-based path planning algorithm called Agile-RRT*(A-RRT*) is proposed. It tackles the problem of existing variants of RRT* that suffer from slow initial solution generation and slow convergence towards the optimal path in intricate environments. This study mainly makes two key contributions: Firstly, we proposed an adaptive goal-biased sampling strategy for initial solution generation. This strategy formulates an adaptive policy for adjusting the step size of shifting randomly sampled states, which prevents the path planner from getting trapped while still benefiting from the advantages of the original goal-biased sampling strategy. Secondly, we introduced a path optimization approach using a secondary tree and subset-informed sampling. This method optimizes the path by gradually shrinking the designed elliptical planning space around local states. Compared to informed sampling, this approach further narrows down the sampling range, leading to significant improvements in convergence rates and the soundness.

The subsequent sections of this manuscript are outlined as follows. Section II introduces the definition of the problem and several off-the-shelf algorithms including RRT*, Informed-RRT*, P-RRT* and Quick-RRT*. Section III presents the details of the proposed algorithm A-RRT*. In Section IV, the experimental design process and corresponding results are presented. Finally, Section V makes a conclusion of this paper.

II. BACKGROUND

This section formulates the path planning problems, and subsequently provides a formal exposition of RRT*, along with the Informed Sampling Strategy used in Informed-RRT* and the path optimization approach applied in Quick-RRT*.

A. PROBLEM DEFINITION

Let $X = \mathbb{R}^d$ define set of all states in the configuration space, where $d \in N$, and $d = 2$ in this paper. Let $X_{obs} \subseteq X$ represent the set of states collision with obstacles, and $X_{free} = X \backslash X_{obs}$ denote set of the permissible states in the collision-free space.

A path planning problem is defined by a triplet (X, x_{init}, X_{goal}). X_{goal}, as defined by the planning problem, is the goal region centered at the target state x_{goal} with a radius of r_{goal}.

$$X_{goal} = \{x \in X_{free} | \|x - x_{goal}\| \leq r_{goal}\} \tag{1}$$

A path is defined as a continuous function $\sigma : [0, 1] \mapsto \mathbb{R}^d$ of bounded variation. If σ is a collision-free path, it has $\sigma(\tau) \in X_{free}, \forall \tau \in [0, 1]$. Moreover, σ is a feasible path if it is collision-free, and satisfied $\sigma(0) = x_{init}$ and $\sigma(1) \in X_{goal}$.

Define $\sum$ as the set of all paths, and let $\sum_{fea}$ denote the subset of all the feasible paths. Let $c(\sigma) : \sum_{fea} \mapsto \mathbb{R} > 0$ be the cost function, i.e. the total length in Euclidean space, of a feasible path connecting the initial state x_{init} to the goal region X_{goal}.

Therefore, the optimal path planning problem can be formulated as: Given a path planning problem (X, x_{init}, X_{goal}) and a cost function $c(\sigma)$, find an optimal path $\sigma^* \in \sum_{fea}$ by minimizing the cost function $c(\sigma)$, which could be expressed as $\sigma^* = argmin_\sigma \{c(\sigma)\}$.

Algorithm 1 RRT*

Input: $x_{init}, X_{goal}, \delta, r_{near}, X, n$
Output: $G = (V, E)$
$V \leftarrow \{x_{init}\}$, $E \leftarrow \emptyset$, $G \leftarrow (V, E)$;
for $i \leftarrow 0$ *to* n **do**
  $x_{rand} \leftarrow RandomSample(i)$;
  $x_{nearest} \leftarrow Nearest(G = (V, E), x_{rand})$;
  $x_{new} \leftarrow Steer(x_{nearest}, x_{rand}, \delta)$;
  if $CollisionFree(x_{nearest}, x_{new})$ **then**
    $X_{near} \leftarrow Near(G = (V, E), x_{new}, r_{near})$;
    $V \leftarrow V \cup \{x_{new}\}$
    $x_{parent} \leftarrow BestParent(X_{near}, x_{new}, x_{nearest})$;
    $E \leftarrow E \cup \{x_{parent}, x_{new}\}$;
    $G \leftarrow Rewire(G = (V, E), x_{new}, X_{near})$;
  end if
end for
return $G = (V, E)$;

B. RRT*

RRT* is an optimal path planning algorithm. The algorithm's pseudocode is presented in Algorithm 1. The algorithm explores to reach the goal region X_{goal} from the initial state x_{init} through a tree $G = (V, E)$, where V is the set of nodes representing the states $x \in X_{free}$ in the collision-free space, and E is the set of edges $e \in X_{free} \times X_{free}$ which represent the connectivity of two states in the tree. The tree iteratively expands by generating new nodes and edges through randomly sampling in the configuration space. Rewiring is applied to each new node of the tree in each iteration to minimize the cost of the nearby vertices. A feasible path is generated once the goal region is reached. The relevant functions employed in RRT* include:

RandomSample: uniformly sample a random state x_{rand} in the configuration space X;

Nearest: given a random sampling state $x \in X_{free}$ and a random tree $G = (V, E)$, traverse the tree for the state x_{nest} nearest to x;

Steer: given two states $x_1, x_2 \in X_{free}$, create a new state x_{new} by advancing a distance δ from state x_1 toward x_2;

CollisionFree: given two states $x_1, x_2 \in X$, verify whether there exist obstacles between the two states;

Near: given a state $x \in X_{free}$ and a tree $G = (V, E)$, traverse the tree and return a set of states within a circular area centered at x with a radius of r_{near};

Cost: given a state $x \in X_{free}$, traverse the tree and compute the cost from x_{init} to x;

BestParent: given a set of states X_{near} and two states x_{nest} and x_{new}, return a state x_p that minimize the cost of x_{new} when it transitions from $\{X_{near} \bigcup x_{nest}\}$ to become its parent. The pseudocode is shown in Algorithm 2;

Parent: given a state $x \in X_{free}$, traverse the tree and find the parent node of x;

Rewire: given a set of states X_{near}, a tree $G = (V, E)$ and a new state x_{new}, when x_{new} is added to the tree $G = (V, E)$, this function searches for states in X_{near} in the tree that could be connected to this new state to create a path with lower cost. The detailed explanation is expressed in Algorithm 3.

The RRT* has two main parameters, including a constant step size δ in the Steer function, and a radius r_{near} in the Near function.

Algorithm 2 BestParent

Input: $X_{near}, x_{new}, x_{nearest}$
Output: x_{parent}
$x_{parent} \leftarrow x_{nearest}$;
$Cost(x_{new}) \leftarrow Cost(x_{nearest}) + ||x_{nearest} - x_{new}||_2$;
for $x_{near} \in X_{near}$ **do**
  if $CollisionFree(x_{near}, x_{new})$ **then**
    if $Cost(x_{near}) + ||x_{near} - x_{new}||_2 < Cost(x_{new})$ **then**
      $x_{parent} \leftarrow x_{near}$;
      $Cost(x_{new}) \leftarrow Cost(x_{near}) + ||x_{near} - x_{new}||_2$;
    end if
  end if
end for
return x_{parent};

Algorithm 3 Rewire

Input: $G = (V, E)$, x_{new}, X_{near}
Output: $G = (V, E)$
for all $x_{near} \in X_{near}$ **do**
  if $CollisionFree(x_{near}, x_{new})$ **then**
    if $Cost(x_{new}) + \|x_{near} - x_{new}\|_2 < Cost(x_{near})$ **then**
      $E \leftarrow E \backslash \{x_{near}, Parent(x_{near})\}$;
      $Parent(x_{near}) \leftarrow x_{new}$;
      $E \leftarrow E \cup \{x_{near}, Parent(x_{near})\}$;
      $Cost(x_{near}) \leftarrow Cost(x_{new}) + \|x_{near} - x_{new}\|_2$;
    end if
  end if
end for
return $G = (V, E)$;

C. INFORMED-RRT*

Informed-RRT* is a classical extension of RRT*. Compared to random sampling, it introduces an informed sampling strategy (Algorithm 4) to reduce the sampling space which can significantly improve the convergence speed. It behaves in the same way as RRT* until an initial path is found. Once the initial path is obtained, Informed-RRT* utilizes the informed sampling strategy to sample a random state x_{rand} within a hyper-ellipsoid region, which centered at the initial state x_{init} and the target state x_{goal}. The random state x_{rand} satisfies following conditions:

$$\|x_{rand} - x_{init}\|_2 + \|x_{rand} - x_{goal}\|_2 \leq d_{best}, x_{goal} \in X_{goal} \tag{2}$$

in which d_{best} is initialized to the cost of the initial solution and changes with the cost of the best solution.

Algorithm 4 InformedSample

Input: $x_{init}, x_{goal}, c_{max}, c_{min}, X$
Output: x_{rand}
$x_{centre} \leftarrow (x_{init} + x_{goal})/2$
$C \leftarrow RotationToWorldFrame(x_{init}, x_{goal})$
$r_1 \leftarrow c_{max}/2$
$r_i \leftarrow (\sqrt{c_{max}^2 - c_{min}^2})/2$
$L \leftarrow diag\{r_1, \ldots, r_i\}$
$x_{rand} \leftarrow RandomSample(i)$
$x_{rand} \leftarrow (CLx_{rand} + x_{centre}) \cap X$
return x_{rand};

In Algorithm 4, the function *RotationToWorldFrame* returns a rotation matrix C, which transforms from the hyperellipsoid-aligned frame to the world frame. *diag* represents a diagonal matrix and L denotes a transformation matrix from the hyperellipsoid-aligned frame to the world frame. The detailed explanation of these symbols is referenced in [17]

D. QUICK-RRT*

As another extension of RRT*, Quick-RRT* is efficient in both searching initial solutions and converging to optimal solutions. Based on the triangular inequality, its main modification is to connect a further node for reducing the cost of path. Different from other path optimization algorithms, Quick-RRT* applies the triangular inequality with a parameter d in *BestParent*, and *Rewire* these two functions.

III. AGILE-RRT*

Agile-RRT* algorithm (Algorithm 5) was developed to solve the issue of time-consuming initial solutions and slow convergence rates of sampling-based path planning algorithms, which involves two improvements:

1) An adaptive goal-biased sampling strategy is used to quickly initialize a solution and avoid getting trapped.

2) A subset informed sampling strategy with a secondary tree is proposed to improve the convergence rate and success rate.

A. ADAPTIVE GOAL-BIASED SAMPLING STRATEGY

The concept of goal-biased sampling strategy is to generate a new node x'_{rand} advancing from x_{rand}, generated by *RandomSample* function, toward the goal region with a fixed step size Fig. 1(a). Since the x'_{rand} could be trapped in local minima with a fixed step size, an adaptive regulation principle was proposed for the step size τ in (3).

$$x'_{rand} = x_{rand} + \tau \cdot \frac{x_{goal} - x_{rand}}{\|x_{goal} - x_{rand}\|} \tag{3}$$

$$\tau = (1 - \frac{\|x_{goal} - x_{rand}\|}{\|x_{goal} - x_{init}\|}) \cdot \|x_{goal} - x_{rand}\| \tag{4}$$

The adaptive goal-biased strategy is based on two principles: (1) Using a smaller step size during the initial expansion of the tree to increase the chances of escaping traps, and (2) Employing a larger step size as the random nodes approach the target state to expedite the search progress. However, to avoid excessive collision checking, the step size is dynamically adjusted based on the distance between the random nodes and the target state, with a scaling coefficient ensuring that the step size remains within a reasonable range. Fig. 4 illustrates how the step size is adapted based on the position of the random node relative to the initial and goal states.

As demonstrated in Fig 2, the adaptive goal-biased strategy effectively guides random states using variable step sizes, promoting diverse tree expansion directions and facilitating escape from traps, in contrast to traditional goal-biased strategies that may lead to trapped edges being discarded.

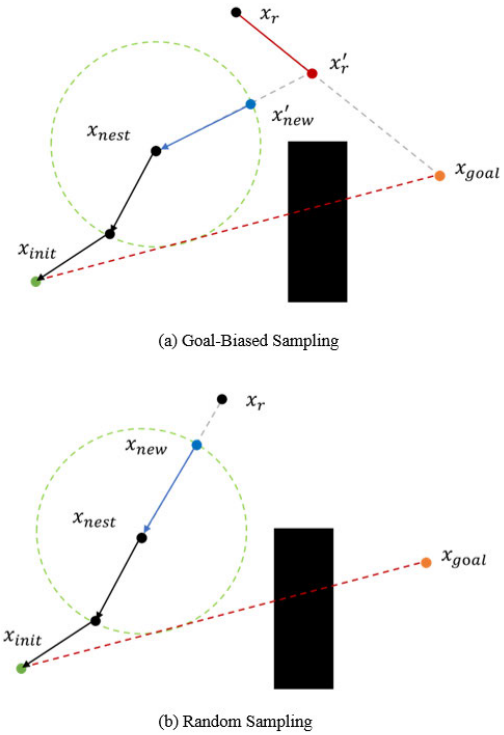


FIGURE 1. The process of *Steer* with goal-biased sampling (a) and random sampling (b). x_r represents a random node generated by *RandomSample* function. x'_r represents the generated node advancing from x_r towards goal by goal-biased sampling. x_{nest} is the nearest node to x_r as found by the *Nearest* function. x_{new} and x'_{new} are the nodes created by the *Steer* function. x_{init} denotes the initial state and x_{goal} denotes the target state.The solid red line in (a) represents the fixed step size.

B. PATH OPTIMIZATION USING A SECONDARY TREE WITH SUBSET-INFORMED SAMPLING

The dual-tree structure, which was reported effective in reducing the computation cost [21], [22], was adopted along with the subset-informed sampling strategy in this study. After the generation of the first tree representing the initial solution features the adaptive goal-biased strategy, a secondary tree is initialized with an iterative process, which

Algorithm 5 A-RRT*

1: **Input:** $x_{init}, x_{goal}, \delta, X_{goal}, r_{near}, X, n$
2: **Output:** $G2 = (V2, E2)$
3: $V1 \leftarrow \{x_{init}\}, E1 \leftarrow \emptyset, G1 \leftarrow (V1, E1)$;
4: $V2 \leftarrow \emptyset, E2 \leftarrow \emptyset, G2 \leftarrow (V2, E2)$;
// To solve the initial solution
5: **while** $i < n \wedge not\ FindInitialSolution$ **do**
6: $x_{rand} \leftarrow AdaptiveGoalBiasedSample(i)$;
7: $x_{nearest} \leftarrow Nearest(G1 = (V1, E1), x_{rand})$;
8: $x_{new} \leftarrow Steer(x_{nearest}, x_{rand}, \delta)$;
9: **if** $CollisionFree(x_{nearest}, x_{new})$ **then**
10: $X_{near} \leftarrow Near(G1 = (V1, E1), x_{new})$;
11: $V1 \leftarrow V1 \cup \{x_{new}\}$;
12: $x_{parent} \leftarrow BestParent(X_{near}, x_{new}, x_{nearest})$;
13: $E1 \leftarrow E1 \cup \{x_{parent}, x_{new}\}$;
14: $G1 \leftarrow Rewire(G1 = (V1, E1), x_{new}, X_{near})$;
15: **end if**
16: $i++$;
17: **end while**
18: $G2 \leftarrow CreateNewTree(G1 = (V1, E1), x_{goal})$;
// To optimize the solution
19: **while** $i < n$ **do**
20: $x_{rand} \leftarrow SubInformedSample(G2 = (V2, E2), X)$;
21: $x_{nearest} \leftarrow Nearest(G2 = (V2, E2), x_{rand})$;
22: $x_{new} \leftarrow Steer(x_{nearest}, x_{rand}, \delta)$;
23: **if** $CollisionFree(x_{nearest}, x_{new})$ **then**
24: $X_{near} \leftarrow Near(G2 = (V2, E2), x_{new}, r_{near})$;
25: $V2 \leftarrow V2 \cup \{x_{new}\}$;
26: $x_{parent} \leftarrow BestParent(X_{near}, x_{new}, x_{nearest})$;
27: $E2 \leftarrow E2 \cup \{x_{parent}, x_{new}\}$;
28: $G2 \leftarrow Rewire(G2 = (V2, E2), x_{new}, X_{near})$;
29: **end if**
30: $i++$;
31: **end while**
32: **return** $G2 = (V2, E2)$;

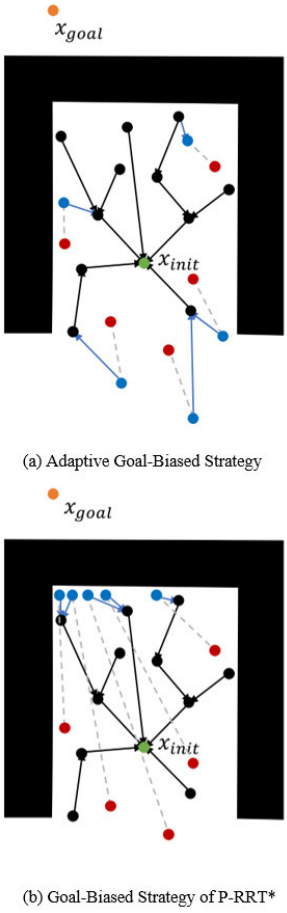


FIGURE 2. One iteration in expanding a tree with adaptive goal-biased strategy (a) and goal-biased strategy of P-RRT* (b). x_{init} denotes the initial state and x_{goal} denotes the target state. The black dots and black lines represent the nodes and edges in the tree from the previous iteration. The red dots represent the same uniformly sampled nodes generated by the *RandomSample* function. The blue dots and lines represent newly created nodes and edges in the tree by *Steer* function.

starts from the goal state of the initial solution, and reversely checks for directly connected nodes with successive parents, until no more directly connectable nodes are present. The process of initializing the secondary tree is explained in Algorithm 6, illustrated in Fig. 3.

The secondary tree was then asymptotically optimized by iteratively adding new sampled states and rewiring the secondary tree (Fig. 4). The new sampled states here are generated by subset informed sampling within a designed hyper-ellipsoid region. Algorithm 7 explains how to generate the new states.

The subset informed sampling function first finds a state $x_{nearest}$ in the secondary tree, which is nearest to x_{rand}, generated by *RandomSample* function. Then the parent state of $x_{nearest}$ called x_{parent} (Line 3), and the nearest child state of $x_{nearest}$ called x_{child} (Line 4), are chosen to be the centers of the designed hyper-ellipsoid region. The distance d_{min} from $x_{nearest}$ to line generated by x_{parent} and x_{child} is set to be the minor axes of the ellipse (Line 6).

$$d_{min} = \sqrt{(\|x_{parent} - x_{child}\|_2)^2 - d_s} \tag{5}$$

$$d_s = \frac{(x_{nearest} - x_{parent}) \cdot (x_{nearest} - x_{child})}{\|x_{parent} - x_{child}\|_2^2} \tag{6}$$

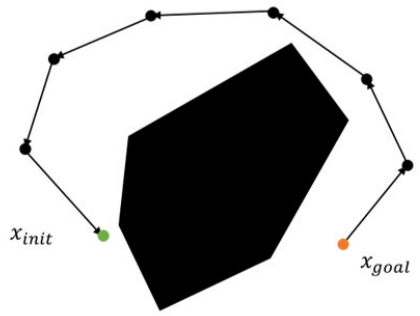
And the path length between x_{parent}, $x_{nearest}$ and x_{child} is set to be the major axes of the ellipse (Line 7). Detailed numerical equations of generating x_{rand} in Line 8 can refer [17].

The primitive functions used in Agile-RRT* are explained below:

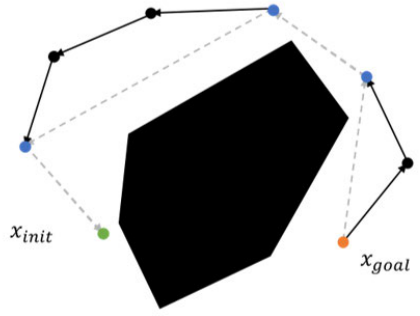
FindInitialSolution: verify if the initial feasible path has been identified or not;

AdaptiveGoalBiasedSample: uniformly sample a random state x_{rand} in the configuration space X and then return a random state x'_{rand} based on x_{rand} by applying the adaptive goal-biased samlping strategy;

NearestChild: given a state x, traverse the tree and return the nearest child of x.



(a) First Tree Representing in Initial Solution



(b) Second Tree for Path Optimization

FIGURE 3. The initialization of the secondary tree by *CreateNewTree* function from the initial solution. x_{init} denotes the initial state and x_{goal} denotes the target state. The green dot represents the initial state, the orange dot represents the goal state, and the blue dots represent the preserved nodes for the secondary tree as no more directly connectable nodes are present.

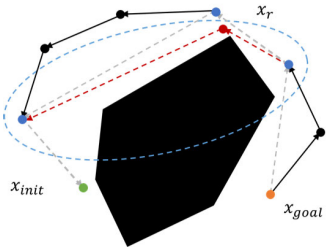


FIGURE 4. The procession of subset informed sampling procession. x_{init} denotes the initial state and x_{goal} denotes the target state. The red dot x_r represents a random state sampled by subset informed sampling method. The red dashed line represents the updated local path.

IV. SIMULATIONS

In this section, our proposed Agile-RRT* was compared with four baseline algorithms including RRT*, Informed-RRT*, P-RRT* and Quick-RRT* in terms of robustness and efficiency. Four narrow and cluttered environments were designed which consisted of a corridor environment, a narrow environment, a cluttered environment and a maze environment. These four environments are of the same size, which is 250*250 pixels (Fig. 5).

Due to the inherent randomness of these algorithms, each algorithm was repeated for 50 times for each simulation environment to ensure statistical significance. The average value of each metric was calculated. As an exception, if the algorithm failed to converge to a sub-optimal solution within

Algorithm 6 CreateNewTree

Input: $G1 = (V1, E1), x_{goal}$
Output: $G2 = (V2, E2)$
$V2 \leftarrow \{x_{goal}\}$, $E2 \leftarrow \emptyset$, $G2 \leftarrow (V2, E2)$;
$x_{pick} \leftarrow x_{goal}$;
$x_{temp} \leftarrow x_{goal}$;
$x_{desc} \leftarrow x_{goal}$;
while $x_{temp} \neq x_{init}$ **do**
  if $CollisionFree(x_{pick}, x_{temp})$ **then**
    $x_{temp} \leftarrow Parent(x_{temp})$;
    $x_{desc} \leftarrow x_{temp}$;
  else
    $V2 \leftarrow V2 \cup \{x_{desc}\}$;
    $E2 \leftarrow E2 \cup \{x_{desc}, x_{pick}\}$;
    $G2 \leftarrow G2 \cup \{(V2, E2)\}$;
    $x_{pick} \leftarrow x_{desc}$;
  end if
end while
return $G2 = (V2, E2)$;

Algorithm 7 SubInformedSample

Input: $G2 = (V2, E2), X$
Output: x_{rand}
$x_{rand} \leftarrow RandomSample(i)$
$x_{nearest} \leftarrow Nearest(G2 = (V2, E2), x_{rand})$;
$x_{parent} \leftarrow Parent(x_{nearest})$;
$x_{child} \leftarrow NearestChild(x_{nearest})$;
$d_s = (\frac{(x_{child} - x_{parent}) \cdot (x_{nearest} - x_{child})}{\|x_{parent}, x_{child})\|_2})^2$;
$c_{min} \leftarrow \sqrt{\|x_{parent}, x_{nearest})\|_2^2 - d_s}$;
$c_{max} \leftarrow \|x_{child}, x_{nearest})\|_2 + \|x_{parent}, x_{nearest})\|_2$;
$x_{rand} \leftarrow InformedSample(x_{parent}, x_{child}, c_{max}, c_{min}, X)$;
return x_{rand};

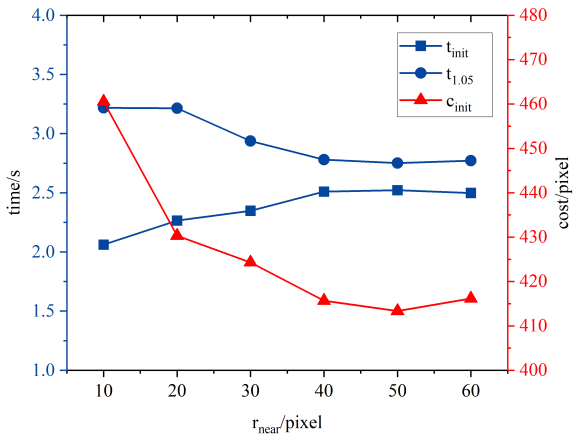


FIGURE 5. The performance of A-RRT* changes with r_{near} when δ was fixed at 5.

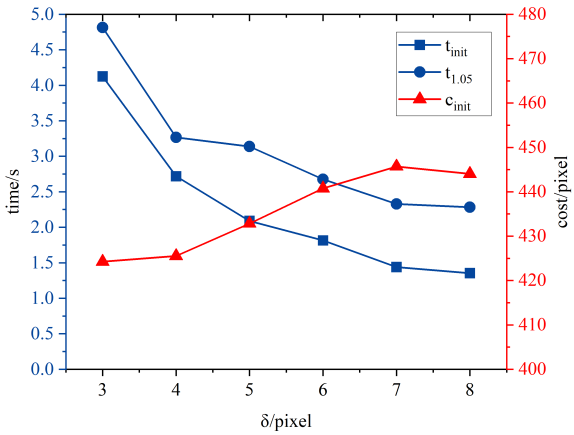


FIGURE 6. The performance of A-RRT* changes with δ when r_{near} was fixed at 20.

5 minutes, it was classified as a ''Failed path planning trail'' and was excluded from the statistical analysis. The experiments were conducted on a workstation with an Intel Core i7-10700F CPU and 16G RAM.

A. EVALUATION METRICS AND PARAMETERS SELECTION

To improve the navigation performance of a mobile robot following a path from the initial solution, the cost of the initial solution is a crucial factor [4]. Besides, the proposed algorithm utilized a secondary tree based on the initial solution. Thus, a rapidly solution initialization and a smaller initial cost enables to accelerate the optimization process. Therefore, three key metrics were employed to assess the performance of these algorithms: c_{init} denotes the cost of the initial solution; t_{init} is the time taken to generate an initial solution; and $t_{1.05}$ indicates the time required to find a sub-optimal solution of cost $1.05{*}c_{opt}$, in which c_{opt} is the cost of the theoretical optimal solution.

There are two parameters have an impact on the performance of our proposed A-RRT* and the baseline algorithms, which are the neighbor search radius r_{near} using in the Near function, and the step size δ used in Steer function. Fig. 6 and Fig. 7 present the performance of A-RRT* with different parameter settings.

Fig. 6 illustrates how the key measures are influenced by changing the value of r_{near} with a fixed step size $\delta = 5$. While $r_{near} = 10$ achieved the fastest initial solution initialization speed, the initial solution cost was the highest. Increasing $r_{near} = 20$ increased the initial solution time by 0.2 seconds compared to $r_{near} = 10$, but significantly decreased the initial solution cost by 30. Furthermore, $r_{near} = 20$ had the fastest initial solution cost reduction rate. Therefore, this paper chose $r_{near} = 20$.

Fig. 7 demonstrates the impact of varying the step size δ on the performance of A-RRT*, with a fixed neighbor search radius of 20. The figure indicated that $\delta = 3$ resulting in the lowest initial path cost, but it took the longest time to obtain an initial solution. By choosing $\delta = 4$, the initial solution time reduced by 1.4 seconds compared to $\delta = 3$. Although $\delta = 5$ increased the initial solution cost by 7 pixels compared to $\delta = 4$, it reduced the initial solution time by half compared to $\delta = 3$. Increasing parameters with $\delta > 5$ didn't significantly improve the speed of path initialization, and the initial solution cost continued to increase. Therefore, $\delta = 5$ was selected as the optimal step size in this study.

To ensure fairness in comparison, r_{near} was set to 20, and δ was set to 5 for all tested algorithms in the experiments. For the specific parameters for the baseline methods, this paper set $\lambda = 0.1$, $d^*_{obs} = 0.1$, $k = 300$ for P-RRT*, and $d = 2$ for Quick-RRT* based on our preliminary experiments.

B. EXPERIMENTAL RESULTS

1) CORRIDOR ENVIRONMENT

Fig. 8 illustrates the generated paths and explored space in corridor simulation environment for each algorithm. It is evident that both A-RRT* and P-RRT* narrowed down the searching space in generating the initial solution comparing to other baseline methods. Additionally, A-RRT* had a narrower searching space during path optimization.

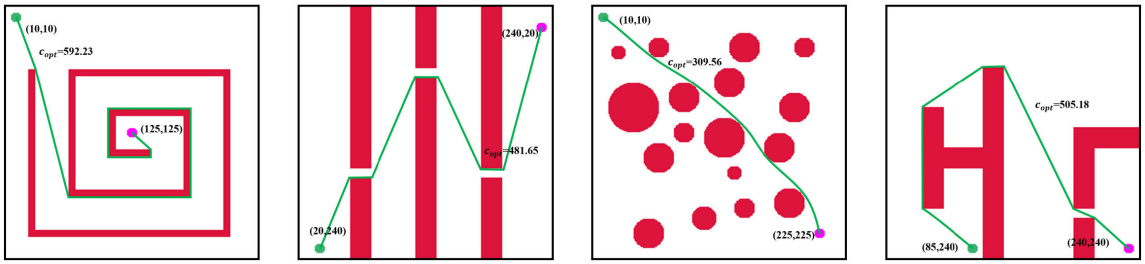
Table 1 provides statistical information that further supports these observations. It is observed that A-RRT* generated an initial solution and converged to the sub-optimal solution in the shortest time. Although Q-RRT* generated an initial path with least cost, it consumed more than 10 times of time compared to A-RRT*. This suggests that A-RRT* offers a balance between solution quality and computational efficiency.

The data presented in Table 1 also indicates that while P-RRT* discovered an initial solution with a short time, it failed to converge to a sub-optimal solution in this narrow environment for most of the tests. In contrast, A-RRT* demonstrated 41% reduction in time consumption to search an initial solution and 55% reduction in time consumption to search a sub-optimal solution compared to P-RRT*, with no failures. These results highlight the superior performance of A-RRT* in terms of speed and convergence compared to P-RRT*.

Because of the initial solution of Informed-RRT* is highly depended on RRT*, both Informed-RRT* and RRT* exhibited no significant convergence in narrow environment. Q-RRT*, informed-RRT* and RRT* are all inefficient in finding the initial solution due to the random space searching. In contrast, A-RRT* has the best success rate of the sub-optimal planning and has the best overall performance in searching initial solutions and converging to sub-optimal solutions in this scenario.

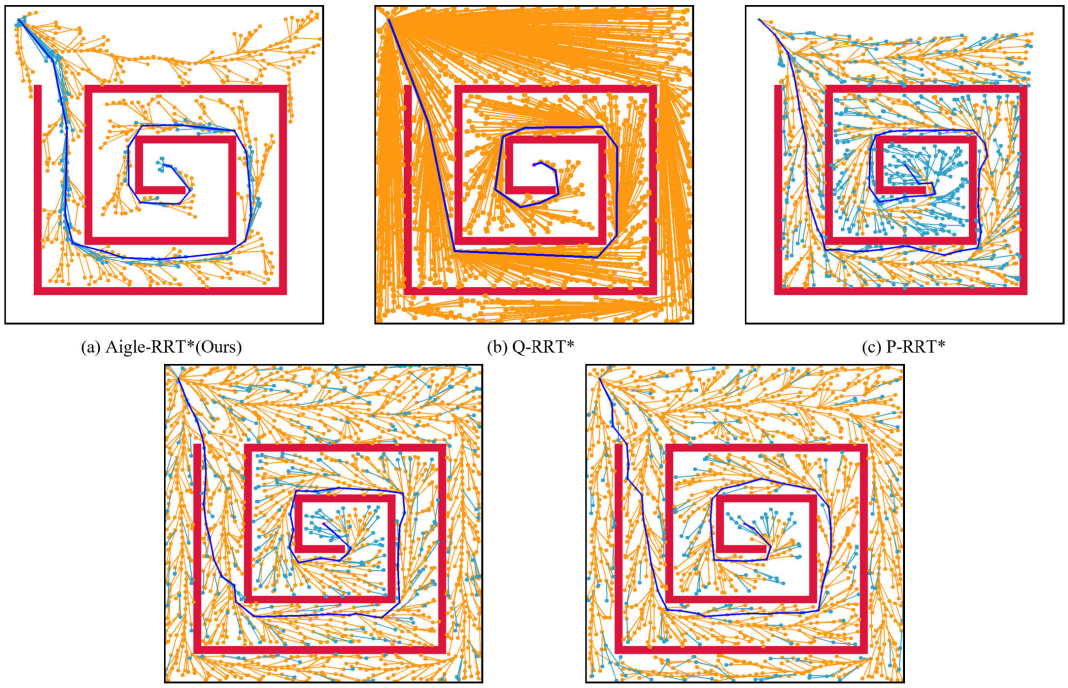
2) NARROW ENVIRONMENT

Fig. 9 displays the sub-optimal paths and explored space in the narrow environment for each algorithm. It is evident that A-RRT* took the least space in both generating the initial solution and path optimization.



(a) Corridor (b) Narrow (c) Clutter (d) Maze

FIGURE 7. The simulation environments for testing the algorithms. The green dots represent the initial states and the magenta dots are the goal states. The green line is the theoretical shortest path, and the c_{opt} is the corresponding cost.



(a) Aigle-RRT*(Ours) (b) Q-RRT* (c) P-RRT*

(d) Informed-RRT* (e) RRT*

FIGURE 8. Performance of five algorithms in the corridor environment. The orange thin lines and points represent the tree constructed for the initial solution, while the blue think lines and points represent the edges and nodes used for solution convergence. The blue thick line represents the suboptimal path.

TABLE 1. Algorithm performance in corridor environment.

Algorithm	c_{init}(pixel)	t_{init}(s)	$t_{1.05}$(s)	Failed(%)
A-RRT*(Ours)	629.14 ± 7.51	$\mathbf{14.93 \pm 2.97}$	$\mathbf{15.08 \pm 2.979}$	**0**
Q-RRT*	$\mathbf{602.76} \pm 6.23$	147.6 ± 55.13	147.6 ± 55.13	20
P-RRT*	$624.30 \pm \mathbf{3.91}$	25.12 ± 11.25	33.36 ± 15.82	72
Informed-RRT*	622.70 ± 6.74	120.4 ± 39.61	136.1 ± 39.90	84
RRT*	624.00 ± 6.15	135.5 ± 59.37	160.5 ± 70.54	76

The statistical results of the narrow environment are displayed in Table 2. It is observed that A-RRT* spent the shortest time for searching for initial solutions and sub-optimal solutions. Although Q-RRT* found the initial solution with the least cost, it spent almost twice the time, mainly spent on sampling to move through the narrow passages during initializing the solution (Fig. 9(b)) in narrow environments.

P-RRT* is efficient in finding the initial solution, but inefficient in optimizing the path and subject to failures, as it almost explored the entire space (Fig. 9(c)). Informed-RRT* and RRT* are both inefficient in finding the initial solution

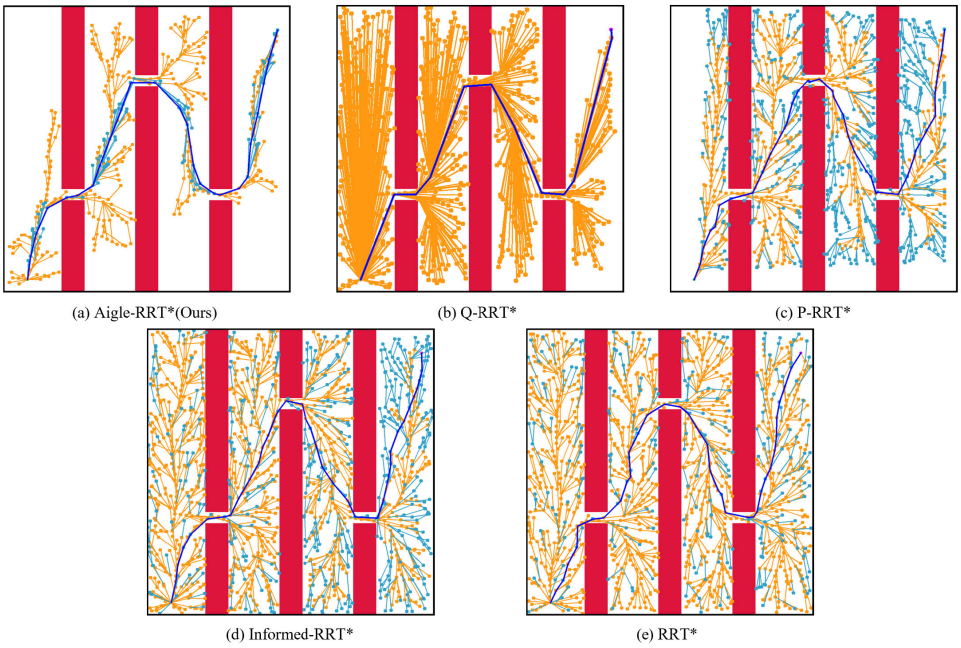


FIGURE 9. Performance of five algorithms in the narrow environment. The orange thin lines and points represent the tree constructed for the initial solution, while the blue think lines and points represent the edges and nodes used for solution convergence. The blue thick line represents the suboptimal path.

TABLE 2. Algorithm performance in narrow environment.

Algorithm	c_{init}(pixel)	t_{init}(s)	$t_{1.05}$(s)	Failed(%)
A-RRT*(Ours)	515.55 ± 5.68	$\mathbf{5.764 \pm 1.544}$	$\mathbf{6.936 \pm 1.802}$	0
Q-RRT*	$\mathbf{497.29 \pm 5.04}$	13.47 ± 4.854	13.81 ± 4.788	0
P-RRT*	515.51 ± 6.40	6.081 ± 2.341	45.05 ± 49.21	18
Informed-RRT*	513.75 ± 6.86	15.91 ± 9.529	66.75 ± 74.29	28
RRT*	516.06 ± 7.16	15.21 ± 5.739	61.74 ± 42.96	22

and path optimization, as they got trapped during searching for initial solutions, and almost explored the entire space during converging (Fig. 9(d-e)).

A-RRT* reduced 5% of the time for solving the initial solution compared to P-RRT*, and reduced 50% of the time for generating the sub-optimal solution with higher success rate compared to Q-RRT*. Therefore, A-RRT* has the best overall performance and robustness in searching initial solutions and converging to sub-optimal solutions in this scenario.

3) CLUTTERED ENVIRONMENT

Fig. 10 displays the paths generated and exploration results for five algorithms in a cluttered environment. It indicates that A-RRT* (Fig. 10(a)) had a narrower search space during both solution initialization and optimization compared to other baseline algorithms.

The statistical analysis results are presented in Table 3. The table highlights that A-RRT* generated an initial solution with the lowest cost and converged to a sub-optimal solution in the shortest time among the algorithms. A-RRT* outperformed Q-RRT* with an 8% reduction in terms of initial solution cost. It also reduced the time taken to initialize a solution by 41%, along with an impressive reduction of 70% time taken to converge to a sub-optimal solution.

Table 3 also indicates that P-RRT* took the least time for generating an initial solution and narrowed the search space. However, it is inefficient in path optimization as it explored almost the entire map (Fig. 10(c)). In comparison, A-RRT* reduced time consumption in optimizing the solution by 69% and decreased the cost to initialize a solution by 4% compared to A-RRT*.

Both RRT* and Informed RRT* are inefficient in path optimization, as they searched the entire map. A-RRT* achieved 85% and 72% time reduction in searching for optimal paths compared to RRT* and Informed-RRT*, respectively. Hence, A-RRT* has a superior performance in generating an initial solution and converging to a sub-optimal solution in this map.

4) MAZE ENVIRONMENT

Fig. 11 displays the generated paths and explored space in the maze environment for five algorithms. It indicates that A-RRT* significantly reduced the exploration space during solution initialization and path optimization compared to the other baseline methods.

The statistical result presented in Table 4 supports this observation. It manifests that A-RRT* is capable of generating initial solutions and converging to sub-optimal solutions in the shortest time. While Q-RRT* found initial solutions with the least costs, it required 49% more time to

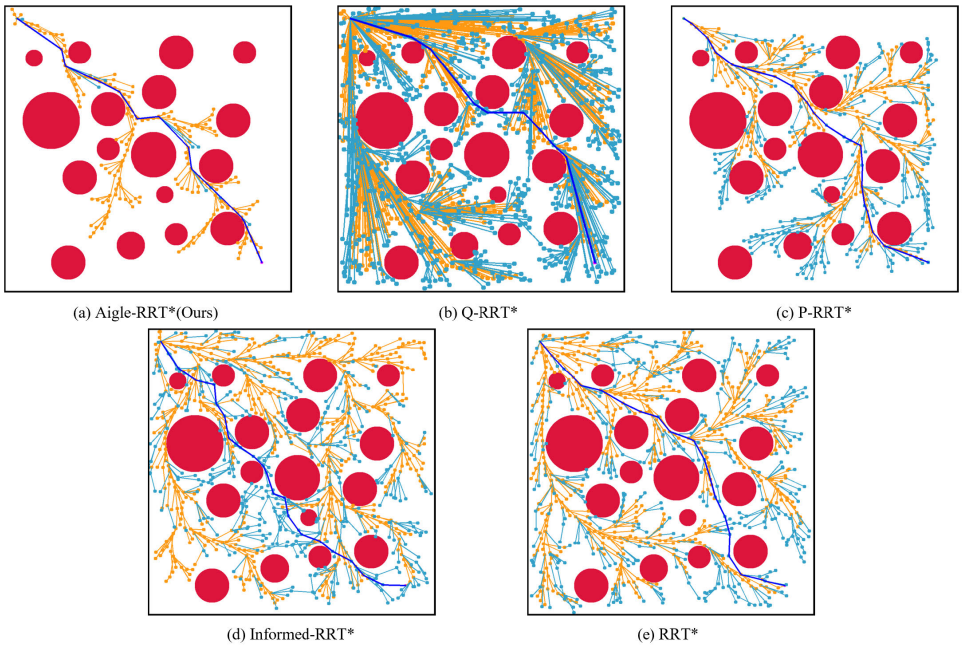


FIGURE 10. Performance of five algorithms in the cluttered environment. The orange thin lines and points represent the tree constructed for the initial solution, while the blue think lines and points represent the edges and nodes used for solution convergence. The blue thick line represents the suboptimal path.

TABLE 3. Algorithm performance in cluttered environment.

Algorithm	c_{init}(pixel)	t_{init}(s)	$t_{1.05}$(s)	Failed(%)
A-RRT*(Ours)	$\mathbf{328.44 \pm 11.26}$	1.405 ± 0.314	$\mathbf{2.285 \pm 2.359}$	0
Q-RRT*	357.37 ± 15.96	2.396 ± 0.815	7.675 ± 2.754	0
P-RRT*	342.27 ± 14.24	$\mathbf{1.046 \pm 0.201}$	7.287 ± 5.133	0
Informed-RRT*	365.77 ± 16.95	2.465 ± 0.830	8.228 ± 2.939	0
RRT*	371.40 ± 18.88	2.303 ± 0.710	15.35 ± 10.46	0

generate an initial solution and 40% more time to converge to a sub-optimal solution, as it searched randomly and extensively explored the entire map (Fig. 11(b)).

According to the information presented in Table 4, P-RRT* algorithm demonstrates a better performance in terms of the cost of initial solutions. However, this algorithm is inefficient in converging to sub-optimal solutions and tends to be trapped in this scenario (Fig. 11(c)), as it is moves directly towards the goal region with a fixed step. Moreover, it took twice as much time as A-RRT* to find an initial solution and converge to a sub-optimal solution.

Due to random sampling nature of Informed-RRT* (Fig. 11(d)) and RRT* (Fig. 11(e)), the generated nodes and edges were distributed throughout the entire space. Additionally, compared to A-RRT*, these algorithms took nearly three times longer to initialize a solution and optimize the path. Although A-RRT* consumed more cost in initial solutions, it escaped from traps and demonstrated almost a 50% reduction in solution initialization and path optimization compared to the other baseline algorithms. Therefore, A-RRT* exhibits superior performance in quickly generating an initial solution and achieving a high-speed convergence rate in this environment.

C. DISCUSSIONS

Simulations and statistics conducted across four different intricated environments have demonstrated the superior performance of A-RRT* over other algorithms in solving time-consuming initial solutions and converging towards sub-optimal solutions.

A-RRT* was found to be highly efficient in sampling during the solution initialization process, especially effective in guiding random states out of trapped. As shown in Fig. 8(a), Fig. 9(a), Fig. 10(a) and Fig. 11(a), A-RRT* samples less states and generate less edges in this narrow and intricate environment. This is due to the algorithm's adaptive goal-biased strategy. For instance, in the corridor environment, the algorithm employed a smaller step size to guide the sampling point towards the target area when it is in close proximity to the starting point. It enabled the algorithm to expand the tree in multiple directions and explore the interior of the corridor with ease. As the sampling point progressed further into the corridor, the A-RRT* algorithm guided states with a moderate step size towards the target area, but kept a longer distance with obstacles compared to P-RRT*. When the sampling point was located very close to the target, the A-RRT* algorithm directed it towards the target

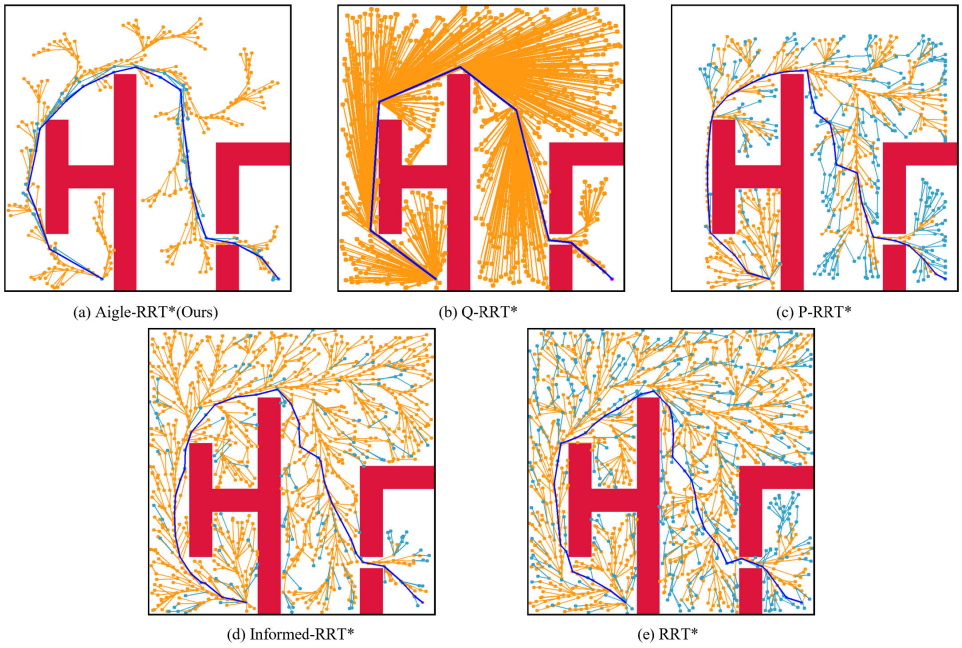


FIGURE 11. Performance of five algorithms in the maze environment. The orange thin lines and points represent the tree constructed for the initial solution, while the blue think lines and points represent the edges and nodes used for solution convergence. The blue thick line represents the suboptimal path.

TABLE 4. Algorithm performance in maze environment.

Algorithm	c_{init}(pixel)	t_{init}(s)	$t_{1.05}$(s)	Failed(%)
A-RRT*(Ours)	530.07 ± 13.30	$\mathbf{4.420 \pm 0.825}$	$\mathbf{5.182 \pm 0.863}$	0
Q-RRT*	$\mathbf{490.68 \pm 5.15}$	8.670 ± 2.464	8.678 ± 2.464	0
P-RRT*	505.74 ± 9.68	10.70 ± 7.026	12.40 ± 7.218	0
Informed-RRT*	509.30 ± 7.55	12.65 ± 6.372	17.19 ± 7.939	0
RRT*	507.27 ± 9.40	11.55 ± 6.613	15.67 ± 6.381	0

area by taking larger steps, thus resulting in the tree growing almost directly towards the target state, which enabled A-RRT* to find the target more quickly.

Fig. 8(a), Fig. 9(a), Fig. 10(a) and Fig. 11(a) also showed that A-RRT* generates fewer states in a narrower space while optimizing the sub-optimal solutions, revealing its high efficiency in path optimization. This is because the algorithm utilizes subset-informed sampling based on the secondary tree, which reduces the sampling range for converging. In the narrow environment, for example, new states were generated within an elliptical region surrounding the secondary tree based on an initial solution. This restricted the sampling range during convergence, which not only decreased the number of states but also speeded up the convergence towards the optimal solution.

V. CONCLUSION

In this paper, we present a novel sampling-based path planning algorithm, A-RRT*, which incorporates an adaptive goal-biased sampling strategy for initial solution generation and a path optimization approach utilizing a secondary tree and subset-informed sampling. Experimental results across various simulation environments demonstrate the effectiveness of A-RRT*. During the initial solution generation phase, A-RRT* efficiently samples the search space, narrowing down the search area and preventing trapping in local minima, particularly in corridor and maze environments. Subsequently, during the path optimization phase, the algorithm applies a secondary tree and subset-informed sampling approach. This approach narrows the sampling region to an elliptical area around local states in established paths and gradually shrinks it during optimization, significantly enhancing the efficiency of convergence to the optimal path. These findings indicate that the proposed adaptive goal-biased sampling strategy improves the efficiency and reliability of sampling-based path planning in intricate environments. Additionally, the utilization of a secondary tree and subset-informed sampling in path optimization enhances the efficiency of convergence to the optimal path. The proposed method has the potential to significantly enhance efficient motion planning for a range of navigation applications in complex environments, including autonomous driving, robotic rescue operations, home service robots, and game design, etc.

Although A-RRT* can quickly search an initial solution and converge to a sub-optimal solution, this study only considered the two-dimensional environment. Further investigation is needed before applying the proposed algorithm in high-dimensional environment. In addition, in our future research, curve-smoothing and robot kinematic

constraints will be considered for practical application in robotics.

VI. 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] Y. Li, W. Wei, Y. Gao, D. Wang, and Z. Fan, "PQ-RRT*: An improved path planning algorithm for mobile robots," *Expert Syst. Appl.*, vol. 152, Aug. 2020, Art. no. 113425, doi: 10.1016/j.eswa.2020.113425.

[2] Q. Chai and Y. Wang, "RJ-RRT: Improved RRT for path planning in narrow passages," *Appl. Sci.*, vol. 12, no. 23, p. 12033, Nov. 2022, doi: 10.3390/app122312033.

[3] P. Liu and B. Zhang, "An autonomous quadrotor exploration combining frontier and sampling for environments with narrow entrances," in *Proc. 41st Chin. Control Conf. (CCC)*, Jul. 2022, pp. 3656–3661.

[4] I.-B. Jeong, S.-J. Lee, and J.-H. Kim, "Quick-RRT*: Triangular inequality-based implementation of RRT* with improved initial solution and convergence rate," *Expert Syst. Appl.*, vol. 123, pp. 82–90, Jun. 2019, doi: 10.1016/j.eswa.2019.01.032.

[5] D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock, "Randomized kinodynamic motion planning with moving obstacles," *Int. J. Robot. Res.*, vol. 21, no. 3, pp. 233–255, Mar. 2002, doi: 10.1177/027836402320556421.

[6] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Autom.*, vol. 12, no. 4, pp. 566–580, Aug. 1996, doi: 10.1109/70.508439.

[7] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *Int. J. Robot. Res.*, vol. 20, no. 5, pp. 378–400, May 2001, doi: 10.1177/02783640122067453.

[8] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 846–894, Jun. 2011, doi: 10.1177/0278364911406761.

[9] C. Urmson and R. Simmons, "Approaches for heuristically biasing RRT growth," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Las Vegas, NV, USA, Oct. 2003, pp. 1178–1183.

[10] A. H. Qureshi and Y. Ayaz, "Potential functions based sampling heuristic for optimal path planning," *Auto. Robots*, vol. 40, no. 6, pp. 1079–1093, Aug. 2016, doi: 10.1007/s10514-015-9518-0.

[11] J. Fan, X. Chen, Y. Wang, and X. Chen, "UAV trajectory planning in cluttered environments based on PF-RRT* algorithm with goal-biased strategy," *Eng. Appl. Artif. Intell.*, vol. 114, Sep. 2022, Art. no. 105182, doi: 10.1016/j.engappai.2022.105182.

[12] J. Fan, X. Chen, and X. Liang, "UAV trajectory planning based on bi-directional APF-RRT* algorithm with goal-biased," *Expert Syst. Appl.*, vol. 213, Mar. 2023, Art. no. 119137, doi: 10.1016/j.eswa.2022.119137.

[13] F. Islam, J. Nasir, U. Malik, Y. Ayaz, and O. Hasan, "RRT*Smart: Rapid convergence implementation of RRT* towards optimal solution," in *Proc. IEEE Int. Conf. Mechtron. Automat.*, Aug. 2012, pp. 1651–1656.

[14] A. H. Qureshi, S. Mumtaz, Y. Ayaz, O. Hasan, M. S. Muhammad, and M. T. Mahmood, "Triangular geometrized sampling heuristics for fast optimal motion planning," *Int. J. Adv. Robotic Syst.*, vol. 12, no. 2, p. 10, Feb. 2015, doi: 10.5772/59763.

[15] B. Liao, F. Wan, Y. Hua, R. Ma, S. Zhu, and X. Qing, "F-RRT*: An improved path planning algorithm with improved initial solution and convergence rate," *Expert Syst. Appl.*, vol. 184, Dec. 2021, Art. no. 115457, doi: 10.1016/j.eswa.2021.115457.

[16] Z. Wang, Y. Li, H. Zhang, C. Liu, and Q. Chen, "Sampling-based optimal motion planning with smart exploration and exploitation," *IEEE/ASME Trans. Mechatronics*, vol. 25, no. 5, pp. 2376–2386, Oct. 2020, doi: 10.1109/TMECH.2020.2973327.

[17] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2014, pp. 2997–3004.

[18] D. Armstrong and A. Jonasson, "AM-RRT*: Informed sampling-based planning with assisting metric," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2021, pp. 10093–10099.

[19] K. Wang, J. Xu, K. Song, Y. Yan, and Y. Peng, "Informed anytime bi-directional fast marching tree for optimal motion planning in complex cluttered environments," *Expert Syst. Appl.*, vol. 215, Apr. 2023, Art. no. 119263, doi: 10.1016/j.eswa.2022.119263.

[20] B. Ma, C. Wei, Q. Huang, and J. Hu, "APF–RRT*: An efficient sampling-based path planning method with the guidance of artificial potential field," in *Proc. 9th Int. Conf. Mechatronics Robot. Eng. (ICMRE)*, Feb. 2023, pp. 207–213, doi: 10.1109/TIE.2018.2886798.

[21] C.-b. Moon and W. Chung, "Kinodynamic planner dual-tree RRT (DT-RRT) for two-wheeled mobile robots using the rapidly exploring random tree," *IEEE Trans. Ind. Electron.*, vol. 62, no. 2, pp. 1080–1090, Feb. 2015, doi: 10.1109/TIE.2014.2345351.

[22] H. Yang, J. Qi, Y. Miao, H. Sun, and J. Li, "A new robot navigation algorithm based on a double-layer ant algorithm and trajectory optimization," *IEEE Trans. Ind. Electron.*, vol. 66, no. 11, pp. 8557–8566, Nov. 2019.

CHUANRONG HUANG received the B.S. degree in mechanical design, manufacturing and automation from the Southwest University of Science and Technology. She is currently pursuing the M.S. degree in mechanical engineering with the College of Mechanical Engineering, Guangxi University. Her research interests include mobile robotics and path planning.

BINGJIE TANG received the B.S. degree in mechanical engineering from Hohai University. He is currently pursuing the M.S. degree in mechanical engineering with the College of Mechanical Engineering, Guangxi University. His research interest includes mobile robot control and navigation.

ZHIYANG GUO received the B.S. degree in mechatronics engineering from Liaoning University of Science and Technology. He is currently pursuing the M.S. degree in mechanical engineering with the College of Mechanical Engineering, Guangxi University. His research interests include mobile robotics and simultaneous localization and mapping.

QI SU received the B.S. degree in mechatronics engineering from Guangxi University, where he is currently pursuing the M.S. degree in mechanical engineering. His research interests include hyperspectral information and machine learning.

JINGYAO GAI received the Ph.D. degree in agricultural biosystems engineering from Iowa State University. He is currently an Assistant Professor with the College of Mechanical Engineering, Guangxi University. His current research interests include artificial intelligence, robot autonomous navigation, deep learning, and machine vision.