2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

RBI-RRT*: Efficient Sampling-based Path Planning for High-dimensional State Space

Fang Chen, Yu Zheng, *Senior Member, IEEE*, Zheng Wang, *Senior Member, IEEE*, Wanchao Chi, Sicong Liu, *Member, IEEE*

***Abstract*—Sampling-based planning algorithms such as RRT have been proved to be efficient in solving path planning problems for robotic systems. Various improvements to the RRT algorithm have been presented to improve the performance of the extension and convergence of the random trees, such as Informed RRT*. However, with the growth of spatial dimensions, the time consumption of randomly sampling the entire state space and incrementally rewiring the random trees raises drastically before a feasible solution is found. In this paper, to enhance the convergence performance of optimal solutions, we present Reconstructed Bi-directional Informed RRT* (RBI-RRT*) path planning algorithm. The algorithm acts as RRT-Connect to rapidly find a feasible solution, which helps compress the sampling space as Informed RRT* does. After the random trees are transformed into RRT* structure by the reconstruction process in RBI-RRT*, the algorithm continues to find the near-optimal path. A series of simulations and real-world robot experiments were conducted to evaluate the algorithm against existing planning algorithms. Compared to Informed RRT* Connect, RBI-RRT* reduced the computation time of achieving a specific cost by 22.1% on average in simulations and 11.2% in the real-world robotic arm experiments. The results show that RBI-RRT* is more efficient in high-dimensional planning problems.**

I. Introduction

Efficient path planning is fundamental research in robotics. Various path planning algorithms have been developed for different types of robots and application scenarios, such as mobile platform [1], unmanned aerial vehicle (UAV) [2], self-driving car [3], [4], robotic arm with six or more degrees of freedom (DOF) [5] and so on, which are expected to find a feasible or optimal collision-free path in various environments. An ideal algorithm enables robots to efficiently move along this path from start to destination, either in minimal computation time or at the lowest cost.

There are two typical kinds of planning algorithms, graph-based and sampling-based planning algorithm. Graph-based planning algorithms such as Dijkstra [6] and A* [7] search for a path after constructing the entire state space. Such algorithms are mainly used for solving path planning problems in 2D or 3D spaces, since their performance deteriorates significantly with increasing spatial dimensions. On the other hand, sampling-based planning algorithms, such as Probabilistic Roadmaps (PRM) [8] and Rapidly-exploring Random Tree (RRT) [9], search for paths through randomly sampled states instead of constructing the entire state space. These algorithms have demonstrated their effectiveness in addressing path planning problems, for robotic systems like robotic arms and manipulators in high-dimensional state spaces. Sampling-based algorithms have been proven to be probabilistically complete so the solution will be found if exists. RRT* [10] represents the optimal version of RRT, utilizing an incremental, asymptotically optimal strategy to converge towards the best possible solution as the number of samples increases. However, as the dimensionality of the state space increases, the extension and convergence of the random trees become slower. Standard RRT and RRT* may take considerable computation time to find a feasible solution, thereby hindering their real-time applicability. Various improvements to the RRT algorithm have been presented to further improve the performance.

RRT-Connect [11], a bi-directional variant of RRT, extends trees from both the start and goal simultaneously at each sampling step, thus accelerating the search for solutions. However, the algorithm only finds a feasible solution instead of the optimal one. Klemm et al. combined RRT-Connect with RRT* and presented RRT*-Connect [12], the bi-directional variant of RRT* which reduces iterations and time needed to find a near-optimal solution. Based on RRT*, Gemmell et al. presented Informed RRT* [13]. Once a solution is found, the informed sampling is focused on an ellipsoidal heuristic subset of the entire state space, which expedites the convergence towards the optimal solution. The BIT* is proposed in [14], combining incremental graph-search techniques with random geometric graphs (RGG) theory to build an explicit tree. BIT* finds a solution and converges towards the optimum faster than Informed RRT*. However, it doesn't perform as effectively in finding the initial solution as RRT-Connect. Berget et al. presented BI2RRT* [15], a bi-directional variant of Informed RRT*, and showed the advantages against Informed RRT* and other RRT-based algorithms on task-constrained mobile manipulation.

This work originates from the joint graduate program of Southern University of Science and Technology and Tencent Robotics X Laboratory. This work was partly supported by the National Key R&D Program of China (Grant No. 2022YFB4701200), Shenzhen Science and Technology Program Grant JCYJ20220530114615034, JCYJ20220818100417038, ZDSYS20220527171403009 and RCBS20210609104446099, Guangdong Basic and Applied Basic Research Foundation Grant 2021A1515110658, NSFC Grant 51975268. (Corresponding authors: Wanchao Chi and Sicong Liu)

Fang Chen is with Department of Mechanical and Energy Engineering, Southern University of Science and Technology, Shenzhen 518055, China, and Robotics X, Tencent, Shenzhen, Guangdong 518057, China. (E-mail: 12132245@mail.sustech.edu.cn)

Zheng Wang and Sicong Liu are with Department of Mechanical and Energy Engineering, Southern University of Science and Technology, Shenzhen 518055, China. (E-mail: {wangz, liusc}@sustech.edu.cn)

Yu Zheng and Wanchao Chi are with Robotics X, Tencent, Shenzhen, Guangdong 518057, China. (Email: {yuzheng001, wanchao.chi}@gmail.com)

Masayuki et al. presented the Hybrid RRT [16] algorithm, which rapidly finds an initial solution as RRT-Connect and then transitions to standard Informed RRT* with a tree-merging strategy to continue optimizing the solution. However, the unidirectional optimization is not efficient enough. The Informed RRT* Connect was introduced in [17], the informed version of RRT*-Connect. The algorithm finds an initial solution as fast as RRT*-Connect and is more efficient to find a near-optimal solution than Informed RRT*.

Although algorithms with bidirectional-tree optimization, for example, RRT*-Connect, demonstrate better near-optimal solution convergence, it takes a considerate amount of time to find an initial solution due to sampling the entire state space and rewiring states simultaneously. Moreover, the computation time increases with the growth of spatial dimensions [18], consequently prolonging this process. On the contrary, RRT-Connect proves to be more efficient in finding a feasible solution. This helps preliminarily measure the volume of the ellipsoidal heuristic subset of state space and focus on searching through this part earlier than other algorithms. To further optimize the initial solution towards near-optimal, a random tree reconstruction method is required to transform the acquired RRT-Connect tree into the RRT* structure, since RRT-Connect cannot optimize cost of each state in the tree.

To reduce the near-optimal solution convergence time for path planning problems in high-dimensional state spaces, in this paper, we propose the Reconstructed Bi-directional Informed RRT* (RBI-RRT*) path planning algorithm.

(1) RBI-RRT* extends two random trees as RRT-Connect in the initial solution-finding process. The algorithm rapidly finds a feasible solution without rewiring to focus informed sampling on a hyper-ellipsoidal subset, which reduces the time consumption of sampling the entire high-dimensional state space and enhances the success rate.

(2) RBI-RRT* reconstructs two trees respectively into RRT* structure once a solution is found for further bidirectional solution optimization. The reconstruction rewires states in both trees and initially optimizes the solution. Additionally, states which do not belong to the subset will be pruned off trees for a known solution cost. Therefore, the algorithm expedites the convergence towards a near-optimal solution compared to Informed RRT* Connect.

(3) Compared to existing state-of-art planning algorithm, Informed RRT* Connect, RBI-RRT* reduced planning time for 6-DOF robotics arm by 22.1% on average in simulations and 11.2% in real-world experiments. The results demonstrated the effectiveness of RBI-RRT* for solving high-dimensional planning problems.

II. ALGORITHM

A. Optimal path planning

The optimal path planning problem is defined in this section. Let $X \subseteq \mathbb{R}^n$ be the state space of the planning problem, $X_{free} \subseteq X$ be the set of all collision-free states, $x_{start} \in X_{free}$ be the start state and $x_{goal} \in X_{free}$ be the goal state. A feasible path can be represented as $\sigma: [0,1] \to X_{free}$ where $\sigma(0) = x_{start}$ and $\sigma(1) = x_{goal}$. Therefore the optimal path planning problem can be defined as searching the collision-free path σ^* and minimizing a given cost function $C(\sigma)$. It can be presented as

$$\sigma^* = \arg\min \{C(\sigma) \mid \sigma(0) = x_{start}, \sigma(1) = x_{goal}, \forall s \in [0,1], \sigma(s) \in X_{free}\}. \quad (1)$$

The informed sample strategy $f(\cdot)$ compresses the state space with a given path σ into a subset $\hat{X}_{free} \subseteq X_{free}$ that helps improve the current solution. Sampling in $\hat{X}_{free}$ means that there is a higher probability to make the sample be closer to the optimal path σ^*.

B. Reconstructed Bi-directional Informed RRT*

The pseudocode of the RBI-RRT* is shown in Algorithm 1, and shaded rows represent the differences from Informed RRT* Connect. A random tree is presented as $T = (V, E)$, consisting of collision-free states $V \subseteq X_{free}$ in the random tree and the set of motions $E = (x_{from}, x_{to})$that connect two states. Additionally, T_1 and T_2 represent trees rooted at start and goal respectively.

RBI-RRT* extends two trees as RRT-Connect until the first feasible solution is found. The minimum cost of solutions c_{best} is firstly set to infinity, so that a random state is sampled in the entire state space. The *InformedSample* function returns a random uniform sample $x_{rand} \in X_{free}$ in the entire state space when $c_{best} = \infty$ or $x_{rand} \in \hat{X}_{free}$ in the hyper-ellipsoid subset when $c_{best} < \infty$. The principles and details of *InformedSample* refer to [19]. When a collision-free state $x_{new} \in X_{free}$ is added to the tree, the cost c_{new} from x_{new} to the root state is calculated during each iteration. Let x_{new1} and x_{new2} represent the new states added to T_1 and T_2 respectively. After T_1 extend a single state x_{new1} towards x_{rand}, T_2 keeps extending towards x_{new1} until it reaches the state or encounters a collision (Algorithm 1, Line 9-12). The last state added to the tree is recorded as x_{new2}. The essential difference between RBI-RRT* and Informed RRT* Connect is that the state optimization processes, choosing the best parent for a state and rewiring its neighbors, are skipped before the first solution is found in function *Extend**, as shown

Algorithm 1 Reconstructed Bi-directional Informed RRT*

```
1  V_1 ← {x_start}; E_1 ← ∅;
2  V_2 ← {x_goal}; E_2 ← ∅;
3  X_soln ← ∅;
4  c_best ← ∞;
5  T_1 ← (V_1, E_1); T_2 ← (V_2, E_2);
6  while !TerminationCondition
7      c_best ← min{Cost(X_soln)};
8      x_rand ← InformedSample(x_start, x_goal, c_best);
9      if Extend*(T_1, x_rand) ≠ TRAP then
10         do
11             result ← Extend*(T_2, x_new1);
12         while result ≠ ADVANCE;
13         if result = REACH then
14             X_soln ← X_soln ∪ {(x_new1, x_new2)};
15     if FirstSolution then
16         Reconstruct(T_1, c_best);
17         Reconstruct(T_2, c_best);
18     swap(T_1, T_2);
19 return T_1, T_2;
```

in Algorithm 2, Line 13. When x_{new2} reaches x_{new1}, T_1 connects to T_2 and these two states represent the same state in two trees respectively. This pair of connection states is added to a feasible solution set X_{soln} (Algorithm 1, Line 14).

Algorithm 2 $Extend^*(T, x)$

$x_{nearest} \leftarrow Nearest(T, x)$;
$x_{new} \leftarrow Steer(x_{nearest}, x)$;
if $CollisionFree(x_{nearest}, x_{new})$ **then**
  $V \leftarrow V \cup \{x_{new}\}$;
  $x_{min} \leftarrow x_{nearest}$;
  if $c_{best} < \infty$ **then**
    $X_{near} \leftarrow Near(T, x_{new}, r_{RRT^*})$;
    $Parent(x_{new}) \leftarrow ChooseParent(x_{min}, x_{new}, X_{near})$;
    $E \leftarrow E \cup \{(x_{min}, x_{new})\}$;
    $Rewire(T, x_{new}, X_{near})$;
  else
    $E \leftarrow E \cup \{(x_{min}, x_{new})\}$;
  if $x_{new} = x$ **then**
    return $REACH$;
  else
    return $ADVANCE$;
return $TRAP$;

C. Random tree reconstruction

The trees will be reconstructed after finding the first solution (Algorithm 1, Line 15-17) and the pseudocode of the function *Reconstruct* is shown in Algorithm 3. Fig.1 shows the illustration of the random tree reconstruction process. Given a random tree $T(V, E)$ as shown in Fig.1(a), we have a batch of states $V \subseteq X_{free}$ with unoptimized cost and a set of edges E with the standard RRT structure. The *Reconstruct* function transforms the tree into RRT* structure, as well as minimizing the cost of each state. It can be presented as

$$V = \min\{Cost(x) | \forall x \in V \cap \hat{X}_{free}\}. \quad (2)$$

Clear the tree first and the reconstruction process starts with the root state x_{root} of the tree. All child states of the current state x_{cur} are added to a queue of states X_{child} for processing. x_{cur} is then changed to the front state in the queue. Since the informed hyper-ellipsoidal subset $\hat{X}_{free}$ has been defined by c_{best}, states that cannot help to find a better solution should be pruned off the tree. States with estimate cost c_{est} below c_{best} will be added to the tree, where c_{est} is

Algorithm 3 $Reconstruct(T, c_{best})$

$T \leftarrow \emptyset$;
$V \leftarrow \{x_{root}\}$;
$X_{child} \leftarrow Children(x_{root})$;
for $\forall x_{child} \in X_{child}$ **do**
  $c_{est} \leftarrow Cost(x_{child}) + CostToGo(x_{child})$;
  $x_{min} \leftarrow Parent(x_{child})$;
  if $c_{est} < c_{best}$ **then**
    $X_{child} \leftarrow X_{child} \cup Children(x_{child})$;
    $X_{near} \leftarrow kNear(T, x_{child}, k_{RRT^*})$;
    $Parent(x_{child}) \leftarrow ChooseParent(x_{min}, x_{child}, X_{near})$;
    $E \leftarrow E \cup \{(x_{min}, x_{child})\}$;
    $Rewire(T, x_{child}, X_{near})$;
  $X_{child} \leftarrow X_{child} \setminus x_{child}$;
$c_{best} \leftarrow Cost(x_{soln1}) + Cost(x_{soln2})$;

the sum of cost of this state $c_{cur} = Cost(x_{cur})$ and distance to the random tree's target state c_{togo}. Add children of the current state in front of the state queue for a depth-first optimization.

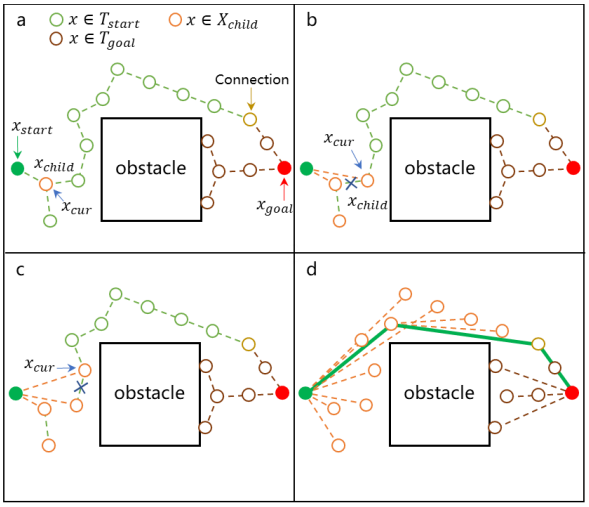

Fig. 1. Illustration of the reconstruction process. (a) Random trees in RRT structure. The green dots represent the tree of start state and the red ones represent the trees of goal states. The solid dots represent the root states and the hollow dots represent the vertices on the tree. (b) Choose the best parent of the current state and rewire its neighbors. Then add children states into the queue and change x_{cur} to the front state in the queue. (c) Optimize the new x_{cur} and add its children repeatedly for a depth-first optimization. (d) The reconstructed trees and the optimized path.

For each state added to the new tree, choose the best parent state for itself in *ChooseParent* function and its neighbors in *Rewire* function, as shown in Fig.1(b). The *ChooseParent* function picks out the best parent state among X_{near} for x_{child} that minimize the cost of x_{child}. The *Rewire* function changes the parent of states among X_{near} to x_{child} if the cost will be smaller. The principles of these two functions refer to RRT* [10] and pseudocodes are shown in Algorithms 4 and 5 for a complete description of RBI-RRT*. It is notable that, instead of r-Rewire method, we use k-Nearest [10] when finding neighbors of a state. The former returns states with distance below radius r, while the latter returns k states nearest to the current state and k is calculated by

$$k = 2^{n+1} e \left(1 + \frac{1}{n}\right), \quad (3)$$

where n is the dimension of the state space. With a given batch of states V, k-Nearest method ignores the distance between states and is more likely to connect to the farthest collision-free state. Therefore, it is more efficient to minimize the cost of each and helps further compress the hyper-ellipsoidal subset with a lower solution cost. Pop the current state out of the queue after finishing the above steps and then turn to the next state until the queue is empty.

So far, the reconstruction process has completed and the cost of the solution should be calculated again to check whether the hyper-ellipsoidal subset can be further compressed. After the reconstruction of both trees, each state on the tree satisfies the optimal cost condition with a tree structure similar to RRT*. The algorithm then turns to Informed RRT* Connect for further path optimization.

Algorithm 4 $ChooseParent(x_{min}, x_{new}, X_{near})$

$c_{min} \leftarrow Cost(x_{min}) + cLine(x_{min}, x_{new})$;
for $\forall x_{near} \in X_{near}$ **do**
  $c_{new} \leftarrow Cost(x_{near}) + cLine(x_{near}, x_{new})$;
  if $c_{new} < c_{min}$ **then**
    if $CollisionFree(x_{near}, x_{new})$ **then**
      $x_{min} \leftarrow x_{near}$;
      $c_{min} \leftarrow c_{new}$;
return x_{min}

Algorithm 5 $Rewire(T, x_{new}, X_{near})$

for $\forall x_{near} \in X_{near}$ **do**
  $c_{new} \leftarrow Cost(x_{new}) + cLine(x_{near}, x_{new})$;
  if $c_{new} < Cost(x_{near})$**then**
    if $CollisionFree(x_{near}, x_{new})$ **then**
      $x_{parent} \leftarrow Parent(x_{near})$;
      $E \leftarrow E \setminus \{(x_{parent}, x_{near})\}$;
      $E \leftarrow E \cup \{(x_{new}, x_{near})\}$;

III. Experimental Results

Experiments were conducted on a variety of planning problems both in simulation and on a real-world robotic arm platform. RBI-RRT* was compared to Hybrid RRT and Informed RRT* Connect in simulation to validate the advantages of the proposed algorithm in initial solution finding and optimal path converging efficiency. The proposed algorithm was then applied to path planning on a real-world robotic arm, compared with standard Informed RRT* and Informed RRT* Connect to further validate the advantages of bi-directional extending and random tree reconstruction of RBI-RRT*. The algorithms were implemented using the Open Motion Planning Library (OMPL) [20]. All algorithms in each experiment had the same parameters. All experiments were repeated 100 times and the lowest cost of all feasible solutions found within a specific time interval was recorded with an independent thread. The time interval varied across different spatial dimensions, as the time needed for calculation and near-optimal solution convergence increases with higher spatial dimensionality.

A. Simulations in randomly generated world

To verify the reliability and assess the performance of the proposed algorithm, simulations of solving planning problems in $\mathbb{R}^2$ state space were conducted. The planning problems involved minimizing path length in randomly generated worlds. The size of the $\mathbb{R}^2$ random world was 25*25. Square obstacles were randomly generated in the world, as depicted in Fig.2(a) in black. The size of each obstacle ranged randomly from 0.3 to 0.8. The start and goal states in each planning problem were randomly selected and ensured to be separated by a certain distance. In each experiment, the planning algorithms were executed for 5s, with a time interval of 0.001s for recording the current lowest cost. The simulations were carried out on a laptop with the Intel Core i5-12500H processor and 16GB of RAM, and running Ubuntu 20.04.

Since the optimal solutions were different and unknown for each problem definitions and state dimensions, comparison between different problems was meaningless. Therefore, a representative set of simulation results were presented in Fig.2. Fig.2(b) shows the success rate of finding a feasible solution up to a certain time of each algorithm in $\mathbb{R}^2$. The success rate curve reflected the speed of finding a feasible solution of the algorithm. RBI-RRT* and Hybrid RRT demonstrated high success rates due to the similar process when finding the initial solution, significantly higher than that of Informed RRT* Connect. Fig.2(c) shows the mean computation time and path cost of the initial solution found by each algorithm, as well as those of RBI-RRT* finishing tree reconstruction process. Error bars on the plot indicated 95% confidence intervals. Despite the initial mean cost of RBI-RRT* was considerably higher without the optimization process, it could be significantly reduced after the random tree reconstruction. Additionally, the mean computation time was even less than that of Informed RRT* Connect with an average reduction of 15.2%. This indicated that RBI-RRT* could converge to the optimal solution faster than Informed RRT* Connect, since the former would transform into the latter after completing the reconstruction process.

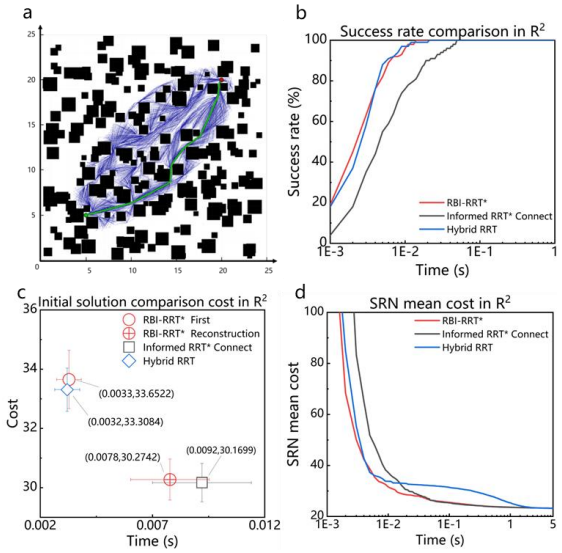

Fig. 2. A set of simulations result in $\mathbb{R}^2$ state space randomly generated world. (a) A simulation test result of RBI-RRT*. The green line represents the solution found after 5s. The blue line represents an edge of the random tree. (b) Comparison of success rate for 100 repetitions in $\mathbb{R}^2$ random world of each algorithm. (c) Comparison of mean computation time and path cost of initial solutions of each algorithm as well as reconstructed solutions of RBI-RRT*. (d) Comparison of success-rate-normalized mean cost (SRN mean cost) of each algorithm.

The calculation of the mean cost took into account only the experiments that had successfully found a solution, as the costs for cases without feasible solutions were treated as infinite. Hence, the mean cost would display fluctuations when the success rate was less than 100%. To simultaneously account for both the success rate and mean cost, we introduce the concept of success-rate-normalized mean cost (SRN mean cost) as

$$\hat{c}_{mean} = \frac{c_{mean}}{r}. \tag{4}$$

It represents the equivalent mean cost if all planning trials were successful. The success-rate-normalized mean cost is infinite when the success rate $r = 0$, which means there is no feasible solution and it turns to mean cost c_{mean} when $r = 100\%$. It can be represented by

$$\hat{c}_{mean}\,|_{r=0} = \infty\,, \tag{5}$$

$$\hat{c}_{mean}\,|_{r=1} = c_{mean}\,. \quad (6)$$

The SRN mean cost of each algorithm was plotted in Fig.2(d). The solution optimization process of Hybrid RRT was slower than the other algorithms due to the single-tree unidirectional optimization. RBI-RRT* had a significantly lower SRN mean cost in the early stage of planning than Informed RRT* Connect due to higher success rate. However, the time reduction in low-dimensional state spaces was negligible so that the proposed algorithm could not have a definite impact on cost convergence compared to Informed RRT* Connect.

B. Simulations of 6-DOF robotic arm

To validate the advantages of RBI-RRT* in high-dimensional state spaces, simulations of a 6-DOF ABB robotic arm were conducted. To grasp the glass cup on the table, the planning algorithms were expected to minimize the cost of moving the robotic arm from the start state to the goal and avoid collision to obstacles. The simulation environment was constructed using Pybullet, as shown in Fig.3. The positions of obstacles and grasp target were changed in different experiments. In each experiment, the planning algorithms were executed for 20s, with a time interval of 0.01s for recording the current optimal cost. Fig.3(a)-(d) show the trajectory of the robotic arm planned by RBI-RRT* in one run.

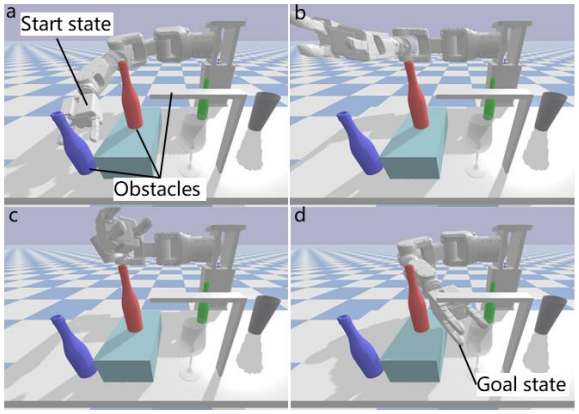

Fig. 3. The simulation of the 6-DOF robotic arm grasping task scenario. The environment was conducted by Pybullet. (a) – (d) show the trajectory of the robotic arm planned by RBI-RRT* in one run.

Fig.4(a) shows the success rate of each algorithm. The success rate of RBI-RRT* is comparable to that of Hybrid RRT, both of which are higher than Informed RRT* Connect. Over 100 repetitions, the average time for RBI-RRT* to find a feasible solution was 0.42s while Informed RRT* Connect took an average of 1.79s. As shown in Fig.4(b), RBI-RRT* found the first solution in 0.1730s on average and finished tree reconstruction process in 0.2772s on average. The percentage of time reduction was 10.6% compared to Informed RRT* Connect finding its first solution in 0.3100s on average. Fig.4(c) shows the comparison of SRN mean cost of the solutions within the first 5s of planning process. The SRN mean cost of RBI-RRT* was lower than Informed RRT* Connect due to a higher success rate. The mean cost dropped rapidly and stabilized at a lower level. The path cost of both bi-directional RRT* variants is lower than that of Hybrid RRT, indicating higher efficiency in solution optimization. Fig.4(d) shows the time reduction percentage of RBI-RRT* compared to Informed RRT* Connect when finding a solution below certain cost. When the path cost dropped from 3.5 to 2.0, the time required by RBI-RRT* was lower than that of Informed RRT* Connect, with an average of 22.1% and a maximum of 30.7%. These results shows that the reconstruction algorithm improves the efficiency of path convergence to the optimal in high-dimensional state spaces.

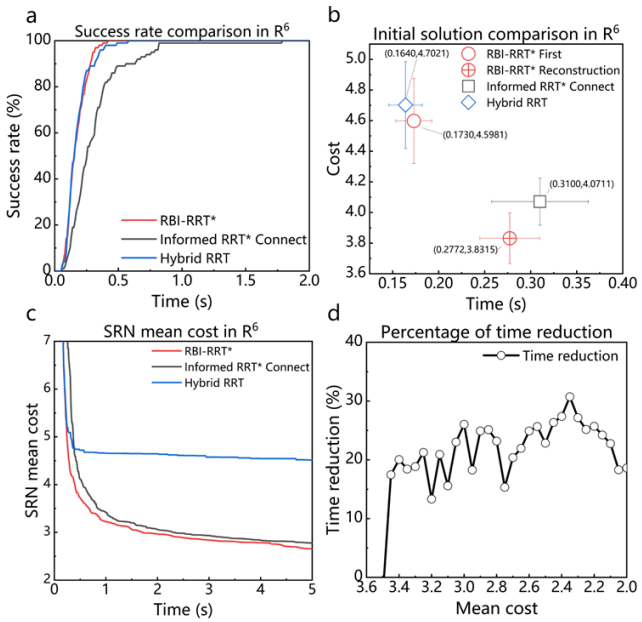

Fig. 4. A set of results of the 6-DOF robotic arm grasping task simulation. (a) Comparison of success rate for 100 repetitions of each algorithm. (b) Comparison of mean computation time and path cost of initial solutions of each algorithm as well as reconstructed solutions of RBI-RRT*. (c) Comparison of SRN mean cost in the first 5s of each algorithm. (d) The time reduction percentage of RBI-RRT* compared to Informed RRT* Connect when finding a solution below certain cost.

C. Experiments on real-world robotic arm

The task for the robotic arm was to grasp the blue bottle without colliding with the green bottle, as shown in Fig.5(a). The position of the bottles was recognized by the binocular camera on the head of the robot. Similar to the simulations, the planning algorithms were expected to find a collision-free path to move the robotic arm from start state to the grasp state. There were two termination conditions for the path planning: (1) the planning stopped after a certain time, (2) given a cost threshold factor, the planning stopped when the cost of path is below this factor times distance from start to the goal. After planning for a path, the simplification process would be carried out attempting to simplify the planned path. It was implemented using OMPL. Then interpolated 100 points into the path to obtain a smoother trajectory. Time duration of each process as well as the cost of planned and simplified path was recorded. The experiments were carried out on a desktop with the Intel Core i7-10700 processor and 32GB of RAM, and running Ubuntu 20.04.

Fig.5(b-e) show the comparison of the trajectories of the robotic arm between RBI-RRT* and Informed RRT* Connect with or without simplification process in a single run. The results of each run were recorded in the accompanying video. The green lines represent the start pose and the red lines represent the grasping pose. Dots represent the position of each joint in the Cartesian coordinate system. In both cases, the path costs of RBI-RRT* were lower than those of

Informed RRT* Connect. Specifically, the cost of RBI-RRT* with or without simplification was 2.60 or 5.18 respectively, which is 16.3% or 15.9% lower than those of Informed RRT* Connect, which was 3.09 or 6.19.

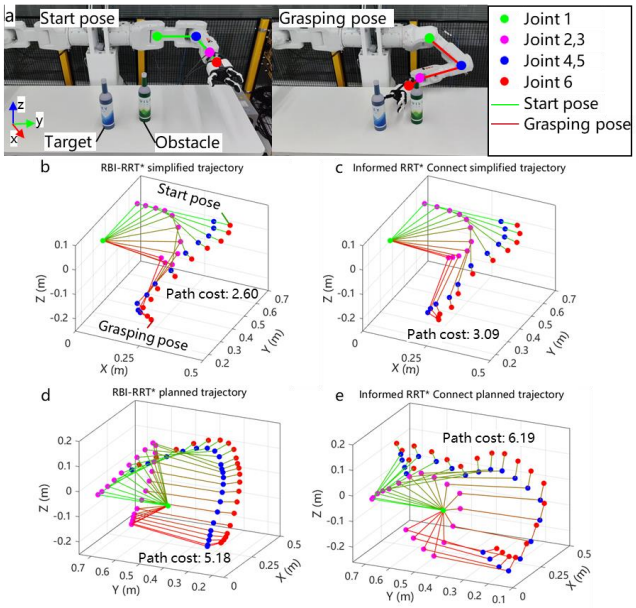

Fig. 5. The experiments setup and comparison of trajectories between different planning algorithms with or without simplification process in a single run. (a) The experiments setup. The task for the robotic arm was to grasp the blue bottle without colliding with the green bottle. Figs. (b, c) show simplified trajectories of RBI-RRT* and Informed RRT* Connect respectively. Figs. (d, e) show trajectories of the two planner without simplification. Path costs of RBI-RRT* in both cases were lower than those of Informed RRT* Connect.

Table I and Fig.6 shows the results of 100 successful planning attempts experiments in different planning time. The entire process was repeated until successfully completed 100 times to calculate the average time for each stage. The cost threshold factor was 2.0 in both experiment. Planning time of experiment in Fig.6(a) was 0.2s. In this case, all three planning algorithms terminated planning due to the time condition. RBI-RRT* demonstrated the highest success rate. The cost of planned path of RBI-RRT* was 9.5% lower than that of Informed RRT* Connect. Planning time of experiment in Fig.6(b) was 0.5s. In this case, the planning algorithms would likely terminate planning due to the cost condition. With the given termination conditions, both bi-directional planning algorithms completed 100 repetitions in success rate of 100% while Informed RRT* was only 36%. RBI-RRT* finished planning process 11.2% faster than Informed RRT* Connect. The cost of planned path of RBI-RRT* was 10.0% lower than that of Informed RRT* Connect. The planning time and cost of Informed RRT* was much higher than the other two algorithms.

The results of the experiment showed that, compared to Informed RRT* Connect, RBI-RRT* showed considerable advantages when applied to a real robotic arm both in planning time and path cost. The reconstruction algorithm has a positive impact on rapidly compressing the sampling space and further converge the solution to the nearly-optimal.

TABLE I. RESULTS OF 100 SUCCESSFUL PLANNING ATTEMPTS

	RBI-RRT*	**Informed RRT* Connect**	**Informed RRT***
Planning in 0.2s			
Success rate	100 %	94 %	23 %
Planning time	216 ms	199 ms	201 ms
Path cost	4.19	4.63	4.83
Planning in 0.5s			
Success rate	100 %	100 %	36 %
Planning time	403 ms	454 ms	475 ms
Path cost	4.12	4.58	4.90

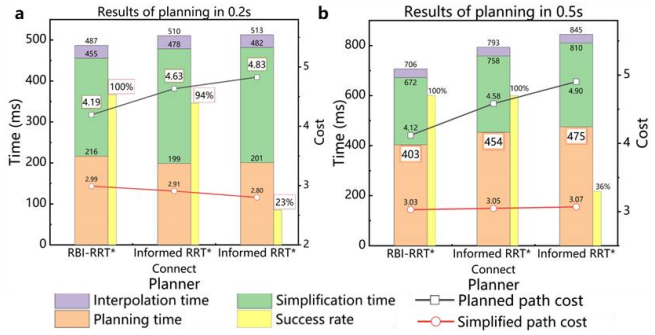

Fig. 6. Results of the experiments for 100 successful attempts in different planning time. (a) Results of planning in 0.2s. (b) Results of planning in 0.5s.

IV. CONCLUSION

In this paper, we propose a sampling-based planning algorithm named Reconstructed Bi-directional Informed RRT*. The algorithm combines RRT-Connect with Informed RRT*. After RRT-Connect has found the first feasible solution, states on both random trees in RRT structure are transformed to optimal, by utilizing the introduced random tree reconstruction method. This optimization procedure involves selecting the optimal parent and rewiring the neighbors of a state, as RRT* does. The reconstruction process decreases the solution's cost and enables RRT-Connect to employ incremental, asymptotically optimal strategies, ultimately converging towards an optimal solution. Subsequently, the sampling and solution optimization process continues within the hyper-ellipsoidal informed subset defined by the solution cost.

The results of simulations and experiments on a real-world robotic arm show that the proposed algorithm can rapidly find a feasible solution and obtain a near-optimal solution faster than those that extending random trees and optimizing states simultaneously. Compared to Informed RRT* Connect, RBI-RRT* reduced the time required to achieve a specific cost by 22.1% on average in simulations and 11.2% in the real-world robotic arm experiments. The performance of RBI-RRT* shows its superiority in solving planning problems in high-dimensional state spaces.

Future work includes further improving the RBI-RRT* algorithm to accelerate the near-optimal path convergence after finishing the random trees reconstruction to achieve efficient path planning on robot platforms with higher dimensions.

ACKNOWLEDGMENT

The authors would like to thank Xiangchi Chen for his support with the robotic arm experiments.

REFERENCES

[1] P. Raja and S. Pugazhenthi, "Optimal path planning of mobile robots: A review," International journal of physical sciences, vol. 7, no. 9, pp. 1314-1320, Feb. 2012.

[2] L. Yang, J. Qi, J. Xiao and X. Yong, "A literature review of UAV 3D path planning," Proceeding of the 11th World Congress on Intelligent Control and Automation, Shenyang, 2014, pp. 2376-2381.

[3] D. González, J. Pérez, V. Milanés and F. Nashashibi, "A Review of Motion Planning Techniques for Automated Vehicles," in IEEE Transactions on Intelligent Transportation Systems, vol. 17, no. 4, pp. 1135-1145, April 2016.

[4] Y. Kuwata, J. Teo, G. Fiore, S. Karaman, E. Frazzoli and J. P. How, "Real-Time Motion Planning With Applications to Autonomous Urban Driving," in IEEE Transactions on Control Systems Technology, vol. 17, no. 5, pp. 1105-1118, Sept. 2009.

[5] T. Sandakalum and M. H. Ang, "Motion Planning for Mobile Manipulators—A Systematic Review," Machines, vol. 10, no. 2, p. 97, Jan. 2022.

[6] E. W. Dijkstra, "A Note on Two Problems in Connexion with Graphs," in Edsger Wybe Dijkstra: His Life,Work, and Legacy, 2022, pp. 287-290.

[7] P. E. Hart, N. J. Nilsson and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," in IEEE Transactions on Systems Science and Cybernetics, vol. 4, no. 2, pp. 100-107.

[8] L. E. Kavraki, P. Svestka, J. -C. Latombe and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," in IEEE Transactions on Robotics and Automation, vol. 12, no. 4, pp. 566-580, Aug. 1996.

[9] S. M. LaValle and J. J. Kuffner, "Randomized Kinodynamic Planning," The International Journal of Robotics Research, vol. 20, no. 5, pp.378-400, May 2001.

[10] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," The International Journal of Robotics Research, vol. 30, no. 7, pp. 846-894, June 2011.

[11] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065), San Francisco, CA, USA, 2000, pp. 995-1001.

[12] S. Klemm et al., "RRT*-Connect: Faster, asymptotically optimal motion planning," 2015 IEEE International Conference on Robotics and Biomimetics (ROBIO), Zhuhai, China, 2015, pp. 1670-1677.

[13] 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," 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, 2014, pp. 2997-3004.

[14] J. D. Gammell, S. S. Srinivasa and T. D. Barfoot, "Batch Informed Trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs," 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, USA, 2015, pp. 3067-3074.

[15] F. Burget, M. Bennewitz and W. Burgard, "BI2RRT*: An efficient sampling-based path planning framework for task-constrained mobile manipulation," 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Daejeon, Korea (South), 2016, pp. 3714-3721.

[16] R. Mashayekhi, M. Y. I. Idris, M. H. Anisi and I. Ahmedy, "Hybrid RRT: A Semi-Dual-Tree RRT-Based Motion Planner," in IEEE Access, vol. 8, pp. 18658-18668, 2020.

[17] R. Mashayekhi, M. Y. I. Idris, M. H. Anisi, I. Ahmedy and I. Ali, "Informed RRT*-Connect: An Asymptotically Optimal Single-Query Path Planning Method," in IEEE Access, vol. 8, pp. 19842-19852, 2020.

[18] J. M. Esposito, "Concentration of Measure Phenomenon and its Implications for Sample-based Planning Algorithms in Very-High Dimensional Configuration Spaces," 2023 IEEE International Conference on Robotics and Automation (ICRA), London, United Kingdom, 2023, pp. 7865-7871.

[19] J. D. Gammell, T. D. Barfoot and S. S. Srinivasa, "Informed Sampling for Asymptotically Optimal Path Planning," in IEEE Transactions on Robotics, vol. 34, no. 4, pp. 966-984, Aug. 2018.

[20] I. A. Sucan, M. Moll and L. E. Kavraki, "The Open Motion Planning Library," in IEEE Robotics & Automation Magazine, vol. 19, no. 4, pp. 72-82, Dec. 2012.