Proceeding of the IEEE
International Conference on Robotics and Biomimetics
Dali, China, December 2019

Locally Guided Multiple Bi-RRT∗ for Fast Path Planning in Narrow Passages*

Xin Shu, Fenglei Ni, Zhou Zhou, Yechao Liu and Hong Liu
Department of the State Key Laboratory of Robotics and System
Harbin Institute of Technology
Harbin, Heilongjiang Province, China
{x.shu, flni, zhouzhou, yechaohit, hong.liu}@hit.edu.cn

Tian Zou
Department of system dynamics
KUKA JV(Shunde)
Foshan, Guangdong Province, China
tian.zou@kuka.com

***Abstract*— Rapidly-exploring Random Tree Star (RRT∗) and its variants can provide a collision-free and asymptotic optimal solution for many path planning problems. However, it is inefficient for many RRT∗ based variants to rapidly find one initial solution in a clustered environment with narrow passages, since consuming high memory as well as time, due to a large number of iterations in sampling critical nodes. To overcome this problem, the paper proposes the Locally Guided Multiple Bi-RRT∗ (LGM-BRRT∗) method, which can provide a fast solution by incorporating an improved bridge-test and a novel search strategy based on local guidance. It ensures an accelerated success rate and more efficient memory utilization compared with Bidirectional RRT∗(BRRT∗), and it is easy for implementation as well. It was verified on different types of scenarios in terms of efficiency and success rate. The results demonstrated that LGM-BRRT∗ is beneficial for fast path planning in a clustered environment with narrow passages.**

***Index Terms*— Path Planning, Narrow Passage, Local Guidance, Fast Planning.**

I. Introduction

Motion planning is one of the fundamental research topics in robotics and it can be used extensively in diverse fields, such as surgical planning, autonomous exploration and computational biology. There are two major classes of motion planning algorithms. One assures completeness, and the other assures probabilistic completeness. However, many of them are inefficiency in a cluttered environment [1]. Probabilistic Road Maps (PRM) [2] and Rapidly-exploring Random Trees (RRT) [3], as sampling based algorithms, are effective. However, PRM and its variants are unsuited for planning online or an unknown environment [4]. RRT, a type of single query method, is capable of most motion planning problems, which is online and constrained [3]. And its many variants have better performance, e.g., RRT∗ [4], RRT-Connect [5], BRRT∗ [6]. Nevertheless, the RRT-based variants tend to be trapped around narrow passages [7]. As a result, it consumes lots of time due to a large number of iterations utilized in sampling critical nodes.

To overcome that shortcoming, a series of improved RRT variants have been reported in the literature. One solution is to adopt a non-uniform sampling strategy. For example, forcing sampling towards the surfaces of obstacles was proposed in [8]; Similar to boundary sampling, Gaussian sampling strategy increases the probability of sampling around obstacles [9]; Dynamic-domain RRT [10] puts obstacles into the Voronoi bias and controls the Voronoi bias of the nodes to make a better exploration; A method to control probabilistic diffusion in motion planning algorithms is presented in [11] . Another solution is to construct multiple trees to increase the probability of key sampling, e.g., bidirectional and multiple RRT∗ algorithms [12], [13], [14], [15], [16]. Wang et al. [17] presented a multi-RRTs which generates multiple trees in narrow passages, and models the tree selection process as a multi-armed bandit problem to explore. Zhong et al. [18] proposed a method to grow the third tree by identifying narrow passages and employed RRT-Connect to heuristically expand. Clearly, it is critical for sampling based methods to make samples fall into hard-reached regions. However, a method, which is high-efficient and easy for implementation, still needs to be further explored.

This paper proposes a novel method, called Locally Guided Multiple Bi-RRT∗ (LGM-BRRT∗), which can provide a fast solution by incorporating an improved bridge-test and a novel search strategy based on local guidance. It is divided into pre-processing phase and search phase. In the pre-processing phase, identify narrow passages and obtain unique identification points (*IP*). *IPs* may be the root node of the local trees in search phase. In the search phase, two global trees constitute bidirectional RRT∗ global exploration, and local trees will be generated to guide sampling in the direction of the hard-reached region when required.

This idea of employing both local-tree generated strategy and search strategy to ensure the accelerated success rate is novel. This algorithm can ensure an accelerated rate and more efficient memory utilization compared to BRRT∗. It has been tested and verified on various types of scenarios. The rest of the paper is organized as follows. In Section II, related work about BRRT∗ and an improved bridge-test method is introduced. Section III describes LGM-BRRT∗ path planning algorithm in detail. Section IV analyses the properties of the proposed algorithm simply. Then the performance is verified in Section V and it is summarized in Section VI.

*This project was supported by National Key R&D Program of China (Grant No. 2017YFB1300400), National Natural Science Foundation of China (Grant No. 51875114), and the Foundation for Innovative Research Groups of the National Natural Science Foundation of China (Grant No.51521003).

978-1-7281-6321-5/19//$31.00 ©2019 IEEE

II. Related Work And Preprocessing

A. Problem Definition and Terminology

Let $X \subseteq \mathbb{R}^d$ denote the configuration space, and let $X_{obs} \subset X$ denote the obstacle region. The set defined as $X_{free} := X \backslash X_{obs}$ is called obstacle-free space. A path in X is a continuous function $\sigma : [0,1] \rightarrow X$. The path σ is named to be collision-free if $\sigma(\tau) \rightarrow X_{free}$ for all $\tau \in [0,1]$. The set of all collision-free paths is denoted by Σ_{free}.

The motion planning problem $(X_{free}, X_{init}, X_{goal})$ is to find a feasible path $\sigma : [0,1] \rightarrow X_{free}$ such that $\sigma(0) = X_{init}$, and $\sigma(1) = X_{goal}$. Given a cost function $c : \Sigma_{free} \rightarrow \mathbb{R}_{\geq 0}$. The optimal motion planning problem is to find a feasible path $\sigma^* : [0,1] \rightarrow X_{free}$ such that $c(\sigma^*) = min\{c(\sigma) : \sigma\ is\ feasible\}$.

B. Bidirectional RRT∗ Algorithm

Jordan and Perez [12] proposed an optimal Bi-directional Rapidly-exploring Random trees (BRRT∗), which is a two-tree variant of RRT∗. It can provide an asymptotic optimality guarantee with higher convergence rate. The brief flow is as below. Firstly, two trees, denoted T_a and T_b, are maintained. Then, the initial operations are just like RRT∗, where first a vertex is randomly sampled, and then insertion and rewiring of the sample into the selected tree T_a, and next the nearest vertex $x_{connect}$ from the opposing tree T_b to x_{new} are searched. After that, the *connect* procedure attempts to connect to the vertex on the other tree only if the resulting solution of cost is lower than the current best. Finally, the algorithm updates its current best solution. Next, the trees T_a and T_b are swapped and the loop will be terminated until i to N. Its pseudo code can be found in [12].

C. Sampling in Narrow Passages

Narrow passage is usually a small sub-region surrounded by obstacles, and it's hard for uniform sampling methods to pass through [1]. One effective approach is to identify such sub-regions and to locally guide the exploration when required.

The bridge-test [19] is one of the effective methods to identify narrow passages. It randomly generates a short segment with two ends (q_f and q_s) in obstacles and one midpoint (q_m) in free spaces. Such a segment line is called a bridge, shown in Fig.1, and the set of q_m is called identification points. Obviously, building a "bridge" in the narrow regions is much easier than that in the wide-open free space [19], [20]. At last, obtain unique identification points (*IP*) for each narrow passage via cluster analysis method.

To avoid building "bridges" near the tip of corner, orthogonal test method is employed in [18]. As for hierarchical clustering and density-based clustering method, although they can automatically obtain the number of *IPs*, hierarchical clustering is less efficient and it may not be accurate. The density-based clustering is not suitable for large differences in cluster density. Thus, K-means++ [21] clustering algorithm is employed, since it can manually set the amount (K value) of the cluster (i.e., the number of *IPs*) and it is fast and simple to use. Elbow Method (EM)[22] is employed to obtain the number of *IPs*. Algorithm 1 is a slightly modified implementation of bridge-test in [17] (Lines 6,10,11). The expected identification results is demonstrated in Fig.2c. Following is a brief description of the main processes.

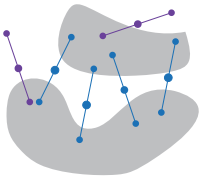
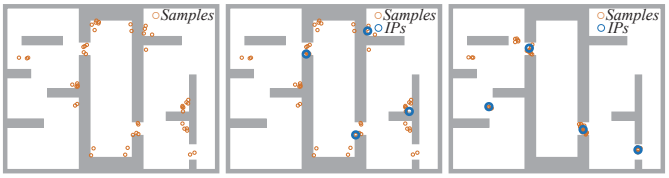


Fig. 1: The principle of bridge test[17]. The color in blue indicates passing the test, otherwise don't passing.



(a) (b) (c)

Fig. 2: The identification results of the bridge-test. (a) The original bridge-test method. (b) The bridge-test with cluster analysis. (c)The improved bridge-test method with cluster analysis.

Algorithm 1:Improved Bridge Test

```
1  mplist ← ∅;
2  for i=1 to N do
3      q_f ← Sample;
4      if not CollisionFree(q_f) then
5          q_r ← Sample;
6          q_s ← Steer(q_f, q_r, l);
7          if not CollisionFree(q_s) then
8              q_m ← (q_f+q_s)/2;
9              if CollisionFree(q_m) then
10                 if OrthogonalTest (q_f, q_s) then
11                     mplist←mplist ∪ (q_m);
12                     if Size(mplist) > maxSize then break;
13 iplist←ClusterAnalysis(mplist);
14 return iplist;
```

The procedures *Sample* and *Steer* return two endpoints (q_f and q_s) in the obstacle. If the midpoint (q_m) of the segment is in collision-free and the "bridge" passes the orthogonal test, the midpoint (q_m) will be collected to *mplist*. The loop will be terminated either reaching the maximum number of *mplist* or reaching the maximum iteration number. Then, the *Cluster Analysis* will be executed to generate unique identification points (*IP*) for each narrow passage and return them to *iplist*. The issues of the length *l* and *maxSize* were mentioned in [20].

Note that, the optimal K value cannot be clear in some scenarios, but it is also optional to choose other value near the optimal one since their impacts on search performance are almost the same, which will be discussed in Section V.

III. LGM-BRRT∗ Algorithm

This section presents a local-trees generated strategy and then proposes the LGM-BRRT∗ algorithm.

A. Local Trees Generated Strategy

This part first discusses an embryonic idea about local trees generated strategy, and inspired by its strengths and limitations,

a novel search strategy based on the triggering criterion is proposed.

1) An Embryonic Idea: After identifying narrow passages, an effective strategy for exploration is needed. One naive strategy is to generate local trees rooted at each *IP* and employ BRRT∗ to heuristically expand one by one, called Segmented Continuous Bidirectional RRT∗ Search (SC-BRRT∗) here. The brief flow is as below. Firstly, identify narrow passages and sort all *IPs* with the start and goal node (x_s, x_g) in the search order. Then, BRRT∗ is employed to heuristically expand one by one, e.g., x_s and IP_1 construct the first bidirectional exploration, IP_1 and IP_2 construct the next exploration. The last exploration is IP_n and x_g. The algorithm will be terminated until reaching the goal node. Fig.3 is one possible processes of search.

Experiments in Section V show that SC-BRRT∗ is remarkable to find a quick solution in specific cluttered environment, where the search order of *IPs* can be arranged automatically and there are no redundant *IPs*. Nevertheless, it has some obvious limitations, (1) The search order of *IPs* is needed as the prior knowledge, otherwise the performance will degrade dramatically, e.g. such scene as shown in Fig.4a and its poor performance is shown in Fig.9b; (2) It would take much more iterations, if there are some redundant *IPs*, e.g., the *IPs* in orange color shows in Fig.4b, and its poor performance is shown in Fig.10b; (3) It loses the asymptotic optimality. Thus, those limitations need to be improved in next.

2) The Strategy Based on Triggering Criterion: The idea of local guidance is remarkable to get out hard-reached regions. So, the strategy applies local trees to guide the search. Consequently, there are two important issues raised, (1) When to grow a local tree and how to ensure efficient memory utilization; (2) How to accelerate efficiency and guarantee asymptotic optimality.

As for the first issue, a distance criterion is employed to generate local trees appropriately. The distance triggering criterion is the Euclidean distance between IP_i and the latest sample x_{new}, which should be less than the pre-specified triggering length (L_{tri}), i.e., $\{\|IP_i - x_{new}\| \leq L_{tri}\}$. If more than one *IP* is triggered at the same time, the nearest one will be selected. There are some related constraint rules, which are the compensation of the distance trigger criterion. They are as following: (1) To ensure efficient memory utilization, only one local tree can exist at a time; (2) To avoid repeating search same sub-region, the same *IP* cannot be the root node twice; (3) To accelerate the success rate and ensure asymptotic optimality, local trees will be merged into nearer global tree after local search done. However, as for the parameter L_{tri}, it's not easy to determine an optimal solution for general problems. The value chosen here is 3 to 4 times the width of the main narrow passages. And a simple analysis can be found in Section V.

So far, the issues above have been addressed. Under this strategy, the search order of *IPs* can be ignored, and some redundant local trees won't be spawned. Additionally, merging local trees into T_a/T_b can make it inherit the asymptotically optimal property from BRRT∗.

B. LGM-BRRT∗ Algorithm

This part presents a variant of BRRT∗, which is asymptotically optimal and compared with the computation time of BRRT∗. It is called Locally Guided Multiple Bi-RRT∗, which is implemented as shown in Algorithm 2.

Firstly, identify narrow passages via *Bridge Test* (line 1), which can be also stored as prior knowledge to reuse. Then, two global trees are generated, denoted T_a and T_b. A vertex x_{rand} is randomly sampled (line 2-4). After insertion and rewiring of the sample into the selected tree T_a (line 5-12), local search between x_{new} and the nearest IP_i will be triggered, if meeting the triggering criterion (line 13-16). After that, the nearest vertex $x_{connect}$ from the opposing tree T_b to x_{new} is searched, and then the *Connect* attempts to connect to the vertex on the other tree and update its current best solution (line 16-19). Fig.5 is one possible processes of search. These operations are just like BRRT∗ described by Jordan and Perez [12] with slight modifications incorporating *Bridge Test* and *Local Search*. Additional procedures and modifications to accelerate efficiency for search are outlined below.

Localsearch employed here (Algorithm 3), is exactly the same as BRRT∗ besides heuristic *Connect* strategy (line 10) and *Merge* mechanism (line 11-14). In local search planner, the procedure *ConnectLocal* is similar to RRT-Connect [5], besides a non-greedy heuristic strategy is adopted. When local search has done, the local tree will be merged into a global tree, and remove the triggered *IP* in *iplist* (line 11-14 in Algorithm 2).

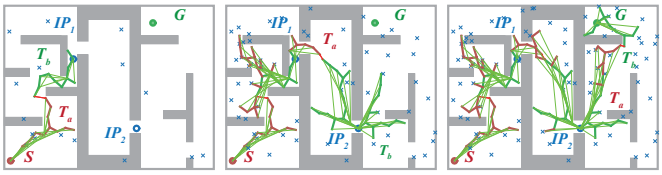


Fig. 3: The search strategy of SC-BRRT∗. (a) P_s and IP_1 construct the first bidirectional RRT∗ search. (b) IP_1 and IP_2 construct the next. (c)IP_2 and P_g construct the last. (Bold line in color red, green and blue denote T_a, T_b and local tree respectively. Green thin lines denotes local cheaper path.)

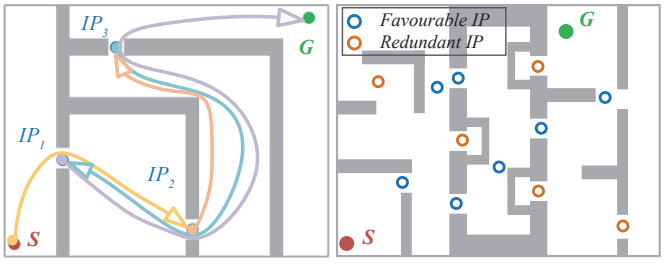


Fig. 4: Limitations of SC-BRRT∗. (a) Duplicate search because of unknown search order. (b) Duplicate search because of some redundant *IPs*.

IV. Analysis

A. Probabilistic Completeness

The probabilistic completeness and the exponential convergence of RRT and its bidirectional version are clearly sure, proved in [5] and [13]. The optimal variants of RRT, e.g., RRG,

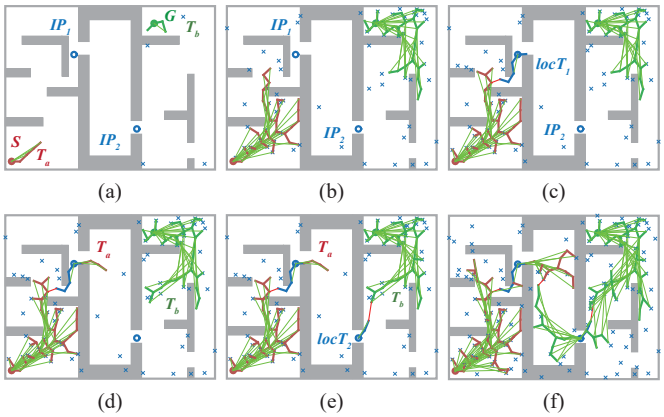


Fig. 5: The Search strategy of LGM-BRRT∗. (a) Two global trees (T_a, T_b) are generated. (b) T_a triggered IP_1, and $locT_1$ is generated. (c) T_a merged with $locT_1$ into one global tree when local search done. (d) T_b triggered IP_2, and $locT_2$ is generated. (e) T_b merged with $locT_2$ into one global tree. (f) T_a reached T_b and an initial solution was found.

Algorithm 2:LGM-BRRT∗ Algorithm(x_{init}, x_{goal})

$iplist \leftarrow$ *BridgeTest*( );
$V = \{x_{init}, x_{goal}\}$; $E = \emptyset$; $T_a \leftarrow \{x_{init}, V\}$; $T_b \leftarrow \{x_{goal}, V\}$;
for *i*=1 to *N* **do**
  $x_{rand} \leftarrow$Sample;
  $x_{nearest} \leftarrow$Nearest(T_a, x_{rand});
  $x_{new} \leftarrow$Steer($x_{nearest}$, x_{rand});
  if CollisionFree($x_{nearest}$, x_{new}) **then**
    $x_{near} \leftarrow$Near(T_a, x_{new});
    $L \leftarrow$ListSort(x_{new},x_{near})
    $x_{parent} \leftarrow$SelectBestParent(T_a, x_{near}, L);
    $V \leftarrow V \cup (x_{new})$; $E \leftarrow E \cup (x_{parent}, x_{new})$;
    $E \leftarrow$Rewire(E, x_{near}, x_{new})
    $x_{connect} \leftarrow$Nearest(T_b, x_{new});
    $d \leftarrow$Pdist(*iplist*, x_{new});
    if $d<L_{tri}$ **then**
      $(V_a, E_a) \leftarrow$LocalSearch(T_a, x_{new}, *iplist*);
      $(c_{sol}, \sigma_{sol}) \leftarrow$Connect($T_b$, x_{new}, $x_{connect}$);
      if $c_{sol} \neq \emptyset$ and $c_{sol}<c_{best}$ **then**
        $\sigma_{best} \leftarrow \sigma_{sol}$; $c_{best} \leftarrow c_{sol}$;
  Swap (T_a, T_b)
Return (T_a, T_b) = (V, E).

Algorithm 3:Local Search(T, x_{new}, *iplist*)

$(V_a, E_a) \leftarrow T$; $E_b = \emptyset$; $V_b = IP_i$; $T_b \leftarrow \{V_b, E_b\}$;
for *i*=1 to *N* **do**
  $x_{rand} \leftarrow$Sample;
  $x_{nearest} \leftarrow$Nearest(T_a,x_{rand})
  $x_{new} \leftarrow$Steer($x_{nearest}$, x_{rand});
  $x_{parent} \leftarrow$SelectBestParent(T_a, x_{near}, L);
  $V \leftarrow V \cup (x_{new})$; $E \leftarrow E \cup (x_{parent}, x_{new})$;
  $E \leftarrow$Rewire(E, x_{near}, x_{new})
  $x_{connect} \leftarrow$Nearest(T_b, x_{new});
  $(E_b, V_b) \leftarrow$ConnectLocal($T_b, x_{connect}, x_{new}$)
  if isreached **then**
    $T_a \leftarrow$Merge($T_a, locT_i$);
    iplist$\leftarrow$Remove(*iplist*(IP_i));
    Return $T_a = (V, E)$
Swap (T_a, T_b);
Return NULL.

RRT∗, inherit the same properties [23]. Additionally, Matthew Jordan and Alejandro Perez have also verified that BRRT∗ inherits the same property as well in [12]. Therefore, LGM-BRRT∗ follows directly from the probabilistic completeness of bidirectional RRT*.

B. Asymptotic Optimality

Theorem (Asymptotic optimality of LGMB-RRT∗) If $\gamma_{LGM-BRRT*} > \gamma^* := (2(1+1/d))^{(1/d)}(\mu(X_{free}/\zeta_{\mathfrak{B}}))^{1/d}$, LGM-BRRT∗ is asymptotically optimal.

The sketch proof of this theorem follows directly from Theorem 38 in [4]. The asymptotic optimality of BRRT∗ is proven in Theorem 2 in [12]. As a variant of BRRT∗, LGM-BRRT∗ is similar to operate analogously to BRRT∗ except two procedures, *local search* and *merge*. In particular, *local search* employs BRRT∗ in the local domain to accelerate local exploration. When local exploration has done, the local tree will be merged into near global tree as branches by *merge* procedure. Then two global trees reconstitute global BRRT∗ search. Hence, LGMB-RRT∗ inherits the properties of asymptotic optimality from BRRT∗. Therefore, the $\mathbb{P}\left(\left\{\lim_{i\to\infty} c_i^{LGM-BRRT*} = c^*\right\}\right) = 1$ result follows directly from Lemmas 56, 71, and 72 by Karaman and Frazzoli [4].

C. Computational Complexity

In this part, the computational complexity of the proposed algorithm is compared with that of BRRT∗. Let M_i^{BRRT*} be the random variable that denotes the number of steps executed in *i*-th iteration by BRRT∗ and LGM-BRRT∗ respectively.

Theorem (Computational ratio of BRRT∗ and LGM-BRRT∗): There exists a constant $\beta \in +$ such that $\limsup_{i\to\infty} \mathbb{E}\left[\frac{M_i^{LGM-BRRT*}}{M_i^{BRRT*}}\right] \leq \beta$.

The sketch proof of this theorem follows directly from Theorem 18 in [4]. The extra procedures employed in LGM-BRRT∗ are *Bridge-Test*, *Cluster Analysis* and *Local Search*. Among them, *Bridge Test*, *Cluster Analysis* are executed in the pre-processing phase, and the steps they execute will not exceed the preset values. As for procedure *Local Search*, it operates analogously to BRRT∗ and incurs almost the same computational cost of BRRT∗. Hence, based on a theorem by Karaman and Frazzoli [4], it can be concluded that the computational cost of the proposed algorithm is also inherited.

V. Experimental Results

A. Experimental Setup

To verify the performance of the proposed algorithm, a series of simulations were carried with LGM-BRRT∗, BRRT∗. SC-BRRT∗ is also tested, in order to verify whether the search strategy of LGM-BRRT∗ overcomes the shortcomings or not. All environment and parameters were kept the same, and simulations were carried out on an Intel core i3 processor at 3.9 GHz with 8 GB RAM based on MATLAB®. The number of runs was set to 100. Some of the experiment results are summarized in TABLE I, including the average time consumed by *Bridge Test* (t_{bt}), the average search time to find an initial solution (t_s), the average number of nodes and calls of collision checking procedure (n_{nodes}, n_{cc}). In TABLE I, *Time* is defined as $(t* - t_{Bi})/t_{Bi} \times 100\%$, and *Efficiency* is defined as $(1/t_{Bi} - 1/t*)/t_{Bi} \times t_{Bi} \times 100\%$, where $t* = sum(t_{bt} + t_s)$, $* \in \{SC, LGM\}$.

B. Effect of Implementation Parameters

As for scenarios with high identification of narrow passages, e.g., scenes similar to Fig.4a and Fig.8a, the optimal K value is clear via EM. However, optimal K (i.e. the number of *IPs*) may vary with different thresholds in EM in scenarios similar to Fig.3b and Fig.10. Thus, the impacts of K on both the location of *IPs* and search performance were tested. It is found that when K is close to or slightly larger than the optimal, the most influential *IPs* can be all identified (see Fig.6), and their impacts on search performance are almost the same (see Fig.7a). Therefore, choosing a slightly larger value will be more guaranteed.

The parameter L_{tri} influences when to generate local trees and the order that the local tree is triggered. Therefore, the impact of a wide range of L_{tri} on performance was tested. It can be found from Fig.7b that the efficiency will drop sharply if the trigger length is too small. However, the differences in performance are inconsistent with the increase of length. The reason is that the triggered order of *IPs* is based on distance priority. So it is mainly changed by the randomness of x_{new}. In general, determining the optimal L_{tri} value for general problems is not easy. The value chosen here is 3 to 4 times than the width of the main narrow passages.

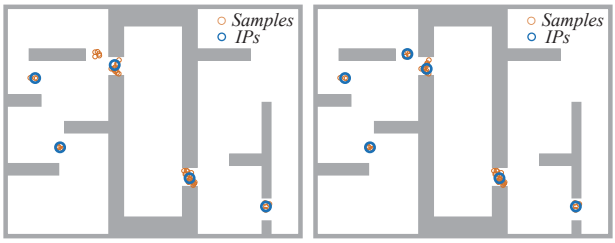
C. Performance of LGM-BRRT∗

Fig.8 demonstrates the performance of LGM-BRRT∗ and BRRT∗ in both 2D and 3D cluttered environment. Particularly, the search order of *IPs* can be arranged in advance, and there are no redundant *IPs*. So, they are also suitable for SC-BRRT∗, of which the analysis will be described in the next paragraph. As for scenario #1 shown in Fig.8a and Fig.8b, LGM-BRRT∗ takes lesser iterations and time (n_{nodes}=115, t_{avg}=7.75s) to find an initial solution compared to BRRT∗(n_{nodes}=537, t_{avg}=59.6s), since BRRT∗ is trapped around narrow passages. Moreover, as summarized in TABLE I, the running time is reduced by 87% and the efficiency is increased about 6.6 times. In another scenario #2, shown in Fig.8c and Fig.8d, the gap in performance is obvious as well. e.g., LGM-BRRT∗ takes lesser iterations and time (n_{nodes}=135, t_{avg}=14.1s) compared to BRRT∗ (n_{nodes}=444, t_{avg}=71.2s); Running time is reduced by 80% and the efficiency is increased about 4 times. More than that, the result of 20 extra scenes were shown in Fig.12, including both 2D and 3D scenes, and a various types of scenarios, e.g., simple and complex cluttered scenarios. Obviously, a similar trend is followed by all scenarios, i.e., the performance of LGM-BRRT∗ to converge to the initial solution is remarkable. The gap would be more obvious in more cluttered scenarios.

As for SC-BRRT∗ in scenario #1 and #2, the performance is similar to LGM-BRRT∗ or even better, according to TABLE I. The reason is that SC-BRRT∗ don't employ *merge* procedure and thus fewer nodes of exploring tree reduce retrieval time. Meanwhile, it reveals that segmental exploring fits such specific scenarios. However, for other unsuitable scenarios, its performance will degrade dramatically, as shown in Fig.9 and Fig.10. However, LGM-BRRT∗ inherits its strengths and overcomes those problems.

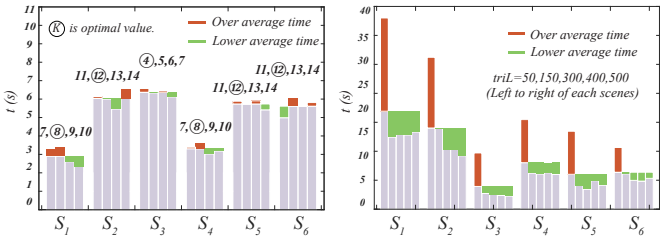
Fig.9 and Fig.10 show the exploration process in scene with unknown search order of *IPs* (#3) and redundant *IPs* (#4), respectively. Obviously, as for SC-BRRT∗, it will duplicate search because of unknown search order of *IPs*, and thus consume much more exploration time, even no significant improvement, e.g. Compared to BRRT∗ in scenarios #3, the time and the efficiency is hardly improvement. However, as summarized in TABLE I, LGM-BRRT∗ takes much fewer iteration as well as time compared to the others, which reveals that LGM-BRRT∗ overcomes those limitations of SC-BRRT∗ by the proposed strategy.

Fig.11 compares the success rate in scenarios #1∼#6. Note that, scene in Fig.2 is #5 and scene in Fig.3 is #6. Let n denotes the times of successful search within the specified time T, letting N is the total number of test. Then the success rate at time T is defined as $S_T = \sum_{t=0}^{T} n \backslash N$. It can be seen that, as for LGM-BRRT∗, the performance on specific scenarios is similar to SC-BRRT∗, and it remains a higher success rate than that of BRRT∗. The advantage would be more obvious in a more cluttered scenario.



(a) K equals 5 (b) K equals 6

Fig. 6: The impact of different K on the location of *IPs*. By comparing (a), (b), and Fig.2c, it can be found all the most influential *IPs* were all identified.

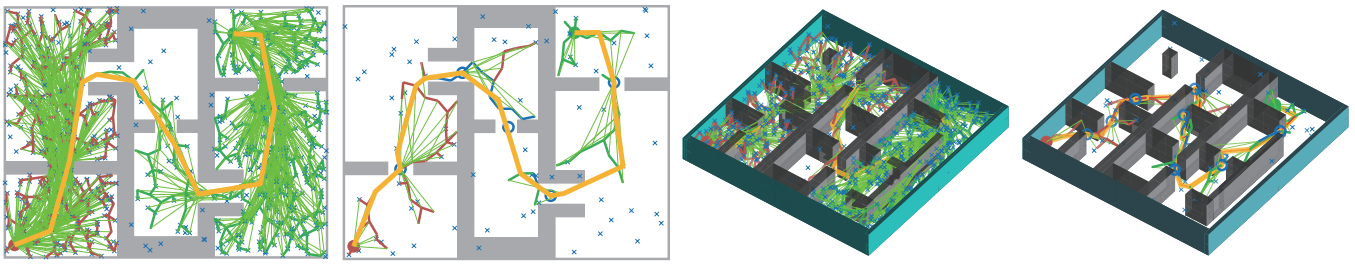


(a) (b)

Fig. 7: The effect of parameter K value (a) and L_{tri} (b)

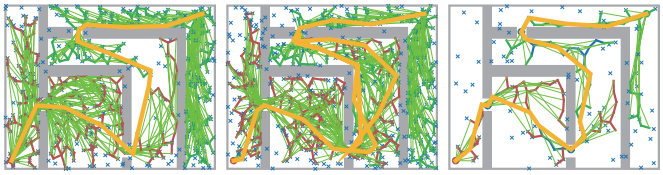
VI. Conclution And Future Work

BRRT∗ is computationally efficient for many motion planning problems. Nevertheless, it's not easy to rapidly find one initial solution in a clustered environment with narrow passages. LGM-BRRT∗ algorithm outlines this issue and provides a rapidly converges strategy by incorporating an improved bridge-test and a novel search strategy. It is demonstrated by both experiments and analysis that, (1) it ensures accelerated success rate as well as less time to find an initial solution in a clustered environment; (2) it utilizes less memory occupation by a fewer number of iterations compared with BRRT∗; (3) it is easy for implementation as well.



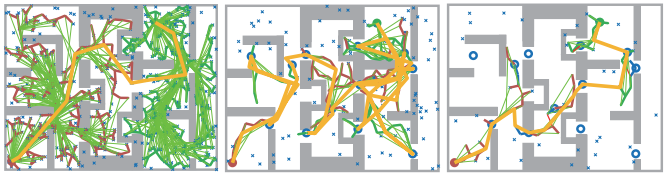
(a) BRRT*(n_{nodes}=537, t_{avg}=59.6s) (b) LGM-BRRT*(n_{nodes}=115, t_{avg}=7.75s) (c) BRRT*(n_{nodes}=444, t_{avg}=71.2s) (d) LGM-BRRT*(n_{nodes}=135, t_{avg}=14.1s)

Fig. 8: Performance of BRRT$*$ and LGM-BRRT$*$ on cluttered Scenarios #1 and #2.



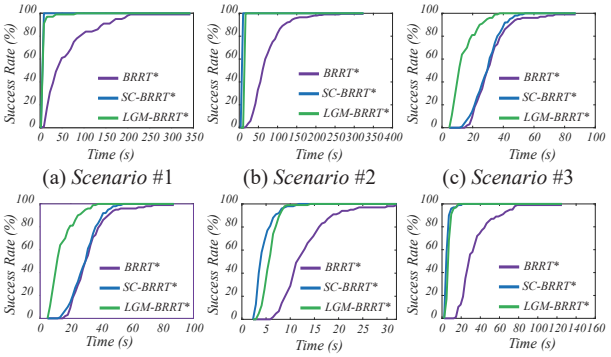
(a) BRRT* (n_{nodes}=384, t_{avg}=31.6s) (b) SC-BRRT* (n_{nodes}=457, t_{avg}=30.2s) (c) LGM-BRRT* (n_{nodes}=207, t_{avg}=14.6s)

Fig. 9: The performance on Scenario #3 with unknown search order of *IPs*



(a) BRRT* (n_{nodes}=259, t_{avg}=22.0s) (b) SC-BRRT* (n_{nodes}=295, t_{avg}=14.6s) (c) LGM-BRRT* (n_{nodes}=106, t_{avg}=6.4s)

Fig. 10: The performance on Scenario #4 with redundant *IPs*.



(a) *Scenario* #1 (b) *Scenario* #2 (c) *Scenario* #3

(d) *Scenario* #4 (e) *Scenario* #5 (f) *Scenario* #6

Fig. 11: The success rate on scenario #1 ∼#6.

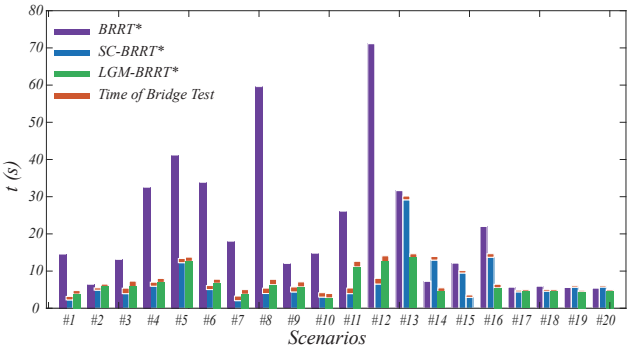


Fig. 12: The average search time of initial solution of extra 20 scenes.

TABLE I: Performance of different methods

Scenario	Algorithm	t_{bt}/s	t_s/s	n_{nodes}	n_{cc}	**Time** and **Efficiency**	*Time*	*Efficiency*
#1	BRRT*	-	59.6	537	17409			
	SC-BRRT*	1.45	4.0	109	740	SC-BRRT* *vs.* BRRT*	91%	999%
	LGM-BRRT*	1.45	6.3	115	1809	LGM-BRRT* *vs.* BRRT*	**87%**	**665%**
#2	BRRT*	-	71.2	444	11762			
	SC-BRRT*	1.53	6.5	110	595	SC-BRRT* *vs.* BRRT*	89%	784%
	LGM-BRRT*	1.53	12.6	135	1851	LGM-BRRT* *vs.* BRRT*	**80%**	**405%**
#3	BRRT*	-	31.6	384	9128			
	SC-BRRT*	1.02	29.2	457	7564	SC-BRRT* *vs.* BRRT*	5%	5%
	LGM-BRRT*	1.02	13.6	207	3724	LGM-BRRT* *vs.* BRRT*	**54%**	**116%**
#4	BRRT*	-	22.0	259	6713	SC-BRRT* *vs.* BRRT*	33%	50%
	SC-BRRT*	0.94	13.7	295	2918			
	LGM-BRRT*	0.94	5.5	106	1526	LGM-BRRT* *vs.* BRRT*	**71%**	**242%**
#5	BRRT*	-	12.1	156	3577	SC-BRRT* *vs.* BRRT*	52%	110%
	SC-BRRT*	1.34	4.4	108	920			
	LGM-BRRT*	1.34	5.8	106	1700	LGM-BRRT* *vs.* BRRT*	**41%**	**70%**
#6	BRRT*	-	33.9	384	11063	SC-BRRT* *vs.* BRRT*	82%	455%
	SC-BRRT*	1.08	5.0	131	994			
	LGM-BRRT*	1.08	6.7	127	1970	LGM-BRRT* *vs.* BRRT*	**77%**	**334%**

REFERENCES

[1] W. Wang and 2009. ICMA 2009. International Conference on Yan Li BT - Mechatronics and Automation, "A multi-RRTs framework for robot path planning in high-dimensional configuration space with narrow passages," 2009.

[2] 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, 2002.

[3] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[4] S. Karaman and E. Frazzoli, "Sampling-based Algorithms for Optimal Motion Planning," Int. J. Robot. Res., vol. 30, pp. 846–894, 2011.

[5] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in Proceedings 2000 ICRA. Millennium

Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065), 2000, vol. 2, pp. 995–1001.
[6] B. Akgun and M. Stilman, "Sampling heuristics for optimal motion planning in high dimensions," in 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2011, pp. 2640–2645.
[7] M. Elbanhawi and M. Simic, "Sampling-based robot motion planning: A review," Ieee access, vol. 2, pp. 56–77, 2014.
[8] N. M. Amato and Y. Wu, "Randomized roadmap method for path and manipulation planning," in Proceedings - IEEE International Conference on Robotics and Automation, 1996, vol. 1, pp. 113–120.
[9] V. Boor, M. H. Overmars, and A. F. van der Stappen, "Gaussian sampling strategy for probabilistic roadmap planners," in Proceedings - IEEE International Conference on Robotics and Automation, 1999, vol. 2, pp. 1018–1023.
[10] A. Yershova, L. Jaillet, T. Siméon, and S. M. LaValle, "Dynamic-domain RRTs: Efficient exploration by controlling the sampling domain," 2005.
[11] G. S. Chirikjian, H. Choset, M. Morales, and T. Murphey, "Algorithmic Foundation of Robotics VIII," vol. 57, 2010.
[12] M. Jordan and A. Perez, "Optimal bidirectional rapidly-exploring random trees," 2013.
[13] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," Int. J. Rob. Res., vol. 20, no. 5, pp. 378–400, 2001.
[14] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065), 2000, vol. 2, pp. 995–1001.
[15] M. Strandberg, "Augmenting RRT-planners with local trees," in IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004, 2004, vol. 4, pp. 3258–3262.
[16] M. Otte and N. Correll, "Path planning with forests of random trees: Parallelization with super linear speedup," Dep. Comput. Sci. Univ. Color. Boulder, Tech. Rep. CU-CS, pp. 1011–1079, 2011.
[17] W. Wang, L. Zuo, and X. Xu, "A Learning-based Multi-RRT Approach for Robot Path Planning in Narrow Passages," J. Intell. Robot. Syst. Theory Appl., vol. 90, no. 1–2, pp. 81–100, 2018.
[18] J. Zhong, D. Xing, and H. Zhao, "Narrow Passage Identification Based Robot Path Planning," Int. J. Adv. Comput. Technol., vol. 4, no. 21, 2012.
[19] Z. Sun, D. Hsu, T. Jiang, H. Kurniawati, and J. H. Reif, "Narrow passage sampling for probabilistic roadmap planning," IEEE Trans. Robot., vol. 21, no. 6, pp. 1105–1115, 2005.
[20] W. Wang, Y. Li, X. Xu, and S. X. Yang, "An adaptive roadmap guided Multi-RRTs strategy for single query path planning," in 2010 IEEE International Conference on Robotics and Automation, 2010, pp. 2871–2876.
[21] D. Arthur and S. Vassilvitskii, "K-means++: The advantages of careful seeding," Proc. Annu. ACM-SIAM Symp. Discret. Algorithms, vol. 07-09-January-2007, pp. 1027–1035, 2007.
[22] P. Bholowalia, "EBK-Means : A Clustering Technique based on Elbow Method and K-Means in WSN," vol. 105, no. 9, pp. 17–24, 2014.
[23] S. Karaman and E. Frazzoli, "Sampling-based Algorithms for Optimal Motion Planning," Int. J. Robot. Res., vol. 30, pp. 846–894, 2011.