SARRT: A Structure-Aware RRT-Based Approach for 2D Path Planning*

Xuefeng Chang[1†] and Yanzhen Wang[1†] and Xiaodong Yi[1] and Nong Xiao[1]

***Abstract*— Motion/path planning remains one of the most important research topics in robotics for decades, since mobility is a defining characteristics of robots. Sampling-based approaches have proven to be effective for problems with complex constraints and high dimensionality. Specifically, Rapidly-exploring Random Tree (RRT) is one of the most popular sampling-based algorithms. However, it suffers from problems such as unstable performance and suboptimal results.**

This paper presents a novel RRT variant, namely, Structure-Aware RRT (SARRT), which utilizes a physically-based costmap to bias the tree growth to regions closer to the goal. Instead of typical distance metrics, such as Euclidean and Manhattan distances, the cost function is based on a simulated diffusion process and is able to reflect the structure of the free space and problem settings. Furthermore, a Laplacian smoothing step is performed on the resulting path to improve the smoothness of the path. Experimental results on 2D path planning problems show the effectiveness of SARRT, in terms of both algorithm runtime and resulting path quality.

I. INTRODUCTION

Mobility is one of the defining characteristics of robots. Nevertheless, robots needs to move wisely according to some specific criteria. Therefore, motion/path planning becomes a very important topic in robotics and has received significant research effort for decades. In the past decades approaches based on Rapidly-expoloring Random Trees (RRT) [1] have been among the most popular sampling-based planning approaches. RRTs explore the configuration space very efficiently, since they implicitly exploit the Voronoi property to direct the exploration to unexplored areas. RRT-based approaches work well in practice, and are able to find a feasible solution to motion/path planning problems very quickly, even with complex constraints and high dimensionality. They also possess theoretical guarantees such as probabilistic completeness. However, the performance of RRT-based approaches is not stable due to their stochastic essense. Sometimes, results obtained by RRT-based approaches are far from optimal, since path cost is not considered in basic RRTs. Existing solutions include biasing the tree growth according to different criteria based on continuous costmaps. They are effective in speeding up the planning process and making the resulting paths possess some specific properties such as high obstacle clearance. However, for moderately-constrained problems, especially those with "dead ends" in the maps (with an example shown in Fig. 1(a)), such solutions may exhibits degraded performance, since the costmaps they utilize cannot effectively reflect the structural information of the obstacles.

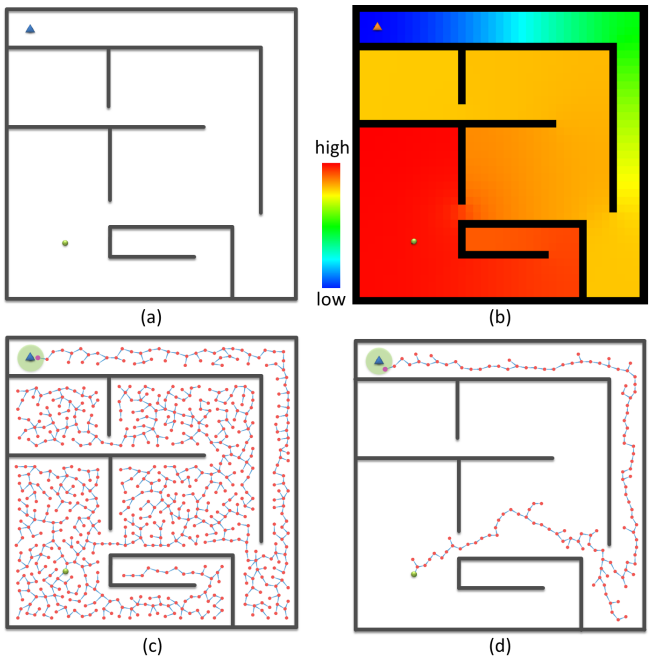

Fig. 1. A typical 2D path planning problem on the "Maze" map, with initial configurations shown in green dots and goal configurations in triangles. (a) The problem setting. (b) The color-coded structure-aware costmap. (c) A random tree generated by the basic RRT approach. (d) A random tree generated by SARRT.

In this paper, we propose a Structure-Aware RRT-based approach (SARRT) for path planning. It is inspired by the research effort of biasing the RRT growth using costmaps. The costmap used in SARRT is computed by solving a diffusion equation and is actually a potential field that approximates the distance to goal modulated by the presence of obstacles. Under the guide of the structure-aware costmap, SARRT tends to grow the tree in a gradient descent manner instead of the pure exploratory manner, as shown in Fig. 1(d). Compared to basic RRT-based planners, SARRT provides significant improvement in time efficiency and the resulting paths are statistically of higher quality. Specifically, contributions of this paper are listed as follows.

1) We propose a generic RRT-based planning framework to get a costmap incorporated to bias the tree growth

*This work was partially supported by National Science Foundation of China under Grant No. 61303185, and University Grants from NUDT under No. 434513322532 and 434513322412.

†Xuefeng Chang and Yanzhen Wang are joint first authors.

[1]The authors are with the State Key Laboratory of High Performance Computing (HPCL), School of Computer, National University of Defense Technology (NUDT), Changsha, Hunan 410073, China changxuefengcn@163.com; yanzhenwang@hotmail.com; xdong_yi@163.com; xiao-n@vip.sina.com

according to some specific strategies.
2) We introduce a costmap defined over the configuration space (shown in Fig. 1(b)), which is able to reflect the structure of the free space, and demonstrate its effectiveness in 2D path planning problems.
3) We propose to use Laplacian smoothing as a postprocess step to locally optimize the resulting global path.

The remaining part of this paper is structured as follows. Section II briefly describes the basic RRT approach, and reviews some important RRT variants. Section III presents the generic RRT-based planning framework, the structure-aware costmap, as well as the local path optimization in the SARRT approach. Section IV demonstrates the effectiveness of SARRT in 2D path planning problems by experimental results for typical problem configurations. Finally, Section V concludes the paper and discusses about limitaitons of SARRT and potential future work.

II. RELATED WORK

In this section, we first briefly describe the basic idea of the original RRT approach, and then review typical RRT variants proposed to improve the performance of basic RRT-based planning approaches. Note that the aim here is not to present a comprehensive survey of RRT-based planning approaches, and only the research works most closely related to SARRT are discussed.

A. Basic RRT

The basic RRT algorithm [1] grows a tree based on the Voronoi property, biasing the search towards unexplored regions of free configuration space. As shown in Algorithm 1, the search process for global path is actually a process of incremental construction of a single random tree rooted at the initial configuration, with the nodes representing valid configurations and edges representing feasible transitions. The random tree is initialized with the initial configuration x_{init} as the root and only node. In each iteration of RRT algorithm, the planner draws a sample x_{rand} from a uniform distribution defined over the entire configuration space. A new configuration x_{new} is generated by extending $x_{nearest}$, which is the nearest configuration to x_{rand} in the current tree, towards x_{rand} by a predefined step size l. If x_{new} is a valid configuration and the transition from $x_{nearest}$ to x_{new} does not violate any constraints, such as obstacles and specific dynamics, then both the node x_{new} and the edge $(x_{nearest}, x_{new})$ are added into the tree. This process iterates until the distance between x_{new} and the goal configuration x_{goal} is less than a predefined threshold, δ. A feasible path from x_{init} to x_{goal} can be extracted by following the corresponding edges edges in the random tree. A commonly adopted trick for the basic RRT algorithm is to use x_{goal} rather than x_{rand} as the target towards which the tree is grown in a certain portion of iterations.

Despite the simplicity, RRT-based planners work well in practice, and can be easily applied to problems with complex constraints. Compared with grid-based planning approaches, RRT-based planners can handle high-dimensional planning problems robustly and efficiently. For example, they are widely applied in motion planning for robotic arms. However, one major weakness of the basic RRT-based planner is that it does not take path cost into account. This can lead to low-quality solutions that are far from optimal.

input : The initial point x_{init}, the goal point x_{goal}, step size l
output: A new path P from the initial point to the goal point

```
1  T.initialize(x_init);
2  while time_remaining() do
3      x_rand ← rand_state();
4      x_nearest ← nearest_neighbor(T,x_rand);
5      x_new ← new_state(x_nearest,l);
6      if x_new then
7          T.add_node(x_new);
8          T.add_edge(x_new,x_nearest);
9          if distance(x_new,x_goal)≤ δ then
10             P←get_path(T,x_new,x_init);
11             return P;
12         end
13     end
14 end
15 return P;
```

Algorithm 1: The basic RRT algorithm

B. RRT variants

In order to improve the performance of RRT-based planners in different application scenarios, many variants have been proposed to modify the behavior of RRTs using different strategies.

In order to improve the time efficiency, RRT-Connect uses two trees rooted at the inital and goal configurations respectively, each exploring space around them and also advancing towards each other [2]. RRT-blossom introduces an implicit flood-fill-like mechanism for escaping local minima in highly constrained problems [3].

The underlying sample generation process also affects the behavior of RRT-based planning approaches. Therefore, some RRT variants use more elaborate sampling strategies instead of the random uniform sampling in original RRT. Dynamic-domain RRT adapts the sampling domain to the problem and shows significant improvement over existing RRT-based planners, with performance orders of magnitude better on many problems [4]. RESAMPL adaptively chooses different sampling strategies for RRT according to the local properties of different regions [5]. Recently, Park et al. proposed Poisson-RRT, which uses the maximal Poisson-disk sampling scheme for tree expansion [6].

Another common extension is to bias the tree growth in RRT-based planners, in order to speed up the exploration or to obtain some desirable properties in the solution. Heuristically-guided RRT (hRRT) provides a probabilistic

implementation of heuristic search concepts to obtain better solutions with less cost in variable cost domains [7]. Transition-based RRT (T-RRT) combines exploration with transition tests from stochastic optimization to compute low-cost paths that follow valleys and saddle points of the configuration-space costmap [8]. Recently, Medial Axis RRT (MARRT) was proposed to bias tree exploration to the medial axis of free space to improve obstacle clearance for a safer path [9].

There are also other research efforts based on RRTs, which aim at providing optimality guarantee [10][11], or exploiting the flexiblity of RRT-based planner to solve problems with complex constraints [12][13]. Moreover, Rapidly-exploring Random Graphs (RRG) generalize the tree structure of RRT to a graph to enable a greater exploration [14].

III. METHOD

A. Planner framework

A generic algorithm framework, as described in Algorithm 2 and illustrated in Fig. 2, is proposed for RRT-based planners with biased tree growth based on a costmap. Specifically, a set of sample configurations C_{rand}, intead of only one sample in basic RRT, are drawn from a uniform distribution over the configuration space in each iteration of tree growth. By finding the corresponding nearest nodes in the current tree to the configurations in C_{rand}, another set of configurations $C_{nearest}$ can be formed. Consequently, a new set of configurations, C_{new}, for potential tree growth, is generated by extending the nodes in $C_{nearest}$ towards the corresponding configurations in C_{rand} by a fixed distance, l.

Suppose that a cost function, $\phi(\cdot)$, is defined over the configuration space. The configuration in C_{new} with the lowest cost function value is chosen for potential extension operation. One of the most commonly used cost functions is inspired by A* search and consists of two terms:

$$\phi(x) = G(x_{init}, x) + H(x_{goal}, x) \tag{1}$$

where $G(x_{init}, x)$ is the past path cost, which can be easily obtained by using information about node depth and the predefined extension distance, l. $H(x_{goal}, x)$ is a heuristics for the future path cost, and popular choices include Euclidean distance and Manhattan distance between the current configuration and the goal. Note that the cost function $\phi(\cdot)$ in the algorithm framework is very generic and could have a variety of different implementations.

For the purpose of avoiding exploring the areas which have already been visited by the random tree, a regression avoidance mechanism is also integrated in the generic planner framework. Whenever adding the configuration with the lowest cost in C_{new} into the current tree is identified as a regression, the corresponding tree growth operation will be aborted and the planner will directly start the next iteration. A typical indicator for regression is defined as follows [3].

$$|x_{new} - x_{nearest}| < \min_{x_n \neq x_{nearest}} \{|x_{new} - x_n|\} \tag{2}$$

where x_{new} is the node with the lowest cost in C_{new}, $x_{nearest}$ is the closest node to x_{new} in the current tree, and x_n represents any node in the current tree. Note that there can also be other choices for the indicator of regression.

input : The initial point x_{init}, the goal point x_{goal}, array size s, step size l
output: A new path P from the initial point to the goal point

```
T.initialize(x_init);
while time_remaining() do
    for i ← 1 to s do
        x_rands[i] ← rand_state();
        x_nearests[i] ← nearest_neighbor(T, x_rands[i]);
        x_news[i] ← new_state(x_nearests[i], l);
    end
    x_new, x_nearest ← lowest_cost(x_news[1..s]);
    if regression(T, x_new, x_nearest) then
        T.add_node(x_new);
        T.add_edge(x_new, x_nearest);
        if distance(x_new, x_goal) ≤ δ then
            P ← get_path(T, x_new, x_init);
            return P;
        end
    end
end
return P;
```

Algorithm 2: The generic framework for biased RRT-based planners.

B. Structure-aware cost function

As mentioned in Section III-A, there are various choices for the cost function used in the generic RRT-based planning framework to bias the RRT growth. Such choices include Euclidean distance and Manhattan distance, which work well for sparse and loosely constrained configuration spaces. However, for moderately to highly constrained configuration spaces, such as the "Rooms" and "Maze" maps shown in this paper, such simple cost functions are not able to reflect the structure of the free space, as shown in Figure 3(a).

SARRT uses a different cost function which mimics a diffusion process over the free space in the planning problem. Specifically, we set initial cost values at the initial and goal configurations to be d_{max} and $-d_{max}$, where d_{max} is positive. Then, we solve the diffusion equation (Equation 3) for the free space using an iterative algorithm.

$$\frac{\partial \phi(\mathbf{r}, t)}{\partial t} = D\nabla^2 \phi(\mathbf{r}, t) \tag{3}$$

where D is the diffusion coefficient, and $\phi(\mathbf{r}, t)$ represents the cost function defined on every positions $\mathbf{r}$ at time step t. Practically, we downsample the original map to decrease the computational cost. Since the result of the diffusion process is only used as a rough guide for the tree growth, we found that even a relatively large downsample ratio can obtain results good enough to bias the following RRT growth.

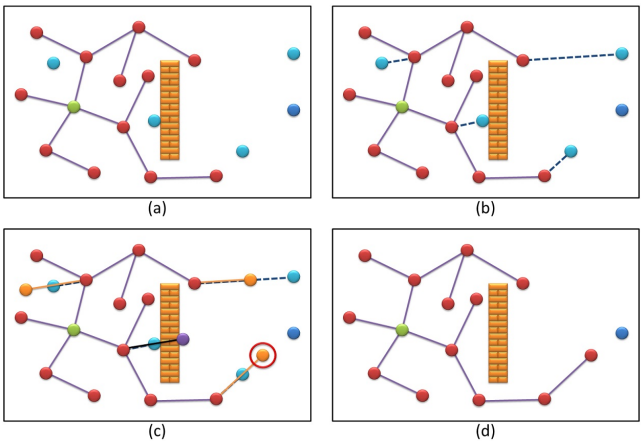

Fig. 2. Extension process of the generic framework. (a) At one iteration, a set of random points (in light blue) are generated. (b) The corresponding nearest nodes in the current RRT are found. (c) New points (in orange) which are considered for potential expansion are generated, with the invalid new point shown in purple and the one with lowest cost value circled. (d) The RRT after extension.

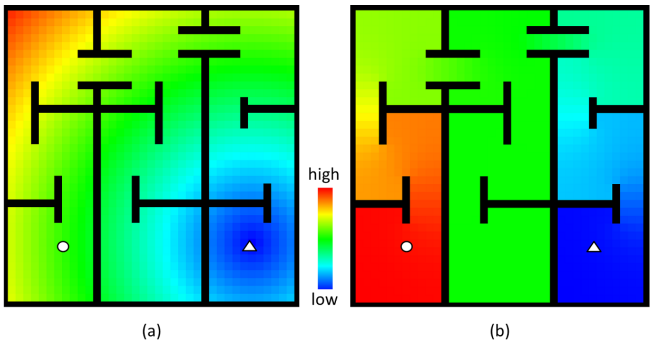

Fig. 3. (a) Cost function based on Euclidean distance vs. (b) structure-aware cost function on the "Rooms" map. The initial and goal configurations are depicted using a circle and a triangle respectively in each sub-figure.

As shown in Figure 1(b) and Figure 3(b), the resulting costmaps are aware of the structure of the free space in the planning problem and the cost function value at every feasible configuration roughly reflects its distance to the goal. To further filter out tree extension operations which cause regression according to the above-mentioned costmap, we introduce an additional regression indicator besides the one presented in Equation 2.

$$\phi(x_{new}) > \phi(x_{nearest}) \tag{4}$$

In all experiments, we downsample the 200×200 maps into a resolution of 40×40. When solving Equation 3 in our experiments, 4000 iterations are used to obtain satisfactory results. To make use of the quadcore CPU, we use OpenMP to parallelize the costmap computation, and the resulting average runtimes is 57ms.

C. Local path optimization

Typical sampling-based planners are aimed at finding a feasible solution, instead of guaranteeing the quality of the solution. As shown in Figure 4(a), most of the paths obtained

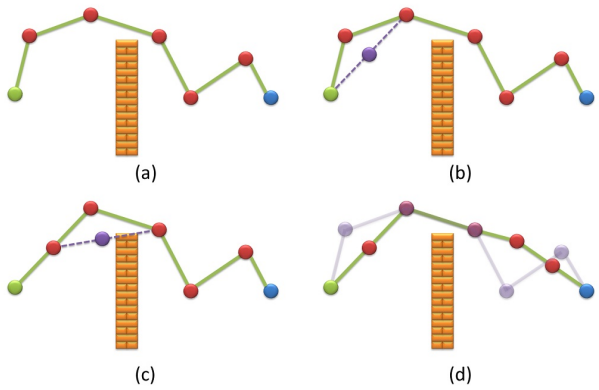

Fig. 4. Local path optimization using Laplacian smoothing. (a) The path returned by the RRT-based planner. (b) Per-vertex smoothing operation moves the circled red vertex to a new position depicted by the purple dot. (c) If the per-vertex smoothing operation would violate some planning constraints, such as obstacles, the operation will be aborted. (d) The path after optimization (shown in green) vs. the original path (shown in light purple).

by RRT-based planners are of a zigzag shape. Therefore, local path optimization is sometimes performed to improve the quality of the resulting path, in terms of path length, obstacle clearance, etc. [15]. Similarly, we propose to use Laplacian smoothing [16] as a postprocessing step to increase the smoothness of the global paths.

The proposed path optimization approach is shown in Figure 4. Given an original global path represented by a series of vertices, $\mathcal{P} = \{\mathbf{x}_i\}(1 \leq i \leq N)$, a new position is chosen for each vertex based on local information and the vertex is moved there, whenever the new configuration satisfies all planning constraints. Specifically, the per-vertex smoothing operation can be described by Equation 5.

$$\mathbf{x}_i = \frac{1}{2}(\mathbf{x}_{i-1} + \mathbf{x}_{i+1}), 2 \leq i \leq N-1. \tag{5}$$

In other application fields such as geometry processing, Laplacian smoothing is typically performed in a iterative manner to obtain a satisfactory result. However, in all our experiments, only one pass of Laplacian smoothing is performed from the initial position to the goal position and the smoothness of the global path is already improved significantly, as shown in Figure 5. It should also be noted that in all the resulting paths presented in Section IV are generated without Laplacian smoothing, since we consider the local path optimization to be a stand-alone postprocessing step.

IV. EXPERIMENTAL RESULTS

Experiments of 2D path planning problems with different maps are conducted to demonstrate the efficiency and effectiveness of SARRT. This section shows experimental results on two different maps, namely the "Maze" and the "Rooms", both given in the format of binary bitmap images with an identical pixel resolution of 200×200. During all experiments, we use a fixed tree growth step of 6 occupancy map

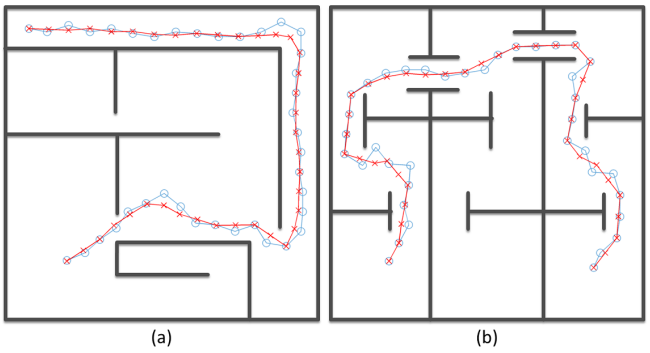

Fig. 5. Comparison of global paths before (shown in blue) and after (shown in red) local path optimization for two different maps.

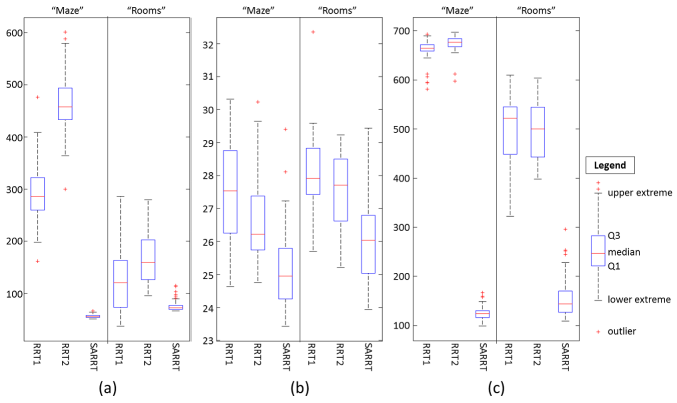

Fig. 6. Performance statistics of algorithm runtimes in milliseconds (a), path lengths in meters (b), and RRT node counts (c). Samples per boxplot: 50.

cell lengths. Because of the stochastic essence of sampling-based approaches, fifty runs of path planning were performed using original RRT with regression avoidance (RRT1), RRT with Euclidean distance as the heuristic cost (RRT2), and SARRT respectively to collect the statistics for performance comparison. Currently, we only consider a holonomic point agent in all experiments for the sake of simplicity.

We have implemented SARRT in ROS [17] (version Indigo Igloo) and all the above-mentioned experiments are conducted using the Stage simulator [18]. The hardware platform is a mainstream desktop with a quadcore Intel Core i7 CPU and 2GB main memory. Note that except for the computation of the structure-aware cost function, only one CPU core is used throughout the experiments shown in the paper.

Figure 6 visualizes the performance statistics for the three different planners on the two maps. It clearly shows that SARRT outperforms the other two planners in algorithm runtime, resulting path optimality, and RRT node counts. The significant gain in time efficiency is due to ... Besides, the additional cost-based regression avoidance mechanism further eliminates unnecessary tree expansions. As a consequence, SARRT produces significantly less tree nodes than the other two planners. Besides the time efficiency gain, less tree nodes also generate a smaller memory footprint.

Figure 7 and 8 provide visualization of experimental results incluing examples of random trees generated and all resulting paths obtained by different RRT-based planners on the "Maze" and "Rooms" maps, respectively. Compared with RRT1 and RRT2, SARRT planner seldom expands into regions which lead to "dead ends" according to the problem configuration, due to the relatively high cost therein. As highlighted by the circled regions in Figure 7 and 8, paths generated by SARRT tend to directly follow the gradient of the structure-aware costmap, and thus statistically have higher quality than paths obtained with RRT1 and RRT2.

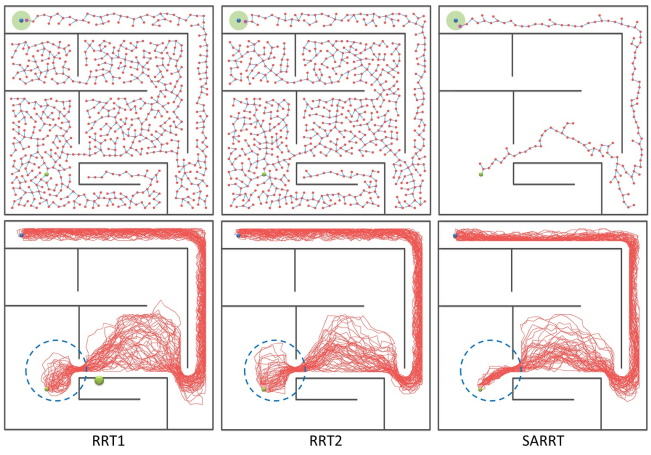

Fig. 7. Comparison of results returned by different RRT-based planners on the "Maze" map, with dashed blue circles highlighting the difference. Upper: examples of random trees generated; Lower: resulting global paths.

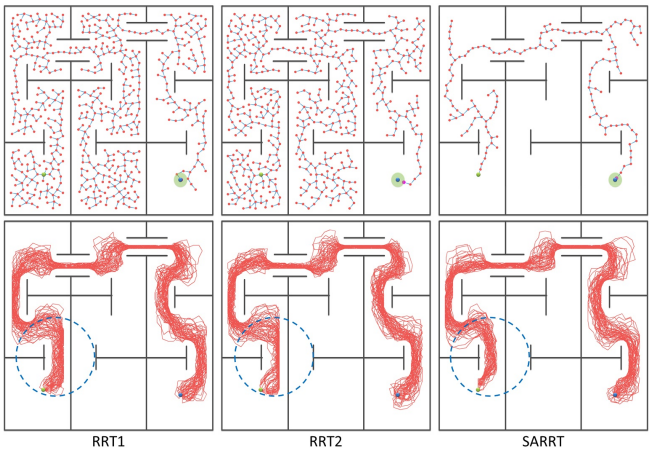

Fig. 8. Comparison of results returned by different RRT-based planners on the "Rooms" map, with dashed blue circles highlighting the difference. Upper: examples of random trees generated; Lower: resulting global paths.

V. CONCLUSIONS

In this paper, we propose a novel RRT variant, namely SARRT, to bias the random tree growth based on a structure-aware costmap defined over the configuration space of a

planning problem. Through comparative experiments of typical path planning problems on 2D maps, we demonstrate that SARRT works well in practice, avoids map regions leading to "dead ends", and shows better performance in algorithm runtime and resulting path quality than the basic RRT planner and RRT planner with Euclidean distance as the heuristic cost. Moreover, various techniques used by other RRT variants, such as more advanced sampling strategies and parallelization, can also be easily integrated to work together with SARRT, since the framework we proposed is quite generic and the adoption of the structure-aware costmap is orthogonal to other improvement techniques.

One limitation of SARRT is the computationally expensive step of obtaining the structure-aware costmap by solving the diffusion equation. It makes the majority of algorithm runtime in our experiments, and will probably render the entire SARRT planner less efficient for simpler problems. However, we believe that the performance of SARRT can be largely improved by further exploiting the parallel computing power of multi-core CPUs or GPUs or using more advanced PDE solvers. Currently, we have only tested SARRT for 2D path planning problems of a holonomic point agent. Future effort will be devoted to applying SARRT in more complex problems, such as path planning for kinodynamic systems and motion planning in higher dimension. This paper demonstrates the performance gain of considering structural information in planning problems, and provides a generic sampling-based framework with a first attempt of using result of a simulated diffusion process as the structure-aware costmap. In the future, we will investigate other indicators or cost functions [19] which are more efficient and able to reflect the structural information better.

REFERENCES

[1] Steven M. Lavalle and James J. Kuffner Jr. Rapidly-exploring random trees: Progress and prospects. In *Algorithmic and Computational Robotics: New Directions*, pages 293–308, 2000.

[2] James J. Kuffner Jr. and Steven M. Lavalle. RRT-Connect: An efficient approach to single-query path planning. In *Proc. IEEE Intl Conf. on Robotics and Automation*, pages 995–1001, 2000.

[3] M. Kalisiak and M. van de Panne. RRT-blossom: RRT with a local flood-fill behavior. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 1237–1242, May 2006.

[4] A. Yershova, L. Jaillet, T. Siméon, and S.M. LaValle. Dynamic-domain RRTs: Efficient exploration by controlling the sampling domain. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 3856–3861, April 2005.

[5] Samuel Rodriguez, Shawna Thomas, Roger Pearce, and NancyM. Amato. Resampl: A region-sensitive adaptive motion planner. In Srinivas Akella, NancyM. Amato, WesleyH. Huang, and Bud Mishra, editors, *Algorithmic Foundation of Robotics VII*, volume 47 of *Springer Tracts in Advanced Robotics*, pages 285–300. Springer Berlin Heidelberg, 2008.

[6] Chonhyon Park, Jia Pan, and D. Manocha. Poisson-RRT. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 4667–4673, May 2014.

[7] Chris Urmson and Reid Simmons. Approaches for heuristically biasing RRT growth. In *In Proceedings of IEEE International Conference on Interlligent Robots and Systems (IROS*, pages 1178–1183, 2003.

[8] L. Jaillet, J. Cortés, and T. Siméon. Sampling-based path planning on configuration-space costmaps. *Robotics, IEEE Transactions on*, 26(4):635–646, Aug 2010.

[9] J. Denny, E. Greco, S. Thomas, and N.M. Amato. MARRT: Medial axis biased rapidly-exploring random trees. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 90–97, May 2014.

[10] S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *Int. J. Rob. Res.*, 30(7):846–894, June 2011.

[11] A. Perez, S. Karaman, Alexander Shkolnik, E. Frazzoli, S. Teller, and M.R. Walter. Asymptotically-optimal path planning for manipulation using incremental sampling-based algorithms. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 4307–4313, Sept 2011.

[12] Alexander Shkolnik, M. Walter, and R. Tedrake. Reachability-guided sampling for planning under differential constraints. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 2859–2865, May 2009.

[13] D.J. Webb and J. van den Berg. Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 5054–5061, May 2013.

[14] Rahul Kala. Rapidly exploring random graphs: motion planning of multiple mobile robots. *Advanced Robotics*, 27(14):1113–1122, 2013.

[15] Roland Geraerts and Mark H. Overmars. Creating high-quality paths for motion planning. *Int. J. Rob. Res.*, 26(8):845–863, August 2007.

[16] G. Taubin. A signal processing approach to fair surface design. In *In Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, pages 351–358, 1995.

[17] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Ng. ROS: an open-source robot operating system, 2009. Accessed: 2015-07-30.

[18] Brian P. Gerkey, Richard T. Vaughan, and Andrew Howard. The player/stage project: Tools for multi-robot and distributed sensor systems. In *In Proceedings of the 11th International Conference on Advanced Robotics*, pages 317–323, 2003.

[19] Kai Xu, Hao Zhang, Daniel Cohen-Or, and Yueshan Xiong. Dynamic harmonic fields for surface processing. *Computers and Graphics*, 33(3):391 – 398, 2009. IEEE International Conference on Shape Modelling and Applications 2009.