Dancing PRM*: Simultaneous Planning of Sampling and Optimization with Configuration Free Space Approximation

Donghyuk Kim[1], Youngsun Kwon[1] and Sung-Eui Yoon[2]

***Abstract*— A recent trend in optimal motion planning has broadened the research area toward the hybridization of sampling, optimization and grid-based approaches. We can expect that synergy from such integrations leads to overall performance improvement, but seamless integration and generalization is still an open problem. In this paper, we suggest a hybrid motion planning algorithm utilizing a sampling-based and optimization-based planner while simultaneously approximating a configuration free space. Unlike conventional optimization-based approaches, the proposed algorithm does not depend on a priori information or resolution-complete factors, e.g., a distance field. Ours instead learns spatial information on the fly by exploiting empirical information during the execution, and decentralizes the information over the constructed graph for efficient access. With the help of the learned information, our optimization-based local planner exploits the local area to identify the connectivity of configuration free space without depending on the precomputed domain knowledge. To show the novelty of proposed algorithm, we evaluate it against other asymptotic optimal planners in both synthetic and complex benchmarks with varying degrees of freedom. We also discuss the performance improvement, properties and limitations we have observed.**

I. INTRODUCTION

Sampling-based motion planning algorithms have been well studied for the past several decades thanks to their probabilistic completeness and wide applicability. Some prominent examples are RRT [1] and PRM [2], which can be viewed as a random geometric graph construction in the configuration free space. For the asymptotic optimal motion planning, Karaman et al. presented RRT*, PRM*, and RRG [3], which guarantee almost-sure asymptotic optimality. These optimal variants successfully opened a new research area in motion planning by providing a theoretical foundation and have been applied to practical solutions of real problems [4]–[6].

In contrast to the sampling-based planners, optimization-based planners convert a non-convex motion planning problem into a sequence of convex problems to quickly find a local optimal solution [7]–[9]. These approaches mainly aim to minimize an objective function with respect to planning constraints, such as smoothness for optimality, while estimating the gradients of obstacle potential for feasibility, i.e., a no-collision constraint.

Some papers have studied the configuration free space approximation method for various purposes [10]–[16]. Among them particularly, Shkolnik et al. [10] represented the configuration free space as a set of hyperspheres using empirical collision information for efficient biased sampling. Bialkowski et al. [14] took a similar approach but used explicit proximity computation to compute the boundary of the hyperspheres. In their following work [15], they used a KD-tree based representation to represent the approximate configuration free space with provable convergence to the ground-truth. Jia Pan et al. [16] suggested a probabilistic collision checking with free space approximation. They introduced an environment learning phase to understand the given geometric structure, then exploited learned knowledge using spatial coherency for a probabilistic collision checking.

As a hybrid approach, RABIT* (Regionally Accelerated Batched Informed Trees) [17] suggested the integration of a sampling-based algorithm BIT* (Batched Informed Tree) [18] and an optimization-based algorithm CHOMP (Covariant Hamiltonian Optimization for Motion Planning) [7]. This hybrid algorithm considers the pros and cons of sampling-based and optimization-based approaches together to identify difficult-to-sample homotopy of the solution path efficiently while preserving the asymptotic optimality. It is, however, only workable under the assumption that a precomputed domain knowledge, e.g., a distance field, is given a priori or is analytically computable on demand, which can be a burden in practical problems.

In this paper, we present a hybrid approach of sampling-based and optimization-based planning, in which the entire planning process is accomplished on the fly without depending on any precomputed domain knowledge. The proposed algorithm uses both empirical collision and collision-free information found during the sampling-based planning to learn the configuration free space during the execution. The optimization-based planner utilizes the learned information to guide trajectories toward a configuration free space to establish more connections between sampled configurations. We also suggest an efficient access structure so that two different planners can work seamlessly with our approximate configuration free space.

In the subsequent sections, we first explain preliminaries of the sampling-based and optimization-based approach (sec. II). Sec. III gives an overview of the proposed algorithm (Sec. III-A), how to approximate configuration free space with sampling-based motion planning (Sec. III-B) followed by optimization-based local planning with learned information (Sec. III-C). We also analyze the properties of our method (Sec. IV) and demonstrate its benefits by experiments with various dimensions (Sec. V).

[1]Donghyuk Kim, Youngsun Kwon are with the School of Computing ({donghyuk.kim, youngsun.kwon}@kaist.ac.kr) and [2]Sung-Eui Yoon is with the Faculty of School of Computing (Corresponding author, sungeui@kaist.edu) , Korea Advanced Institute of Science and Technology (KAIST) at Daejeon, South Korea 34141.

Algorithm 1: NAÏVE PRM*

1 $V \leftarrow \{q_{init}, q_{goal}\}$
2 $E \leftarrow \emptyset$
3 **while** *Termination condition is not satisfied* **do**
4 $\quad q_{rand} \leftarrow Sample()$
5 $\quad$ **if** $IsCollisionFree(q_{rand})$ **then**
6 $\quad\quad$ Insert q_{rand} to V
7 $\quad\quad Q_{near} \leftarrow Near(q_{rand})$
8 $\quad\quad$ **foreach** $q_{near} \in Q_{near}$ **do**
9 $\quad\quad\quad$ **if** $IsCollisionFree((q_{near}, q_{rand}))$ **then**
10 $\quad\quad\quad\quad$ Insert (q_{near}, q_{rand}) to E
11 $\quad UpdateSolutionPath(G)$
12 **return** $SolutionPath(G)$

II. BACKGROUND

This section reviews major previous studies and presents preliminaries and notations employed throughout the paper.

A. Sampling-based Motion Planning

In this paper, we mainly consider the optimal motion planning problem, whose objective is to find a feasible and optimal trajectory ξ connecting two given end points q_{init} and q_{goal} satisfying $\xi(0) = q_{init}$ and $\xi(1) = q_{goal}$ in the configuration space $\mathbb{X}$, where a trajectory $\xi : [0,1] \rightarrow \mathbb{X}$. For feasibility, ξ should lie in the configuration free space $\mathbb{X}_{free}$, where $\mathbb{X}_{free} \subset \mathbb{X}$ and the configuration obstacle space $\mathbb{X}_{obs}$ is defined to be $\mathbb{X} \setminus \mathbb{X}_{free}$.

To explain how sampling-based approaches construct a search graph $G = (V, E) \in \mathbb{X}$, we briefly explain naïve PRM* shown in Alg. 1, which is one of the prominent sampling-based planners.

In each iteration of Alg. 1, PRM* samples a random configuration q_{rand} and checks its validity with $IsCollisionFree(\cdot)$ (Line: 5) which returns *true* if a given configuration q or an edge (q_i, q_j) is valid i.e. $\in \mathbb{X}_{free}$. For each valid configuration q_{rand}, r-nn(r-nearest neighbor) query of $Near(\cdot)$ finds near neighbors (Line: 7) within a ball of radius $\gamma\left(\frac{\log|V|}{|V|}\right)^{1/d}$ centered at q_{rand}, where d is the dimension of the problem, $|V|$ is the cardinality of V and γ is a user defined constant greater than 1 [3]. k-nn(k-nearest neighbor) query can also be an alternative of $Near(\cdot)$; in that case, k is defined by $\lceil \gamma(\mathrm{e} + \frac{\mathrm{e}}{d}) \cdot log(|V|) \rceil$.

Line 9 checks collision for every possible connection of $(q_{near} \in Q_{near},\ q_{rand})$ to find a feasible connection between configurations in G; this is also known as *local planning*. $UpdateSolutionPath(\cdot)$ computes the shortest path from q_{init} to q_{goal} on the constructed graph G if anytime property is required for the given problem. Otherwise, the shortest path is computed (Line: 12) using well-known A* or Dijkstra's algorithm at the end of execution.

In this study, the foundation of the proposed algorithm is based on PRM*. Its sampling-based behavior divides the entire motion planning problem into a set of smaller local planning problems while approximating the configuration free space. We discuss how the optimization-based local planning works with our spatial information in the subsequent section.

TABLE I
NOTATION SUMMARY TABLE

Notation	Description
$\xi(t)$	Configuration on the trajectory ξ at a time $t = [0,1]$.
B	Set of entire body point for a given robot model.
$x(\xi(t), b)$	Mapping of a body point $b \in B$ at a configuration $\xi(t)$ to the corresponding point in the workspace.
x'	Derivative of $x(\cdot)$.
$c(\cdot)$	Obstacle potential for being close or inside $\mathbb{X}_{obs}$.
J	Kinematic Jacobian, i.e., $\frac{d}{dq}x(q,b), q \in \mathbb{X}$ and $b \in B$
V, E	Vertex and edge set of the search graph G.
V^*	Set of free configuration observed during planning.

B. Optimization-based Motion Planning

Our local planner is based on a well-known gradient optimization technique, CHOMP (Covariant Hamiltonian Optimization for Motion Planning) [7]. We first briefly review the concept of CHOMP and discuss the motivation with our observations.

The objective of CHOMP is to find a smooth, collision-free trajectory ξ, exactly like that of sampling-based planning. The objective function, $\mathscr{U}(\xi)$, is then formalized as the following:

$$\mathscr{U}(\xi) = f_{prior}(\xi) + \lambda \cdot f_{obs}(\xi), \tag{1}$$

where f_{prior} can be considered as a sum of squared derivatives for the trajectory ξ to satisfy local optimality and additional constraints, such as controlling smoothness or limiting the maximum acceleration. The obstacle cost function f_{obs} penalizes a configuration of a robot for being close to $\mathbb{X}_{obs}$ to avoid any collision. To be specific, f_{obs} and its gradient ∇f_{obs} are formalized in Eqs. 2 and 3, respectively. The notations used in both equations and throughout the paper are also summarized in Tab. I.

$$f_{obs}(\xi) = \int_0^1 \int_B c(x(\xi(t), b)) \parallel \frac{d}{dt} x(\xi(t), b) \parallel db\ dt \tag{2}$$

$$\nabla f_{obs}(\xi) = \int_B J^T \|x'\| \cdot [(I - \hat{x}'\hat{x}'^T)\nabla c - c\kappa]\ db \tag{3}$$

The computation of the obstacle potential $c(\cdot)$ depends on a body simplification, and workspace information such as distance field [7] which can be analytically computed in the workspace. The robot model B is generally simplified by a sphere covering method [9], [19]. $x(\cdot)$ plays a mapping role from a configuration $\xi(t)$ defined in the configuration space to the workspace. As a result, the integration of whole body B results in the obstacle potential for a configuration $\xi(t)$.

Eq. 3 shows the gradient of Eq. 2, where $\nabla c(\cdot)$ is the gradient of obstacle potential $c(\cdot)$, and κ is the curvature of the trajectory. The objective function of CHOMP contains obstacle potential terms such as $c(\cdot)$ and ∇c, which are hard to compute in the configuration space. For this reason, the conversion from the configuration space to the workspace with $x(\cdot)$ and J is introduced to compute f_{obs} and ∇f_{obs} for a trajectory defined in the configuration space using workspace information. It also makes the planning process less affected

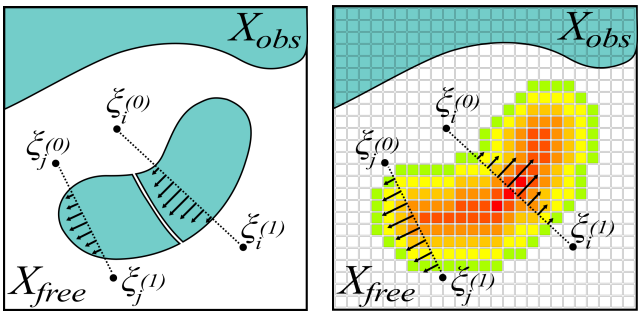


Fig. 1. The left figure shows an example of a trajectory optimization problem with two initial trajectories ξ_i and ξ_j where the black arrow indicates a direction toward the closest free space from each intermediate configuration along the trajectories. The right figure shows the result with a distance field, which is a discretization of a given environment. The penetration depth of each cell is color encoded, red for high and green for low. In this scenario, it fails to capture the narrow passage in the optimal homotopy due to the coarse-grained resolution.

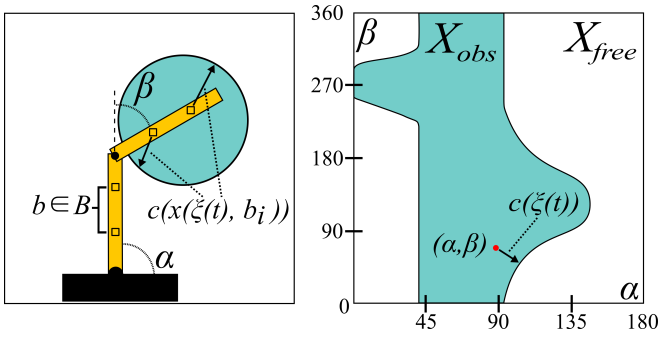


Fig. 2. Both figures show a 2D manipulation problem in a planar space and how the environment of given problem looks different in the workspace (left) and the configuration space (right). In the workspace, a robot arm is simplified by a set of body points $b \in B$ (small squares); its potentials against the obstacle circle are computed by the proximity of those in the workspace (black arrow). The right figure shows the same circumstance, but in the configuration space.

by the dimensions of the configuration space for practical performance.

The aforementioned benefit, however, comes with two drawbacks as follows. First, CHOMP only guarantees the convergence to a local optimum since the optimization process exploits the local convexity of f_{obs}. Fortunately, the local optimality issue has been studied extensively with a stochastic sampling [7], [8] and a hybrid approach of sampling-based and optimization-based planning [17]. Second, CHOMP and its variants assume a discretized representation, e.g., distance fields in finite resolutions, of the workspace information and the simplified robot model B. The discretized representation of the workspace information is a relatively less studied area.

Fig. 1 reveals a limitation on the use of distance field, which is defined in the workspace with a fixed resolution. As we see, the actual environment (left) has a narrow passage, while the distance field (right) fails to capture the narrow passage due to its coarse-grained resolution.

Fig. 2 illustrates the difference between obstacle potentials computed from the workspace (left) and the configuration space (right). In the left figure, $c(\cdot)$ and $\nabla c(\cdot)$ for a configuration needs to be computed by considering the obstacles with the simplified body B and average them according to Eq. 2 and Eq. 3 in order to be used in the configuration space where ξ is defined. For this reason, the resolution of the distance field and the simplified robot representation affect the performance and completeness. On the other hand, the right figure shows what they look like in the configuration space, where obstacle potentials can be computed without the robot body simplification B, $x(\cdot)$ and Jacobian, J. It is, however, a hard problem to compute such information, e.g., a fine-resolution distance field in a high dimensional configuration space.

To deal with this context, we set our goal to perform the entire optimization process solely in the configuration space without using discretized workspace information or expensive overheads related to proximity computation in the configuration space, while preserving the asymptotic optimality by integrating a sampling-based approach.

III. Algorithm

In this section, we elaborate each component of the proposed algorithm and the underlying theoretical meanings.

A. Overview

At a high level, our approach is based on the integration of optimization-based and sampling-based planning, while simultaneously constructing an approximate configuration free space without the priori knowledge of the given environment. The adjective "Dancing' in the title metaphorically describes how the proposed algorithm works, connecting configurations in G with optimized trajectories which are curved in the configuration space. Specifically, sampling-based planning is used for decomposing the entire problem into a set of local planning problems, while constructing an approximate configuration free space, $\widetilde{\mathbb{X}}_{free}$, on the fly. We then solve each sub-problem using optimization with $\widetilde{\mathbb{X}}_{free}$ to exploit the local area around $\mathbb{X}_{obs}$ efficiently.

The pseudocode of the proposed algorithm called *Dancing PRM** is depicted in Alg. 2. The main flow of the algorithm is based on PRM*, and we also apply a lazy collision checking [20] to minimize the number of local plannings by skipping unnecessary ones.

Procedures newly added or that behave differently from a conventional planner are highlighted in Alg. 2. Procedures in (Line: 5, 8, 11, 13, 15, 16) are for $\widetilde{\mathbb{X}}_{free}$ construction, and $DanceSolutionPath(\cdot)$ (Line: 17) validates a best-so-far solution path in a lazy manner with an additional optimization-based local planning step. $UpdateShortestPathTree(\cdot)$ (Line: 12) is invoked when a graph restructuring, such as edge insertion or deletion occurs to maintain the best solution path from q_{init} to q_{goal} over the search graph G, which is necessary for anytime lazy collision checking.

We discuss details of each component in the subsequent subsections.

B. Configuration Free Space Approximation

The main purpose of the approximate configuration free space $\widetilde{\mathbb{X}}_{free}$ is to efficiently guide trajectories towards local configuration free space during the optimization.

Algorithm 2: DANCING PRM*

1 $V \leftarrow \{q_{init}, q_{goal}\}$
2 $E \leftarrow \emptyset$
3 **while** *Termination condition is not satisfied* **do**
4 $\quad q_{rand} \leftarrow Sample()$
5 $\quad$ **if** $IsCollisionFree(q_{rand})$ **then**
6 $\quad\quad$ Insert q_{rand} to V; $V^*_{q_{rand}} \leftarrow \emptyset$
7 $\quad\quad Q_{near} \leftarrow Near(q_{rand})$
8 $\quad\quad$ Insert Q_{near} to $V^*_{q_{rand}}$
9 $\quad\quad$ **foreach** $q_{near} \in Q_{near}$ **do**
10 $\quad\quad\quad$ Insert (q_{near}, q_{rand}) to E
11 $\quad\quad\quad$ Insert q_{rand} to $V^*_{q_{near}}$
12 $\quad\quad UpdateShortestPathTree(\cdot)$
13 $\quad\quad PropagateCFreeSpace(Q_{near}, q_{rand})$
14 $\quad$ **else**
15 $\quad\quad q_{nearest} \leftarrow Nearest(q_{rand})$
16 $\quad\quad UpdateCFreeSpace(q_{rand}, q_{nearest})$
17 $\quad DanceSolutionPath(G)$
18 **return** $SolutionPath(G)$

We choose to represent $\widetilde{\mathbb{X}}_{free}$ as a set of *hyperspheres* motivated by previous studies [10], [14] for scalability and light-weight proximity computation, e.g., the distance function. While the representation is analogously defined, our study differs in terms of the approximation procedure and accessing strategy for the efficient integration with optimization-based planning explained in a later paragraph.

Fig. 3(a) shows a conceptual image of $\widetilde{\mathbb{X}}_{free}$, which can be simply formalized as:

$$\widetilde{\mathbb{X}}_{free} = \{x \mid \|x - q_i\| < r_{q_i}\},\ q_i \in V^* \tag{4}$$

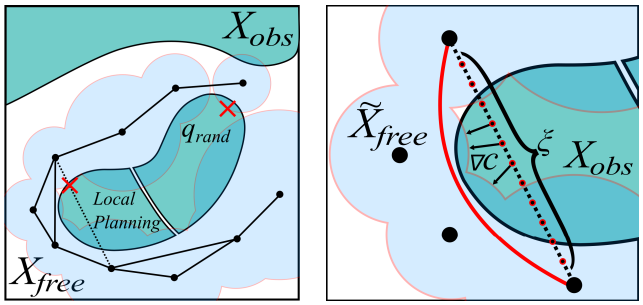
where r_{q_i} is the radius of an approximate collision free hypersphere centered at q_i, and V^* is a set of all $q \in \mathbb{X}_{free}$ found during the execution; thus $V \subset V^*$. Intuitively, each configuration $q \in V^*$ is associated with a single hypersphere approximated by an empirical collision, the so-called *witness*, $w_q \in \mathbb{X}_{obs}$, such that $\|q - w_q\| = r_q$.

A new witness w_q can be generated and replaced by the following samples, as long as it results in a smaller radius:

1) a sampled configuration $q_{rand} \in \mathbb{X}_{obs}$ (Line: 15-16 in Alg 2). This update is done by $UpdateCFreeSpace(\cdot)$.
2) an intermediate configuration that turned out to be in X_{obs} during a solution path validation (Line: 5, 9 in Alg 3).
3) a witness of near neighbors (Line: 13 in Alg. 2). This is handled by our witness propagation step.

Note that the intermediate configuration q_{inter} can be computed in $IsCollisionFree(\cdot)$ using a discrete collision checker, which is widely used in the most conventional sampling-based planners.

Decentralized storage for observed collision states. For efficient access to our approximate free space, $\widetilde{\mathbb{X}}_{free}$ during a local optimization process, we adopt a decentralized storage strategy for V^*, where each $q \in V$ maintains a subset $V^*_q \in V^*$.



(a) (b)

Fig. 3. The left is a visualization of $\widetilde{\mathbb{X}}_{free}$, regions covered by a set of merged light blue circles in 2D. Each configuration $q \in V^*$ is associated with an approximate collision-free hypersphere in $\mathbb{X}$. Their radii are trimmed by *witness* (red cross symbol) which is a configuration in $\mathbb{X}_{obs}$ found during a local planning (dotted black segment on the left side) or a sample q_{rand} (right side). The right shows an example of local optimization for a trajectory ξ. Black arrows show the gradient of obstacle potential computed with our approximate configuration space and the red curved segment shows an optimized trajectory. Each red dot indicates an intermediate configuration $\xi(t)$ on the discretized ξ.

V^*_q is updated with its near neighbors (Line: 8, 11 in Alg. 2) and collision-free configurations q_{inter} found during a local planning (Line: 5, 9 in Alg. 3); the maximum distance from q to those newly added $v \in V^*_q$ is bounded by $\gamma\left(\frac{\log|V|}{|V|}\right)^{1/d}$ when they are inserted into V^*_q, according to the ball radius adopted by $Near(\cdot)$. Note that q_{inter} is not in V but only in V^*, and we initialize their collision-free radii with zero i.e., $r_{q_{inter}} = 0$.

Witness propagation step. To acquire more accurate $\widetilde{\mathbb{X}}_{free}$ in practice, we apply a witness propagation step (Line: 11 in Alg. 2). Our key insight for the witness propagation is to treat finding the closest configuration obstacle for a $q \in \mathbb{X}_{free}$ as a connectivity problem on random geometric graphs [3]. This suggests that for an arbitrary configuration $q \in V$, all witnesses of V located in the ball of radius $\gamma\left(\frac{\log|V|}{|V|}\right)^{1/d}$ centered at q serve as candidates for w_q.

For this purpose, we define $PropagateCFreeSpace(\cdot)$ (Alg. 4), which initializes $r_{q_{new}}$ using witnesses of near neighbors, Q_{near}, and also propagates its witness $w_{q_{new}}$ to Q_{near} at the same time. This process of witness propagation is fundamentally analogous to the *rewire* procedure of RRT* [3], in terms of a threshold bound for the connectivity.

C. Optimization in Configuration Space

Our CHOMP-based optimizer is performed lazily in $DanceSolutionPath(\cdot)$, as depicted in Alg. 3, when we try to validate the best-so-far path. At the beginning of each iteration, we retrieve a provisional solution path $E_{solution} \subset E$ (Line: 3), which possibly contains infeasible edges. When the solution path validation fails by an edge collision checking for an edge e_i (Line: 5), we reject e_i from E and then update $\widetilde{\mathbb{X}}_{free}$ with a configuration $q_{obs} \in \mathbb{X}_{obs}$ found during edge collision checking (Line: 7).

Our optimization-based local planner then attempts to optimize the invalid edge e_i (Line: 8) to find a new trajectory bypassing local $\mathbb{X}_{obs}$ using $\widetilde{\mathbb{X}}_{free}$. Successful optimization yields a non-linear trajectory e_{opt}, which increases the chance

Algorithm 3: *DanceSolutionPath*

Input: G, a search graph

$E_{solution} \leftarrow \emptyset$
repeat
  $E_{solution} \leftarrow ProvisionalSolutionPath(G)$
  foreach $e_i \in E_{solution}$ **do**
    if $\neg IsCollisionFree(e_i)$ **then**
      $E = E \setminus \{e_i\}$
      $UpdateCFreeSpace(e_i)$
      $e_{opt} \leftarrow Optimize(e_i)$
      if $IsFeasible(e_{opt})$ **then**
        $E = E \cup \{e_{opt}\}$
      $UpdateShortestPathTree(G)$
      Break
until $e_i \in \mathbb{X}_{free}, \forall e_i \in E_{solution}$

Algorithm 4: *PropagateCFreeSpace*

Input: q_{new}, a sample configuration,
Q_{near}, a set of near neighbor of q_{new}

$r_{q_{new}} \leftarrow \infty$; $w_{q_{new}} \leftarrow \emptyset$
foreach $q_{near} \in Q_{near}$ **do**
  if $(w_{q_{near}} \neq \varnothing) \wedge (\|w_{q_{near}} - q_{new}\| < r_{q_{new}})$ **then**
    $r_{q_{new}} \leftarrow \|w_{q_{near}} - q_{new}\|$
    $w_{q_{new}} \leftarrow w_{q_{near}}$
foreach $q_{near} \in Q_{near}$ **do**
  if $(w_{q_{new}} \neq \varnothing) \wedge (\|w_{q_{new}} - q_{near}\| < r_{q_{near}})$ **then**
    $r_{q_{near}} \leftarrow \|w_{q_{new}} - q_{near}\|$
    $w_{q_{near}} \leftarrow w_{q_{new}}$

of reducing the cost of the solution path or finding a better solution homotopy.

The local planner explicitly takes into account the obstacle potential computation with our approximate configuration free space $\widetilde{\mathbb{X}}_{free}$ which learns the given arbitrary environment dynamically. This generality separates our work from RABIT* [17], which assumes the obstacle potential, f_{obs} and ∇f_{obs}, to be given as a priori or analytically computable. We also explain features of ours that are different from CHOMP.

In the objective function of Eq. 1, f_{prior} is generally assumed to be independent of the environment [7] and computed in the configuration space. We, therefore, only deal with f_{obs} depicted in the following equation:

$$f_{obs}(\xi) = \int_0^1 c(\xi(t)) \parallel \frac{d}{dt}\xi(t) \parallel dt \tag{5}$$

Unlike the original form in Eq. 2, we can naturally eliminate the following two things:

1) a workspace-configuration mapping function $x(\cdot)$.
2) an integration over the simplified body model B.

These are possible because we can directly compute the obstacle potential $c(\xi(t))$ with the following equation [7]:

$$\begin{cases} -\mathfrak{D}(\xi(t)) + \frac{1}{2}\varepsilon, & \mathfrak{D}(\xi(t)) < 0 \\ \frac{1}{2\varepsilon}(\mathfrak{D}(\xi(t)) - \varepsilon)^2, & 0 < \mathfrak{D}(\xi(t)) \leq \varepsilon \\ 0 & otherwise \end{cases}$$

where $\mathfrak{D}(\xi(t))$ is a distance field value computed with our approximate configuration free space $\widetilde{\mathbb{X}}_{obs}$ i.e., $\mathfrak{D}(\xi(t)) = -min(\|\xi(t) - q\| - \omega(|V_q^*|) \cdot r_q), \forall q$ which gives a negated distance to the closest $\widetilde{\mathbb{X}}_{free}$.

The concept of $\omega(\cdot)$ is to compensate the overestimation of r_q. Since our approximation result entirely depends on the random sampling procedure, we can expect an overestimation of collision-free radius r, i.e., $r \geq r^*$, r^* for a ground-truth distance to the closest obstacle space. To accommodate this approximation error, we define $\omega(n)$, a parameter function of n to be $1.0 - \zeta\left(\frac{\log(n)}{(n)}\right)^{1/d}$, which converges to 1 as $n \to \infty$ for a user parameter $\zeta \geq 0$. This is based on the concept of statistical *dispersion*, which is identical to the radius of a largest empty hypersphere, i.e., containing no $v \in V_q^*$. The multiplication of $\omega(\cdot)$ and the collision-free radius r_q therefore can be considered a probabilistic maximum error of r_q.

Fig. 3(b) visualizes an example of a gradient of obstacle potentials denoted by black arrows in the configuration space. By applying compensation $\omega(\cdot)$, the boundary of $\widetilde{\mathbb{X}}_{free}$ can shrink toward V^* and the optimizer works more conservatively. We can also compute ∇f_{obs}, as follows:

$$\nabla f_{obs}(\xi) = \|\xi'\| \cdot [(I - \hat{\xi}' \cdot \hat{\xi}'^T)\nabla c - c\kappa] \tag{6}$$

where ∇c for a specific $\xi(t)$ is a normalized d-dimensional direction vector toward the closest $\widetilde{\mathbb{X}}_{free}$. To be specific, it can be computed as $\frac{p-\xi(t)}{\|p-\xi(t)\|}$, where $p = \arg\min_{q \in V^*}(\|\xi(t) - q\| - \omega(|V_q^*|) \cdot r_q)$.

It is, however, computationally expensive if we attempt to consider the entire V^* for computing the obstacle potentials $c(\xi(t))$ and $\nabla c(\xi(t))$ which must be evaluated repeatedly during the optimization. To this end, we restrict the configuration space used for a local optimization of ξ within $V^*_{q_{from}} \cup V^*_{q_{to}}$. This restriction can be a reasonable choice in the light of the fact that the maximum length of a newly generated edge is limited to $\gamma\left(\frac{\log|V|}{|V|}\right)^{1/d}$, when r-nn is used for $Near(\cdot)$. According to the definition of V_q^*, all the edges coming out from a configuration q is also completely inside V_q^* at the initial phase of the optimization. Under these circumstances, the optimizer makes the edges, i.e., the initial trajectories, to converge within the configuration space occupied by $\widetilde{\mathbb{X}}_{free}$ of $V^*_{q_{from}} \cup V^*_{q_{to}}$, which is a practically reasonable space for a local planning problem for the faster optimization.

Back to Alg. 3, $IsFeasible(\cdot)$ checks whether the optimized trajectory e_{opt} is valid using a sequence of $IsCollisionFree(\cdot)$ calls where the update of $\widetilde{\mathbb{X}}_{free}$ is involved. $DanceSolutionPath(\cdot)$ is terminated when it yields a feasible solution path after validating the best-so-far provisional solution path by lazy collision checking.

IV. Analysis

In this section, we discuss the properties of the proposed method. We first discuss the asymptotic optimality and then the time complexity of the computational overhead induced by the configuration free space approximation and optimization-based local planning.

A. Almost-sure Asymptotic Optimality

Let $E_{proposed}$ and $E_{lazyPRM^*}$ refer to the valid edges in a graph constructed by the proposed method and lazy PRM* [20], respectively. A vertex set of $V_{lazyPRM^*}$ and $V_{proposed}$ is defined in a similar way. Without loss of generality, we assume that a sequence of random samples and subroutines in both planners are identical.

As described with Alg. 3, the proposed method never rejects any edge $e \in E_{lazyPRM^*}$ because the path validation in the proposed algorithm is identical to that of lazy PRM* except for the additional trajectory optimization. Furthermore, the proposed algorithm optimizes only edges that turns out to be invalid, $E_{proposed}$ rather has a chance to have more edges than $E_{lazyPRM^*}$. (Line: 10 in Alg. 3). Therefore, $E_{lazyPRM^*} \subseteq E_{proposed}$ holds.

The vertex set is identical, because no modification is applied to the sampling and collision checking on q_{rand} as shown in Algs. 1 and 2, i.e., $V_{lazyPRM^*} = V_{proposed}$. Consequently, if we compute a solution path on $G_{proposed} = \{V_{proposed}, E_{proposed}\}$ the optimality of the proposed algorithm follows that of lazy PRM*, which is proven almost-sure asymptotically optimal [20].

B. Computational Time Complexity

The complexity and analysis of primitive operations used for our method follows the discussion in [21], and we thus deal with overheads introduced mainly by the proposed algorithm.

In Alg. 2, we added various steps for $\widetilde{\mathbb{X}}_{free}$ construction. First of all, updates on V^* (Line: 5, 8, 11, 16) linearly increase as the number of elements to be inserted increases, because V^* does not have to be an ordered structure. We also have $O(|Q_{near}|)$ of iterations in $PropagateCFreeSpace(\cdot)$; thus the time complexity of the entire while loop in Alg. 2 is dominated by that of $Near(\cdot)$, i.e., $O(\gamma^d \cdot 2^d \cdot log(|V|))$, which is identical to the expected cardinality of Q_{near}.

We perform an additional nearest neighbor search for $q_{rand} \in \mathbb{X}_{obs}$ (Line: 15), which is proportional to $log(|V|) \cdot \mathfrak{L}(\mathbb{X}_{obs})$, where $\mathfrak{L}(\cdot)$ is a Lebesgue measure, i.e., the hypervolume of $\mathbb{X}_{obs}$. It is an additional overhead compared to PRM*, depending on the volume of the configuration obstacle space.

$DanceSolutionPath(\cdot)$ (Line: 17) validates the provisional solution path as shown in Alg. 3. On top of the procedures for lazy collision checking in this function, we additionally perform $Optimize(\cdot)$ for optimization-based local planning. Its computational overhead with modified obstacle potentials defined in Eqs. 5 and 6 can be expressed as $i_{max} \cdot z \cdot C(\mathfrak{D}(\cdot))$ where i_{max} is the maximum iteration of optimization, z, the number of discretized intermediate nodes and $C(\mathfrak{D})$, the computational complexity for $\mathfrak{D}(\cdot)$ calculation. To compute $\mathfrak{D}(\cdot)$, subsets of V^* associated with the two end points of ξ should be considered. Its computational cost can be bounded by $O(d \cdot \gamma^d \cdot 2^d \cdot log(|V|))$ which is a multiplication of the expected number of near neighbors and d for the distance computation in d dimensional Euclidean space. The number of $optimize(\cdot)$ calls is, however, remarkably reduced by lazy collision checking in practice [20].

V. Experiments

A. Experiment setup

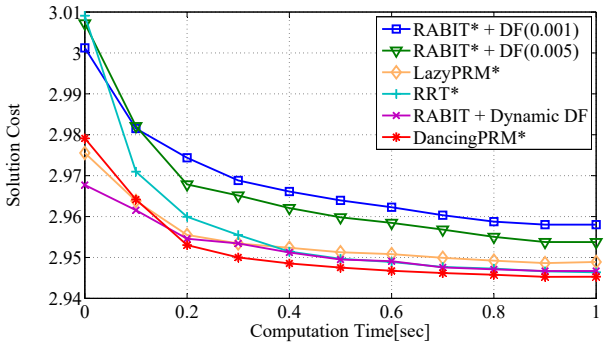
For fair comparison, all the tested methods are built upon the same proximity subroutines such as discrete collision detector and the nearest neighbor search available in OMPL (Open Motion Planning Library) [22] integrated with a CHOMP-based optimizer on V-REP simulator [23]. r-nn is used for $Near(\cdot)$ with a parameter $\gamma = 1.1$ and the CHOMP-based optimizer works with parameters of $\lambda = 1$, $\varepsilon = 10^{-3}$, $\mu = 2.0$, $z = 10$, $\zeta = 1.1$ and $i_{max} = 10$, which are applied identically to RABIT* and our algorithm. The reported results are averaged over 20 trials and we only report values on the plot after all of trials find the initial solution path.

B. Comparison against RABIT*

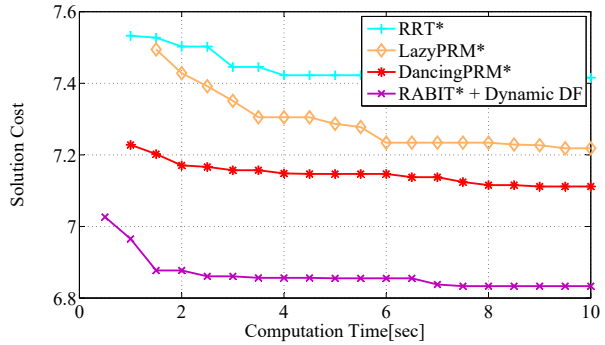
We first compare the performance of DancingPRM* against RABIT*, which works when the explicit representation of $\mathbb{X}_{free}$ is available. Since our approach does not assume such an environment, we can not say that the direct comparison in this setting is completely fair. Nonetheless, we would like to show how our method works, even in these cases, to show the difference against RABIT* through this section.

For this test, our evaluation benchmark is constructed by following the ones used in the original paper of RABIT*. Specifically, they are $\mathbb{R}^2$ and $\mathbb{R}^8$ configuration spaces where a wall with 10 narrow passages are located at the center in a d-dimensional hypercube of a width 2, i.e. $[-1,1]^d$. The boundary of the configuration obstacle space is created to be axis-aligned for easy computation of $c(\cdot)$ and $\nabla c(\cdot)$. A pair of input configurations (q_{init}, q_{goal}) are set to ($[-1,...,-1]$, $[1,...,1]$), and therefore, this benchmark is designed to have multiple difficult-to-sample homotopies of solution paths.

Fig. 4 shows our experimental results of the solution cost as a function of computation time. Since we assume that the domain knowledge is available, "RABIT* + DF(δ)" uses a distance field with a resolution of δ given as a *priori*. "RABIT* + Dynamic DF" uses obstacle potentials computed analytically on the fly. This runtime computation is possible because we consider only axis-aligned configuration obstacles in this test. Fig. 4(a) shows that RABIT* with a dynamic distance field outperforms than those with precomputed distance fields because using the precomputed distance fields works more conservatively, i.e., yielding longer trajectories that are farther from the boundary of $\mathbb{X}_{obs}$ on a given δ. Even in this case, our method shows comparable performance to "RABIT* + Dynamic DF" due to the local optimization that uses configuration free space approximation.



(a) $\mathbb{R}^2$



(b) $\mathbb{R}^8$

Fig. 4. Plot of the solution cost over computation time for four different algorithms. Results are measured against the benchmarks of both $\mathbb{R}^2$ and $\mathbb{R}^8$.

In $\mathbb{R}^8$, it takes a huge amount of time and memory to construct a distance field with reasonable resolution. We therefore only report "RABIT* + Dynamic DF" and other planners. As we have observed, the experimental result can be seen as negative for the proposed algorithm; $\mathbb{R}^8$ especially seems to show our limitation in a high dimensional problem. In practice, however, we do not know the exact configuration free space, and obstacle potential computation in runtime requires heavy computational overhead as well. For these reasons, we further evaluate the proposed algorithm with more general benchmarks, which RABIT* could not handle directly because of the absence of an appropriate obstacle potential computation function due to the complexity of the configuration space.

C. Comparison in complex configuration spaces

In this experiment we compare DancingPRM* against other asymptotic optimal planners, RRT*, lazyPRM* and lazy LBT-RRT [24] which is a near-optimal version of RRT* with lazy local planning. The parameter τ in parentheses of the lazy LBT-RRT makes the planner almost surely converge to within $(1+\tau)$ times of the optimum in terms of the solution cost. Fig. 6 shows a comparison result tested with our benchmark set illustrated in Fig. 7, where the configuration spaces of our benchmarks in Figs. 7(a), 7(b), 7(c) are $\mathbb{R}^2$, $SE(3)$ and $\mathbb{S}^6$, respectively. Fig. 5 shows a computation time breakdown of DancingPRM*.

The benchmarks contain both easy-to-find homotopies and difficult-to-find optimal homotopy of a solution path. The optimization-based local planner cannot only improve the performance by optimizing a solution path in a specific homotopy, but also help to identify a better homotopy earlier than other tested planners. Moreover, the light computation of our $\widetilde{\mathbb{X}}_{free}$ makes the proposed algorithm accomplish the entire process in runtime without any priori information while providing a better result against other tested algorithms.

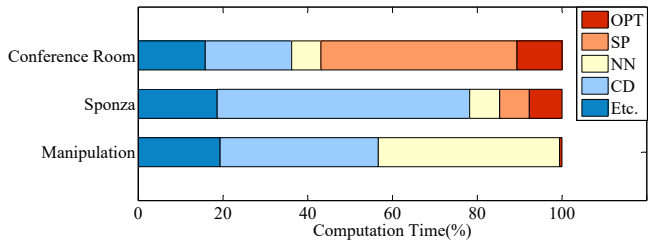


Fig. 5. Computation time breakdown of the proposed algorithm measured in our benchmarks (Fig. 7). Each abbreviation in legend stands for OPTimization (OPT), Shortest Path tree update (SP), Nearest Neighbor search (NN), and Collision Detection (CD). Note that the computation time for $\widetilde{\mathbb{X}}_{free}$ construction is negligible ($< 1\%$) for all of three benchmarks.

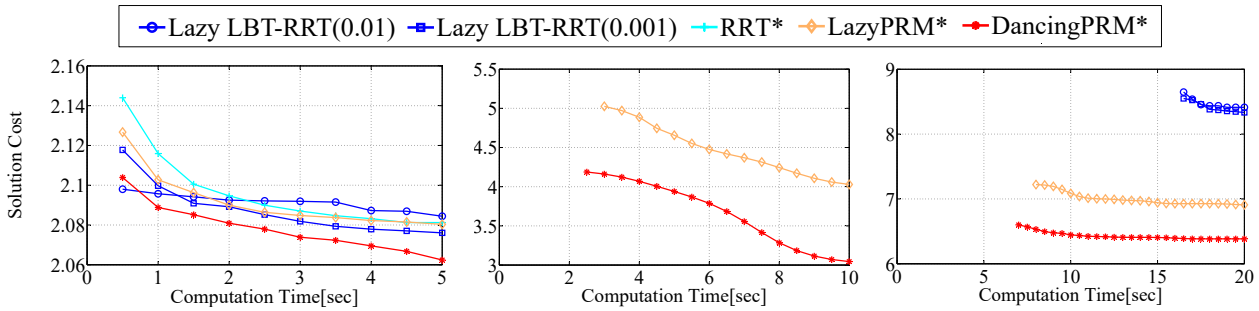
The 6-DOF of Figs. 7(b) and 7(c) has a relatively higher computation cost for collision checking as observed in Fig. 5. The lazyPRM* and DancingPRM* thus show superior performance to the lazy LBT-RRT and RRT* thanks to the local planning minimization by lazy collision checking. Furthermore, the proposed algorithm improves the performance even with the overhead of free space approximation and optimization-based local planning. The details of the benchmark scenes and an example of the approximate configuration free space visualization can be seen in the attached video.

VI. Conclusion and Future Work

In this paper, we present DancingPRM*, an almost-sure asymptotic optimal hybrid planner with sampling-based and optimization-based planning. The proposed algorithm does not depend on any precomputed domain knowledge or expensive computation of obstacle potentials in the configuration space throughout the entire process for seamless integration. We instead utilize the approximate configuration space learned on the fly by empirical collisions and suggest a decentralization of spatial information for efficient access. The limitation is that it may be beyond the capability of a sampling-based approach to approximate configuration free space on the fly in higher dimensional space due to the curse of dimensionality. Further, even though the hypersphere-based representation is simple and effective in low or moderate dimensional space, it would not be an all-purpose solution. Some possible future work includes dimension reduction techniques and adaptive representation of configuration free space for better approximation.

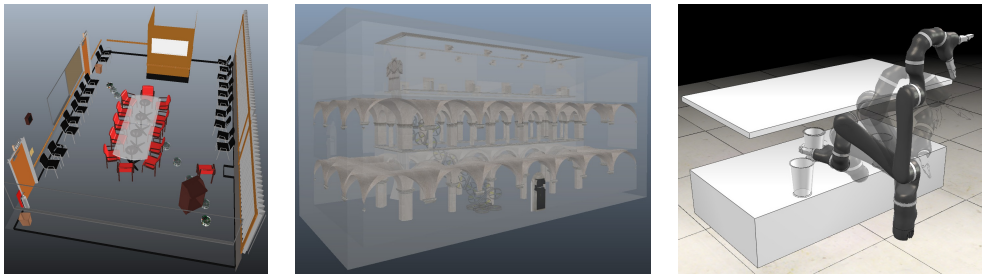
Acknowledgement

We are thankful to the anonymous reviewers for their thoughtful and critical comments, and the researchers



(a) Conference Room (b) Sponza (c) Manipulation

Fig. 6. The plots show the performance of four different asymptotic optimal planners. Each plot corresponds to the scene with same alphabetical numbering in Fig. 7. The algorithms which are not reported on the plot has $50-90\%$ of failure to find an initial solution path.



(a) Conference Room (b) Sponza (c) Manipulation

Fig. 7. a) A 2-DOF mobile robot planning problem in a conference room which has both a wide-open area and narrow passages. To minimize the cost, a robot would pass under the table where chair/table legs compose narrow passages. b) A 6-DOF quadrotor planning in a Sponza scene consisting of over 66K triangles. The Sponza scene is a three-story Gothic building with many pillars, which composes various homotopies of the solution path. c) A 6-DOF manipulation problem to grab a cup in the middle of table through the narrow passage between two other cups and the ceiling. A sequence of afterimage shows an example of the solution path.

whose contributions have inspired our work. This work is supported by SW StarLab program (IITP-2015-0-00199-003), DAPA/DITC (UC160003D) and NRF (NRF-2017M3C4A7066317).

REFERENCES

[1] Steven M LaValle, "Rapidly-exploring random trees a new tool for path planning", Tech. Rep. 98-11, Iowa State University, 1998.
[2] Lydia E Kavraki, Petr Svestka, J-C Latombe, and Mark H Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces", *TRA*, vol. 12, no. 4, pp. 566–580, 1996.
[3] Sertac Karaman and Emilio Frazzoli, "Sampling-based algorithms for optimal motion planning", *IJRR*, vol. 30, no. 7, pp. 846–894, 2011.
[4] Jeong Hwan Jeon, Sertac Karaman, and Emilio Frazzoli, "Anytime computation of time-optimal off-road vehicle maneuvers using the RRT*", in *CDC-ECC*, 2011, pp. 3276–3282.
[5] S. Karaman, M. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the RRT*", in *ICRA*, 2011, pp. 1478–1483.
[6] Joshua Bialkowski, Sertac Karaman, and Emilio Frazzoli, "Massively parallelizing the RRT and the RRT", in *IROS*, 2011, pp. 3513–3518.
[7] Nathan Ratliff, Matt Zucker, J Andrew Bagnell, and Siddhartha Srinivasa, "CHOMP: Gradient optimization techniques for efficient motion planning", in *ICRA*, 2009, pp. 489–494.
[8] Mrinal Kalakrishnan, Sachin Chitta, Evangelos Theodorou, Peter Pastor, and Stefan Schaal, "STOMP: Stochastic trajectory optimization for motion planning", in *ICRA*, 2011, pp. 4569–4574.
[9] Chonhyon Park, Jia Pan, and Dinesh Manocha, "ITOMP: Incremental trajectory optimization for real-time replanning in dynamic environments.", in *ICAPS*, 2012.
[10] A. Shkolnik and R. Tedrake, "Sample-based planning with volumes in configuration space", *arXiv preprint arXiv:1109.3145*, 2011.
[11] Brendan Burns and Oliver Brock, "Sampling-based motion planning using predictive models", in *ICRA*, 2005, pp. 3120–3125.
[12] Brendan Burns and Oliver Brock, "Toward optimal configuration space sampling", in *RSS*, Cambridge, USA, June 2005.
[13] Markus Rickert, Arne Sieverling, and Oliver Brock, "Balancing exploration and exploitation in sampling-based motion planning", *TRO*, vol. 30, no. 6, pp. 1305–1317, 2014.
[14] Joshua Bialkowski, Sertac Karaman, Michael Otte, and Emilio Frazzoli, "Efficient collision checking in sampling-based motion planning", in *WAFR*, 2013, pp. 365–380.
[15] Joshua Bialkowski, Michael Otte, and Emilio Frazzoli, "Free-configuration biased sampling for motion planning", in *IROS*, 2013, pp. 1272–1279.
[16] Jia Pan and Dinesh Manocha, "Fast probabilistic collision checking for sampling-based motion planning using locality-sensitive hashing", *IJRR*, vol. 35, no. 12, pp. 1477–1496, 2016.
[17] Sanjiban Choudhury, Jonathan D Gammell, Timothy D Barfoot, Siddhartha S Srinivasa, and Sebastian Scherer, "Regionally accelerated batch informed trees (RABIT*): A framework to integrate local information into optimal path planning", in *ICRA*, 2016, pp. 4207–4214.
[18] Jonathan D Gammell, Siddhartha S Srinivasa, and Timothy D Barfoot, "Batch informed trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs", in *ICRA*, 2015, pp. 3067–3074.
[19] Matt Zucker, Nathan Ratliff, Anca D Dragan, Mihail Pivtoraiko, Matthew Klingensmith, Christopher M Dellin, J Andrew Bagnell, and Siddhartha S Srinivasa, "CHOMP: Covariant hamiltonian optimization for motion planning", *IJRR*, vol. 32, no. 9-10, pp. 1164–1193, 2013.
[20] Kris Hauser, "Lazy collision checking in asymptotically-optimal motion planning", in *ICRA*, 2015.
[21] Michal Kleinbort, Oren Salzman, and Dan Halperin, "Collision detection or nearest-neighbor search? on the computational bottleneck in sampling-based motion planning", *WAFR*, 2016.
[22] Ioan A Sucan, Mark Moll, and Lydia E Kavraki, "The open motion planning library", *"IEEE R&A Mag."*, vol. 19, no. 4, pp. 72–82, 2012.
[23] M. Freese E. Rohmer, S. P. N. Singh, "V-REP: a versatile and scalable robot simulation framework", in *IROS*, 2013.
[24] Oren Salzman and Dan Halperin, "Asymptotically near-optimal RRT for fast, high-quality, motion planning", in *ICRA*, 2014, pp. 4680–4685.