Abstract—Sampling-based planning algorithms are efficient practical solutions to motion planning challenges. Existing algorithms such as PRM* and RRT* take advantages of random geometric graph theory to answer motion planning queries. This theory requires solving the two-point boundary value problem (BVP) in the state space, which is generally considered to be difficult and impractical. This work presents a different theory of asymptotical optimality. It fills in the gap between optimal kinodynamic planning problems and sampling-based algorithms. The resulting contributions explain some open problems, e.g., the existence of BVP-free asymptotically optimal sampling-based kinodynamic algorithms, properties of a previously proposed heuristic algorithm RRT-BestNear. This work further presents new algorithms STABLE SPARSE-RRT(SST) and SST*. Analysis and experimental results show that SST and SST* are efficient, general, BVP-free, sampling-based optimal kinodynamic planning algorithms that are practical for a great variety of physical systems.

I. INTRODUCTION

Kinodynamic planning is a class of motion planning problems in higher dimensional state space. It is the generalized piano mover’s problem with both kinematic constraints and dynamical constraints. This class of problems is considered difficult to solve, because its simplified kinematic version has been proven to be PSPACE-HARD [26].

Sampling-based planning algorithms have been shown to be practical approaches for quickly returning feasible solutions. The requirement for this class of algorithms is relaxed to probabilistic completeness. It is a very popular approach since it is a practical way to fight against the Curse of Dimensionality. Sampling-based algorithms, e.g., RRT-Connect [18] and PRM [16], are successful for holonomic problems. More recently, the focus of the sampling-based planning community has been on providing optimality. One of the most successful achievements is the discovery of PRM* / RRT* [14] which extends PRM / RRT-Connect to achieve asymptotical optimality.

One of the RRT variants, RRT-Extend [19], has the capability to handle kinodynamic problems. The development of planning algorithms that provide optimal solutions to kinodynamic problems has attracted great attention from many research groups. Kinodynamic problems are quite different from kinematic problems due to the challenge that the two-point boundary value problem BVP cannot be easily solved. One of the consequences of lacking BVP solutions is that building graphs is generally impractical, as well as that no Poisson distribution in the state space is ever available, upon which the current optimality theory (random geometric graph theory) heavily depends. Therefore, existing sampling-based algorithms generally require BVP solvers for solving optimal kinodynamic planning problems.

There are a few existing approaches, e.g., Naïve Random-Tree [10], RRT-BestNear [27] that are underrepresented by the current research community, but are beginning to be explored recently [24]. They cannot be adequately explained by the current optimality theory. This work fills in the gap in the theory of optimality as well as its effectiveness [1]. According to analyses in this paper, it is remarkable that this pipeline starts from Naïve Random-Tree, instead of Rapidly Exploring Random-Tree.

The extended optimal theory relies on the kinodynamic system accessibility (Ball-Box Theorem) and probability theory to achieve probabilistic completeness and asymptotical optimality for all nonholonomic systems where Chow’s condition holds, eliminating the requirement of BVP solutions. Instead, the discussed algorithms in this work only rely on randomized forward propagation, which separates planning algorithms from system specific knowledge.

The first contribution of this paper formally identifies an existing optimal kinodynamic planning algorithm, showing that Naïve Random-Tree is indeed asymptotically optimal. However, it is impractical for requiring intractable number of iterations to converge to (near) optimal solutions. The second contribution of this work uses the extended optimality theory to explain a previously proposed heuristic algorithm RRT-BestNear, showing that it is much more practical.
while delivering certain forms of completeness and optimality guarantees. Then, this work proposes a new algorithm SST, which upgrades RRT-BestNear to further reduce time and space complexities. At last SST∗ is introduced. Analysis shows that SST∗ enjoys the practicability of SST while achieving the uncompromised probabilistic completeness and asymptotical optimality.

Today’s sampling-based algorithms are not generally considered as a stand-alone component. They either require a foreign BVP function, which incorporates a strong form of system dependent domain knowledge, are used as de-coupled approaches, or produce unboundedly random sub-optimal solutions (RRT-Extend). This work opens a door by presenting a class of BVP solution free, direct optimal planning methods, which can serve as a stand-alone global optimizer for a great variety of physical systems.

II. BACKGROUND

Kinodynamic planning problem has been deeply studied during 1980th. It is proved to be at least PSPACE-HARD [26]. Complete methods (e.g. [4]) are shown to be impractical due to their exponential time complexity. Cell decomposition methods (e.g. [3]) have remarkable performance for low dimensional problems. However, it deteriorates significantly as dimension increases. Potential fields based methods (e.g. [17]) are another practical approach. Solutions are built directly on top of the representation of obstacles. However, it suffers from local minima.

Sampling-based algorithms have emerged during late 1990th. The main idea is to avoid explicit representation of the collision-free space. Instead, it employs a set of random sampled vertices in the configuration space or state space. Each of them uniquely represents one possible form of existence of the robot system, and then a data structure is built on top of them. The data structure contains a rich set of paths or trajectories that can be used to answer motion planning queries. This framework provides desirable advantages over traditional methods. Sampling-based methods are a popular approach to fight against the challenge from the curse of dimensions.

Perhaps the first well-known sampling-based algorithm is PRM [16]. It uses a pre-computed graph data structure to represent paths or trajectories. Then it answers queries by connecting new states from/to the graph. RRT-Connect [18] is the next well-known sampling-based algorithm, emerged in late 1990th. It employs a tree, implicitly a trimmed graph, focusing on answering single-query planning problems. RRT-Connect is an incremental fashion algorithm which avoids building a certain size of graph beforehand. It can answer the problem as soon as the tree is rich enough. At the same time, many directions of research were conducted for sampling-based algorithms. [2] investigates the influences of quasi-sampling methods. [20] attempts to learn the cost-to-go metrics to improve explorations and solution qualities.

Arguably, both PRM and RRT-Connect are graph-based algorithms. They rely on the random graph theory to answer feasible motion planning problems. Nevertheless, a different line of research, tree-based algorithms, emerged right after graph-based algorithms. The main two strains, represented by RRT-Extend [19] and Expansive-Space Tree(EST) [10] respectively, were proposed almost simultaneously in approximate 2000. Tree-based algorithms are naturally suitable for kinodynamic problems since they do not require solving the two-point boundary value problem. RRT-Extend inherits the RRT-Connect selection mechanism which results in a rough Voroni bias in exploring the state space, while EST does not have this property.

Perhaps due to this reason, RRT-Extend received dominating attentions from the research community for decades. [8] shows the exponential decay rate of convergence for the failure. [7] designs an anytime fashion for RRT, exhibiting improved experimental performance. Also, RRT-Extend variants have been widely applied to various problem domains (see [8] [22] [23] for examples).

A major event in the history of motion planning research is the discovery of PRM∗ / RRT∗ [14] in 2010. That work formally identifies and develops the theory of sampling-based algorithms. Equipped with the mathematical conclusions, it unifies virtually all previous graph-based approaches, as well as answers many important open questions of that time. One of them is the sub-optimality of RRT-like approaches. Another important influence of [14] is the analysis for the asymptotical optimality property of sampling-based algorithms. It rigorously concludes that graph-based approaches can have guarantees for probabilistical completeness and asymptotical optimality.

[14] has attracted many attentions in the communities.
that sets Naive Random-Tree argument for of the authors, [24] independently provides a different way to support the
sample space, $\Omega$ are subsets of an Euclidean space. Besides, for practical purpose, this work only focuses on manifolds which
re-engineers the Naive Random-Tree framework to improve executions. [15] demonstrates using sub-riemannian geometry theory to approximate cost-to-go metrics to improve performance. [9] focuses on finite time behaviors and properties. Nevertheless, all of them are still graph-based approaches, requiring the capability to build a graph in the first place. There is an early attempt trying to avoid Steering function [13].

On the other hand, there were still gaps between tree-based approaches and solution qualities. Properties of some well-known heuristical algorithms [27] and [21] were still unknown. As the optimality property becomes an important aspect of considerations for planning algorithms, it gradually becomes clear that the previously under-represented Naive Random-Tree (the simplified version of EST) has interesting potentials. Its great advantage, ironically, is hidden in the relatively poor empirical performance. This paper is one of the attempts to fill in this gap. As one of the contributions from this work, rigorous argument for Naive Random-Tree being asymptotically optimal is provided. This EST variant is, perhaps, the first sampling-based kinodynamic algorithm that offers the asymptotical optimality guarantee. However, it is also shown by this work that Naive Random-Tree is computationally intractable.

From a practicability perspective, an optimal-but-intractable algorithm provides no more benefits than suboptimal-but-practical ones. Therefore, it is critical to continue searching for optimal algorithms while being efficient, general, as well as mathematically provable. As the second part of this work, a well-known heuristical algorithm [27] is formally analyzed. Then based on the extended optimality theory, and on top of a previously proposed algorithm Sparse-RRT from the current authors, this work proposes a new family of tree-based sampling-based algorithms: SST and SST*. Analysis and experimental results argue and verify that SST* is probabilistically complete and asymptotically optimal, as well as efficient and practical.

III. PROBLEM SETUP

This paper considers dynamical systems that respect time-invariant differential equations of the following form:

$$\dot{x}(t) = f(x(t), u(t)), \quad x(t) \in \mathbb{X}, \quad u(t) \in \mathbb{U} \quad (1)$$

where $x(t) \in \mathbb{X} \subseteq \mathbb{R}^d$ and $u(t) \in \mathbb{U} \subseteq \mathbb{R}^e$. The collision-free part of the state space is $\mathbb{X}_f$. The Euclidean norm is denoted by $\|\cdot\|$. The closed ball of radius $r > 0$ centered at $x$ is $B_r(x)$. Let $\mu(\mathbb{X})$ denote the Lebesgue measurement of $\mathbb{X}$.

This work makes the same assumptions as in [15] such that sets $\mathbb{X}$ and $\mathbb{U}$ are smooth manifolds (see [15] for the definitions and examples of a smooth manifold). Besides, for practical purpose, this work only focuses on manifolds which are subsets of an Euclidean space.

Let the triple $(\Omega, \mathcal{F}, \mathbb{P})$ denotes a probability space, where $\Omega$ is a sample space, $\mathcal{F}$ is a $\sigma$-field of subsets of $\Omega$, and $\mathbb{P}$ is a probability measure on $(\Omega, \mathcal{F})$. A sequence of random variables $\{\mathbb{Y}_i\}_{i \in \mathbb{N}}$ is said to converge surely to a random variable $\mathbb{Y}$ if $\lim_{i \to \infty} \mathbb{Y}_i(\omega) = \mathbb{Y}(\omega)$ for all $\omega \in \Omega$. The sequence almost surely converges to $\mathbb{Y}$ if $\mathbb{P}(\lim_{i \to \infty} \mathbb{Y}_i = \mathbb{Y}) = 1$.

Assumption 1: The dynamical system described by Equation (1) satisfies the following properties:

- The system satisfies Chow’s condition. In other words, it is Small-time Locally Attainable (or Accessible) [5].
- It has bounded second derivative: $\|\dot{x}(t)\| \leq M_2 \in \mathbb{R}^+$. It is Lipschitz Continuous for both of its arguments. Specifically, $\exists K_1 > 0$ and $\exists K_2 > 0$:

$$||f(x_0, u_0) - f(x_1, u_0)|| \leq K_1 ||u_0 - u_1||, \quad ||f(x_0, u_0) - f(x_1, u_0)|| \leq K_2 ||x_0 - x_1||.$$  

The focus in this paper will be on trajectories for such systems with a certain clearance from obstacles.

Definition 1: ($\delta$-Robust Trajectories) Given a dynamical system of the form in Eq. (1) the collision-free subset $\mathbb{X}_f \subset \mathbb{X}$ and clearance value $\delta$, a $\delta$-robust trajectory is a function $\pi : [0, t_f] \to \mathbb{X}_f$ that satisfies Eq. (1) and the following property for all $x_{obs} \in \mathbb{X} \setminus \mathbb{X}_f$:

$$min(||x(t) - x_{obs}||) \geq \delta.$$  

For such trajectories, it is possible to define a variation of the motion planning problem with dynamics.

Definition 2: ($\delta$-Robust Feasible Motion Planning) Given a dynamical system of Eq. (1) the collision-free subset $\mathbb{X}_f \subset \mathbb{X}$, an initial state $x_0 \in \mathbb{X}_f$, a goal region $\mathbb{X}_G \subset \mathbb{X}_f$, and a clearance $\delta$, find a solution trajectory $\pi : [0, t_*] \to \mathbb{X}_f$, for which $\pi(0) = x_0$ and $\pi(t_*) \in \mathbb{X}_G$, given that a $\delta$-robust trajectory that connects $x_0$ with a state in $\mathbb{X}_G$ exists.

Typically sampling-based algorithms generate trajectories $\pi$ that correspond to piece-wise constant control functions $v \in [0, t_f] \to \mathbb{U}$.

Definition 3: (Piece-wise Constant Control Function) A piece-wise constant control function $v$ related to a trajectory $\pi$ can be expressed as a vector $v_m = \{u_1, u_2, \ldots, u_n\}$, where each control $u_i \in \mathbb{U}$ is executed for a duration $\Delta t > 0$ to generate a trajectory $\pi$ according to Eq. (1). Clearly, $v_m$ has $l \times m$ dimensions, denoted by $w$.

Assumption 2: For a $\delta$-robust feasible motion planning problem, there exists a $\delta$-robust trajectory $\pi$ generated by a piecewise-constant control function $v$.

Sampling-based algorithms, referred generally in this section as ALG, typically maintain a graph data structure of feasible trajectories during their operation. This paper considers the following properties of sampling-based algorithms.

Definition 4: (Probabilistic $\delta$-Robust Completeness) Let $\Pi^{ALG}_n$ denote the set of trajectories discovered by an algorithm ALG at iteration $n$. Algorithm ALG is probabilistically $\delta$-robustly complete, if, for any $\delta$-robustly feasible motion planning problem $(\mathbb{X}_f, x_0, \mathbb{X}_G, \delta)$ the following holds:

$$\lim_{n \to \infty} \mathbb{P}(\exists \pi \in \Pi^{ALG}_n : \pi \text{ solution to } (\mathbb{X}_f, x_0, \mathbb{X}_G, \delta)) = 1.$$  

Def. 4 relaxes the concept of probabilistic completeness for algorithms whose properties depend on the clearance of trajectories they can discover. An algorithm that is probabilistically $\delta$-robustly complete only guarantees it will eventually
find trajectories if one with clearance of $\delta$ exists. The following discussion relates to the cost function of a trajectory $\pi$.

**Assumption 3:** The cost function $cost(\pi)$ of a trajectory in this work is assumed to be its duration. Thus, it is Lipschitz continuous. Specifically, $\exists K_c > 0$:

$$|cost(\pi_0) - cost(\pi_1)| \leq K_c \cdot sup_{t \in [0,T]}(||\pi_0(t) - \pi_1(t)||).$$

As a result, for all $\pi_1$, $\pi_2$ and their concatenation $\pi_1|\pi_2$ (i.e., following trajectory $\pi_2$ after trajectory $\pi_1$) the cost function satisfies:

- $cost(\pi_1|\pi_2) = cost(\pi_1) + cost(\pi_2)$ (additivity)
- $cost(\pi_1) \leq cost(\pi_1|\pi_2)$ (monotonicity)

Then, it is also possible to relax the property of asymptotic optimality and allow some tolerance, which depends on clearance.

**Definition 5:** (Asymptotic $\delta$-robust Near-Optimality) Let $c^*$ denote the minimum cost over all solution trajectories for a $\delta$-robust feasible motion planning problem ($X_f$, $x_0$, $X_G$, $\delta$). Let $Y_{ALG}$ denote a random variable that represents the minimum cost value among all trajectories returned by algorithm $ALG$ at iteration $n$ for the same problem. $ALG$ is asymptotically $\delta$-robust near-optimal if for all independent runs:

$$\mathbb{P}\left(\lim_{n \to \infty} sup_n Y_{ALG} \leq h(c^*, \delta)\right) = 1$$

where $h : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$ is a function of the optimum cost and $\delta$ clearance ($h(c^*, \delta) \geq c^*$).

The following version for the $h$ function is considered for some constant $\alpha \geq 0$:

$$h(c^*, \delta) = (1 + \alpha \cdot \delta) \cdot c^*,$$

when $ALG$ is asymptotically $\delta$-robust near-optimal with a multiplicative error.

The property of asymptotically $\delta$-robust near-optimal (or simply asymptotically near-optimal) guarantees that the quality of the returned solution asymptotically is bounded between the optional and a known function. The solution cost value is still allowed some randomness. Recall that $RRT$-Connect returns solutions with random quality and the randomness is unbounded. The contribution of Def. 5 is that it characterizes the lower bound of the quality. And this definition of optimality implies that the quality bound is controllable through the parameter $\delta$.

If it is possible to argue that an algorithm satisfies the last two properties for all decreasing values of the clearance parameter $\delta$, then this algorithm satisfies the traditional properties of probabilistic completeness and asymptotic optimality.

Please note that, the set of trajectories $\Pi_{n}^{ALG}$ only contains trajectories that are generated by piece-wise constant control function of minimum duration $\Delta t$. Therefore, from now on, asymptotically optimal and asymptotically near-optimal in this paper are in the sense of $\Delta t$ duration piece-wise constant control function. Note that there are other forms of parametrizing trajectory generation. This paper only considers discretizing duration for simplicity and practical purposes.

IV. PROPERTIES OF MONTE-CARLO PROPAGATION

**Definition 6:** (Covering Balls) Given a trajectory $\pi^*(t) : [0, T^*] \rightarrow X_f$, clearance $\delta \in \mathbb{R}^+$ and time step $T$, the set of covering balls $\mathbb{B}(\pi(t), \delta, T)$ is defined as a set of $M + 1$ balls $\{B_\delta(x_0), B_\delta(x_1), ..., B_\delta(x_M)\}$ of radius $\delta$, such that $B_\delta(x_m)$ is centered at $x_i = \pi(iT) \forall i \in [0,M]$, where $M = \frac{T^*}{T}$.

This construction gives rise to the following definition:

**Definition 7:** ($\delta$-Similar Trajectories) Trajectories $\pi, \pi'$ are $\delta$-similar if for a continuous scaling function $\sigma : [0, \delta] \rightarrow [0, t']$, it is true: $\pi'(\sigma(t)) \in B_\delta(\pi(t))$.

The notion of a covering ball sequence as used in this paper is shown in Fig. 2(left).

**Algorithm 1:** MonteCarlo-Prop($x_{prop}$, $U$, $T_{prop}$)

1. $t \leftarrow \text{Sample}(0, T_{prop}); \ x_m \leftarrow \text{Sample}(U, t)$;
2. return $x_{new} \leftarrow \int_{0}^{t} f(x(t), v_m(t)) \ dt + x_{prop}$;

To argue that Naïve Random-Tree finds $\delta$-similar trajectories to optimal ones, it is helpful to argue that MonteCarlo-Prop can eventually generate in a piece-wise fashion a $\delta$-similar trajectory to any trajectory. The following lemma, which corresponds to Fig. 2(right), argues that there is a positive measure of $\delta$-similar trajectories that can be discovered for any $\delta$ below a threshold $\delta_0$ that is system dependent.

**Lemma 1:** Let there be a trajectory $\pi$ satisfying Eq. 1

Then there exists a positive value $\delta_0$, such that: $\forall \delta \in (0, \delta_0], \ \forall x_0' \in B_\delta(\pi(0)), \ \forall x_1' \in B_\delta(\pi(t))$, there exists a trajectory $\pi'$, so that: (i) $\pi'(0) = x_0'$ and $\pi'(t') = x_1'$; (ii) $\pi$ and $\pi'$ are $\delta$-similar trajectories.

Informally speaking, Chow’s condition (assumption $I$) implies that Ball Box theorem holds. It also implies that the manifold $X_f$ is regular and involutory. In the case of real-analytic control-affine system, it is STL, if and only if the distribution satisfies Chow’s condition. Assume every state on the optimal trajectory is a regular point. Then, the subriemannian ball up to a small constant radius $t_s$ contains a weighted box of the same dimension of the state space and it is oriented according to vector fields of the Lie brackets. The bases are real analytical. Therefore there exists a open neighborhood of each point $x$ such that the bases evaluated at a different point $x'$ converge to the bases at $x$ as $x' \rightarrow x$. Then, the weighted boxes centered by two sufficiently close states has non-empty intersection. It implies that a hyper ball of some positive radius $\delta_0$ can be fitted into this intersection region. In sum, there exists two sufficiently close hyper ball regions on the optimal trajectory such that between any point $x$ in one ball and any point in the other ball there exists a horizontal curve.
and the length of the curve is less or equals to the radius \( t \), of the sub-riemannian ball. Then concatenating all hyper balls, for the optimal trajectory, there exists \( \delta \)-similar trajectories. From a different perspective, Lemma 1 is a necessary condition for all systems where RRT* converges.

Given the above lemma, the critical property of MonteCarlo-Prop is the following:

**Theorem 1:** Given a trajectory \( \pi \) of duration \( T \), the success probability for MonteCarlo-Prop to generate a \( \delta \)-similar trajectory to \( \pi \) when called from an input state \( x_{prop} \in B_\delta(\pi(0)) \) and for a propagation duration \( T_{prop} > T \) is lower bounded by a positive value \( \rho_0 > 0 \).

The proof can be found in the appendix (Section XI). The above theorem argues that by calling MonteCarlo-Prop multiple times from a state within \( B_\delta(\pi(0)) \), it is possible to eventually generate a \( \delta \)-similar trajectory \( \pi'(t) \) to any trajectory \( \pi(t) \), such that the Euclidean distance between the end states of \( \pi(T) \) and \( \pi'(T) \) is bounded. The cost difference between the pair of \( \delta \)-similar trajectories is also bounded (Assumption 2).

V. PROPERTIES OF NAÏVE RANDOM TREE

This section considers a naïve, impractical sampling-based tree algorithm shown in Alg. 2 which does not employ a steering function. Instead, it randomly selects a reachable state in the existing tree (line 3) and applies random propagation (line 4) to extend it (lines 5-7). The propagation process is illustrated in Alg. 2. Appropriate modifications on top of Naïve Random Tree are possible, giving birth to efficient near-optimal kinodynamic planners.

Algorithm 2: Naïve Random Tree (\( (X_f, U, x_0, T_{prop}, N) \))

1. \( G = \{V \leftarrow \{x_0\}, E \leftarrow \emptyset\} \);
2. for \( N \) iterations do 
   3. \( x_{nearest} \leftarrow \text{Uniform_Sampling}(V) \);
   4. \( x_{new} \leftarrow \text{MonteCarlo-Prop}(x_{nearest}, U, T_{prop}) \);
   5. if CollisionFree((\( x_{nearest}, x_{new} \), \( X_f \)) then 
      6. \( V \leftarrow V \cup \{x_{new}\}, E \leftarrow E \cup \{(x_{nearest}, x_{new})\} \);
   7. return \( G(V, E) \);

A. Probabilistical Completeness of Naïve Random Tree

The following discussion argues that this algorithm eventually generates trajectories that lie within a \( \delta \)-distance of optimum ones.

**Theorem 2:** Naïve Random Tree is probabilistically complete.

**Proof:** Let \( \pi^* \) denote an optimal trajectory of duration \( T^* \) for a \( \delta \)-robustly feasible motion planning problem \( (X_f, x_0, X_G, \delta, T) \). Consider the covering balls \( B(\pi^*(t), \delta, T) \).

To compute the the probability that a \( \delta \)-similar trajectory \( \pi \) to \( \pi^* \) is generated by the algorithm after \( n \) iterations, consider the sequence of states \( \{x_0', \ldots, x_M\} \), so that \( x_0' = \pi^*(0) \) and \( x_i' \) is a state generated by the algorithm in \( B_\delta(x_i) \), such that the edge \( (x_i' - 1, x_i') \) is a \( \delta \)-similar trajectory to the segment \( \{\pi^*(i - 1)T, \pi^*(iT)\} \) of the optimum trajectory.

Note that the maximum duration \( T \) of a trajectory connecting two states \( (x_{i-1}', x_i') \) in consecutive balls (as in Fig. 2(right)) must be less than the maximum propagation length \( T_{prop} \) considered in MonteCarlo-Prop. This means that given a propagation duration \( T_{prop} > 0 \) for MonteCarlo-Prop, the value \( T \) must be such that \( 0 < T < T_{prop} \). In other words, an edge \( (x_{i-1}', x_i') \) can reach \( B_i \) for sure given the right controls. For \( \delta \) chosen small enough, this is the case.

Let \( A_k^{(n)} \) denote the event that at the \( n \)th iteration, the algorithm generates one \( \delta \)-similar trajectory \( \pi \) to \( x_{k-1}' \xrightarrow{} x_k' \), such that \( \pi(0) \in B_\delta(x_{k-1}') \) and for some \( T_{end} > 0 \), \( \pi(T_{end}) \in B_\delta(x_k) \).

Let \( E_k^{(n)} \) denote the event that from iteration 1 to \( n \), the algorithm generates at least one such trajectory. Then, the event \( \neg E_k^{(n)} \) is the event the algorithm fails to generate any near-optimal trajectory inside \( B_\delta(x_k') \) after \( n \) iterations, which only happens when all the \( n \) consecutive iterations fail one after another, i.e.,

\[
\neg E_k^{(n)} = \neg A_k^{(1)} \cap \neg A_k^{(2)} \cap \ldots \cap \neg A_k^{(n)}
\]

\[
P(\neg E_k^{(n)}) = P(\neg A_k^{(1)}) \cdot P(\neg A_k^{(2)}) \cdot \ldots \cdot P(\neg A_k^{(n)})
\]

\[
\leq P(\neg A_k^{(1)}) \cdot P(E_k^{(1)}(1) - \frac{\rho_{\delta \rightarrow \delta}}{n})
\]

\[
= 1 - P(E_k^{(1)}(1) - \frac{\rho_{\delta \rightarrow \delta}}{n})
\]

(3)

The probability that \( \neg A_k^{(n)} \) happens given \( \bigcap_{j=1}^{n-1} \neg A_k^{(j)} \) is equivalent to the probability of failing to generating a trajectory to the \( B_\delta(x_{k-1}') \) plus the probability that a trajectory has been generated to \( B_\delta(x_{k}') \), but fails to generate a new trajectory to \( B_\delta(x_{k}') \), i.e.,

\[
P(\neg A_k^{(n)}) \bigcap_{j=1}^{n-1} \neg A_k^{(j)}
\]

\[
= P(\neg E_k^{(n)}) + P(E_k^{(1)}(1) - \frac{\rho_{\delta \rightarrow \delta}}{n})
\]

\[
= 1 - P(E_k^{(1)}(1) - \frac{\rho_{\delta \rightarrow \delta}}{n})
\]

(3)

Therefore, using Eq. 2 and Eq. 3

\[
P(E_k^{(1)}) \geq 1 - \prod_{j=1}^{n} (1 - P(E_k^{(j)}(1) - \frac{\rho_{\delta \rightarrow \delta}}{j}))
\]

(4)

For the base case, \( P(E_k^{(1)}) = 1 \) because \( x_0 \) is always in \( B_\delta(x_0) \). Then, consider event \( E_1 \) from iteration 1 to \( n \) using Eq. 4, and set \( y_1^{(n)} = \prod_{j=1}^{n} (1 - \frac{\rho_{\delta \rightarrow \delta}}{j}) \),

\[
P(E_1^{(n)}) \geq 1 - \prod_{j=1}^{n} (1 - \frac{\rho_{\delta \rightarrow \delta}}{j}) = 1 - y_1^{(n)}
\]

(4)
The logarithm of \( y_1^{(n)} \) behaves as the following,

\[
\log y_1^{(n)} = \log \prod_{j=1}^{n} \left( 1 - \frac{\rho \delta \to \delta}{j} \right) = \sum_{j=1}^{n} \log \left( 1 - \frac{\rho \delta \to \delta}{j} \right)
\]

\[
< \sum_{j=1}^{n} - \frac{\rho \delta \to \delta}{j} = - \rho \delta \to \delta \cdot \sum_{j=1}^{n} \frac{1}{j}
\]

Clearly, Eq. 5 diverges as \( n \to \infty \),

\[
\lim_{n \to \infty} \log y_1^{(n)} < \lim_{n \to \infty} - \rho \delta \to \delta \cdot \sum_{j=1}^{n} \frac{1}{j} = - \infty
\]

\[
\iff \lim_{n \to \infty} y_1^{(n)} = 0
\]

Then,

\[
\lim_{n \to \infty} P(E_1^{(n)}) \geq 1 - 0 = 1
\]

**Now consider the induction step.** If \( \lim_{n \to \infty} P(E_k^{(n)}) = 1 \), we need to show that the same will be true for \( E_k^{(n)} \). Similarly, set \( y_k^{(n)} = \prod_{j=1}^{n} (1 - P(E_k^{(j)}) \cdot \frac{\rho \delta \to \delta}{j}) \). The logarithm of \( y_k^{(n)} \) behaves as follows,

\[
\log y_k^{(n)} = \log \prod_{j=1}^{n} \left( 1 - P(E_k^{(j)}) \cdot \frac{\rho \delta \to \delta}{j} \right)
\]

\[
= \sum_{j=1}^{n} \log \left( 1 - P(E_k^{(j)}) \cdot \frac{\rho \delta \to \delta}{j} \right) < - \rho \delta \to \delta \cdot \sum_{j=1}^{n} \frac{P(E_k^{(j)})}{j}
\]

(7)

Next, we want to show that for any constant \( c_1 \in (0, 1) \), there is,

\[
\sum_{j=1}^{\infty} \frac{P(E_k^{(j)})}{j} > \sum_{j=1}^{\infty} \frac{c_1}{j}
\]

To show the above expression holds, let \( c_2 \) be another constant such that \( c_1 < c_2 < 1 \). Clearly, Eq. 4 indicates that \( E_k^{(n)} \) monotonically increasing when \( \rho \delta \to \delta > 0 \) and \( P(E_k^{(j)}) > 0 \). Since the induction assumption \( \lim_{n \to \infty} P(E_k^{(n)}) = 1 \), then there exist corresponding numbers \( j_1 < j_2 \) such that \( c_1 \geq P(E_k^{(j_1)}) \), \( c_1 < P(E_k^{(j_2)}) \) and \( c_2 \geq P(E_k^{(j_2-1)}) \), \( c_2 < P(E_k^{(j_2)}) \).

Now examine the following summation from \( j_1 \) to \( \infty \), according to the definition of summation,

\[
\sum_{j=1}^{\infty} \frac{P(E_k^{(j)})}{j} - \frac{c_1}{j} = \sum_{j=j_1}^{j_2-1} \frac{P(E_k^{(j)})}{j} - \frac{c_1}{j} + \sum_{j=j_2}^{\infty} \frac{P(E_k^{(j)})}{j} - \frac{c_1}{j}
\]

\[
> j_2 - 1 \frac{P(E_k^{(j)})}{j} - \frac{c_1}{j} + \sum_{j=j_2}^{\infty} \frac{c_2 - c_1}{j} = \infty
\]

Clearly, the first term in [9] is positive. The second term in [9] diverges to infinity. Then [9] is positive and unbounded.

Consider the following summation from 1 to \( j_1 - 1 \),

\[
- \infty < \sum_{j=1}^{j_1-1} \frac{P(E_k^{(j)})}{j} - c_1 < 0
\]

(10)

Clearly, [10] is negative but bounded, since there is only finite terms.

Then combine [10] and [8] there is,

\[
\sum_{j=1}^{\infty} \frac{P(E_k^{(j)})}{j} - \frac{c_1}{j} > 0
\]

\[
\iff \sum_{j=1}^{\infty} \frac{P(E_k^{(j)})}{j} > \sum_{j=1}^{\infty} \frac{c_1}{j}
\]

(11)

Combining [7] and [11] there is,

\[
\lim_{n \to \infty} \log y_k^{(n)} = - \rho \delta \to \delta \cdot \lim_{n \to \infty} \sum_{j=1}^{n} \frac{P(E_k^{(j)})}{j}
\]

\[
< - \rho \delta \to \delta \cdot \lim_{n \to \infty} \sum_{j=1}^{n} \frac{c_1}{j} = - \infty
\]

\[
\iff \lim_{n \to \infty} y_k^{(n)} = 0
\]

Then, the induction step holds, e.g.,

\[
\lim_{n \to \infty} P(E_k^{(n)}) = 1 - 0 = 1, \text{ if } \lim_{n \to \infty} P(E_k^{(n)}) = 1
\]

(12)

Both of the base case [6] and induction step [12] hold. Therefore, it is true that

\[
\lim_{n \to \infty} P(E_k^{(n)}) = 1 \implies \liminf_{n \to \infty} P(E_k^{(n)}) = 1
\]

(13)

Therefore, the Naïve Random-Tree algorithm, for any given \( 0 < \delta \leq \delta_0 \), is probabilistically \( \delta \)-robustly complete.

Now consider such a sequence of \( \delta \)-robustly feasible motion planning problems \( \langle X_j, x_0, X_G, \delta \rangle \), where \( \delta \) reduces to 0, where the corresponding covering balls for the optimum path shrink. Unlike RRT*, it is important to note that only the radius of balls shrinks. The distance between consecutive balls stays the same. Clearly, Eq. [12] holds for all instances of this sequence of \( \delta \)-robustly feasible motion planning problems. Therefore, Naïve Random-Tree is also probabilistically complete. ■

Now consider an interesting property of Naïve Random-Tree. Similar to RRT-Connect, each vertex in \( V \) of Naïve Random-Tree asymptotically has unbounded degree.

**Theorem 3:** For any state \( x_i \in V \) such that \( x_i \) is added into \( V \) at iteration \( i \), then Naïve Random-Tree will select \( x_i \) for MonteCarlo-Prop infinitely often as the execution times \( n \to \infty \).

\[
P(\limsup_{n \to \infty} \{ x_i \text{ is selected} \}) = 1
\]

**Proof:** For state \( x_i \) which is added to \( V \) at iteration \( i \) (clearly \( i \leq n \)), let \( S_i^{(n)} \) denote the event such that \( x_i \) being selected for MonteCarlo-Prop at iteration \( n \). Since, during each iteration, the algorithm uniform-randomly selects a state for
MonteCarlo-Prop. The probability of such event can be written as

$$\mathbb{P}(S_i^{(n)}) = \frac{1}{n}$$

The summation of the first \( n - i \) terms of the sequence is,

$$\sum_{j=1}^{n} \mathbb{P}(S_i^{(j)}) = \sum_{j=1}^{n} \frac{1}{j} - \sum_{j=1}^{i} \frac{1}{j} \quad (14)$$

The first term on the right side is harmonic series, and the second term is the \( i \)-th harmonic number.

The property of harmonic series is known as

$$\sum_{j=1}^{\infty} \frac{1}{j} = \ln(n) + c_\gamma + \epsilon_n$$

where \( c_\gamma = 0.577 \ldots \) (Euler-Mascheroni constant)

$$\epsilon_n \sim \frac{1}{2n} \quad \text{such that} \quad \lim_{n \to \infty} \epsilon_n = 0$$

Therefore, (14) diverges as \( n \to \infty \).

$$\sum_{j=1}^{\infty} \mathbb{P}(S_i^{(j)}) = \lim_{n \to \infty} \left[ \ln(n) + c_\gamma + \epsilon_n - \sum_{j=1}^{\infty} \frac{1}{j} \right] \geq +\infty$$

Clearly, selecting \( x_i \) is independent at any two different iterations (and combinations) after \( x_i \) being extended. This is because the algorithm uniform-randomly picks one vertex among existing ones during each iteration.

Then, according to the second Borel-Cantelli lemma,

$$\mathbb{P}(\limsup_{n \to \infty} S_i^{(n)}) = 1$$

Therefore, \( x_i \) shall be selected for MonteCarlo-Prop infinitely often as the number of execution times \( n \to \infty \).

B. Asymptotical Optimality of Naïve Random Tree

Theorem 4: Naïve Random-Tree is asymptotically optimal.

Proof: Thm. 2 indicates that a \( \delta \)-similar trajectory to an optimal trajectory \( \pi^* \) of duration \( T^* \) almost surely exists and is discovered by Naïve Random-Tree. According to the definition of \( \delta \)-similar trajectories and assumption 3

$$|\text{cost}(\pi) - \text{cost}(\pi^*)| \leq K_\epsilon \cdot \delta$$

Then:

$$\text{cost}(\pi) \leq \text{cost}(\pi^*) + K_\epsilon \cdot \delta$$

Therefore, event \( E_k^{(\infty)} \) implies event \( \{Y_{n+1}^{RT} \leq \text{cost}(\pi^*) + k \cdot K_\epsilon \cdot \delta\} \), where \( k = \frac{T^*}{\delta} \). In other words, Thm. 2 implies that:

$$\mathbb{P}(\limsup_{n \to \infty} Y_{n+1}^{RT} \leq (1 + \frac{K_\epsilon \cdot \delta}{T^*}) \cdot \text{cost}(\pi^*)) = 1 \quad (15)$$

Therefore, the Naïve Random-Tree is asymptotically \( \delta \)-robust near-optimal for the given \( \delta \). In fact, however, due to Thms. 1 and 2 the above holds true for any \( \delta > 0 \). Note that \( T^* \) is the duration of the optimal trajectory between \( x_i^* \) and \( x_{i+1}^* \). And, most importantly, \( T \) is determined by MonteCarlo-Prop, which means \( T \) does not shrink when \( \delta \) decreases. Then as \( \delta \to 0 \) there is:

$$\mathbb{P}(\limsup_{n \to \infty} Y_{n+1}^{RT} \leq \text{cost}(\pi^*(t))) = 1$$

Theorem 4 states that it is possible to achieve in a rather naïve way probabilistic completeness and asymptotic optimality. The following section, however, explains that this approach is indeed impractical.

C. Convergence Rate of Naïve Random Tree

Consider the rate of convergence for the probability \( \mathbb{P}(E_k^{(n)}) \) where \( k \) denotes the \( k \)-th ball and \( n \) is the number of iterations. Given Thm. 3, \( \mathbb{P}(E_k^{(n)}) \) converges to 1. Then the following theorem is true.

Theorem 5: For the worst case, the \( k \)-th segments of the trajectory returned by Naïve Random-Tree converges sub-linearly to the near optimal solution. e.g.,

$$\lim_{n \to \infty} \frac{|\mathbb{P}(E_k^{(n+1)}) - 1|}{|\mathbb{P}(E_k^{(n)}) - 1|} = 1$$

Proof: Applying the boundary condition of Thm. 4, the worst case scenario is that only one state us considered to exist inside each ball region (even there could be multiply states). Without any other system properties, this is the only available information. Then consider the ratio of the probabilities between iteration \( k + 1 \) and \( k \).

$$\frac{|\mathbb{P}(E_k^{(n+1)}) - 1|}{|\mathbb{P}(E_k^{(n)}) - 1|} = \frac{\prod_{j=1}^{n+1} (1 - \frac{\mathbb{P}(E_k^{(j)}) \cdot \rho}{j})}{\prod_{j=1}^{n} (1 - \frac{\mathbb{P}(E_k^{(j)}) \cdot \rho}{j})} = 1 - \frac{\mathbb{P}(E_k^{(n+1)}) \cdot \rho}{n + 1}$$

Clearly, \( \mathbb{P}(E_k^{(n)}) \) is at most 1 and \( \rho \) is less than 1. Then,

$$\lim_{n \to \infty} \frac{|\mathbb{P}(E_k^{(n+1)}) - 1|}{|\mathbb{P}(E_k^{(n)}) - 1|} = \lim_{n \to \infty} 1 - \frac{\mathbb{P}(E_k^{(n+1)}) \cdot \rho}{n + 1} = 1$$

Additionally, we want to examine the sub-linearity of the convergence rate.

Theorem 6: For the worst case, the \( k \)-th segments of the trajectory returned by Naïve Random-Tree converges logarithmically to the near optimal solution. e.g.,

$$\lim_{n \to \infty} \frac{|\mathbb{P}(E_k^{(n+2)}) - \mathbb{P}(E_k^{(n+1)})|}{|\mathbb{P}(E_k^{(n+1)}) - \mathbb{P}(E_k^{(n)})|} = 1$$

Proof: Similarly, applying the boundary condition of Eq. 4 there is,
\[ \left| \frac{\mathbb{P}(E_k^{(n+2)}) - \mathbb{P}(E_k^{(n+1)})}{\mathbb{P}(E_k^{(n+1)}) - \mathbb{P}(E_k^{(n)})} \right| = \prod_{j=1}^{n+1} \left( 1 - \frac{\mathbb{P}(E_j^{(\rho)})}{\mathbb{P}(E_j^{(\rho)})} \right) - \prod_{j=1}^{n+2} \left( 1 - \frac{\mathbb{P}(E_j^{(\rho)})}{\mathbb{P}(E_j^{(\rho)})} \right) = (1 - \frac{\mathbb{P}(E_k^{(n+1)})}{n+1} \cdot \frac{\mathbb{P}(E_k^{(n+2)})}{n+2}) \cdot (1 - \frac{\mathbb{P}(E_k^{(n+1)})}{n+1} \cdot \frac{\mathbb{P}(E_k^{(n+2)})}{n+2}) \]

\[ = (1 - \frac{\mathbb{P}(E_k^{(n+1)})}{n+1} \cdot \frac{\mathbb{P}(E_k^{(n+2)})}{n+1} \cdot \frac{\mathbb{P}(E_k^{(n+2)})}{n+1}) = 1 \cdot 1 \cdot 1 = 1 \]

Clearly Eq. 16 as \( n \to \infty \), \( (1 - \frac{\mathbb{P}(E_k^{(n+1)})}{n+1}) \) converges to 1, as well as \( \frac{\mathbb{P}(E_k^{(n+2)})}{n+2} \) converges to 1. And given Thm. 2 such that the limit for \( \mathbb{P}(E_k^{(n+1)}) \) exists and non-zero, as well as the monotonicity of \( \mathbb{P}(E_k^{(n)}) \), it is the necessary condition that \( \frac{\mathbb{P}(E_k^{(n+1)})}{\mathbb{P}(E_k^{(n+2)})} \) converges to 1. Therefore, there is,

\[ \lim_{n \to \infty} \left| \frac{\mathbb{P}(E_k^{(n+2)}) - \mathbb{P}(E_k^{(n+1)})}{\mathbb{P}(E_k^{(n+1)}) - \mathbb{P}(E_k^{(n)})} \right| = 1 \cdot 1 \cdot 1 = 1 \]

Thm. 3 indicates that the completeness and optimality properties of Naïve Random-Tree does not depend on the assumption of the STL property of the system. Informally speaking, all possible collision-free states will be given infinite number of times to propagate. And the duration of the propagation does not decrease, unlike in RRT-Connect where the expected length of new branches converge to 0[14]. The assumption of Lipschitz continuity of the system is enough to guarantee optimality. Due to this reason, Naïve Random-Tree is trivially asymptotically optimal. However, this section shows it takes an intractable number of iterations for Naïve Random-Tree to converge to an optimal solution.

Let \( p \) be the probability of an event to happen. Then the expected number of independent trials for that event to happen is \( 1/p \). It can be shown that the probability of such an event happening converges to and is always greater than \( 1 - e^{-1} \approx 63.21\% \), after \( 1/p \) independent trials, as \( p \to 0 \) [9].

Consider event \( A_1 \) and set the success probability of the MonteCarlo-Prop function to be \( \rho \). If \( x_0 \) is selected for MonteCarlo-Prop, then the probability of \( \mathbb{P}(A_1 \mid (x_0 \text{ is selected})) = \rho \). Then the “expected number” of times we need to select \( x_0 \) for \( A_1 \) to happen is \( 1/\rho \). The expected number of times that \( x_0 \) is selected after \( n \) iterations is \( \sum_{i=1}^{n} \frac{1}{\rho} \). This yields the following expression for sufficiently large \( n \):

\[ \frac{1}{\rho} = \sum_{i=1}^{n} \frac{1}{\rho} \approx \ln(n) + c_{\gamma} \]

where \( c_{\gamma} \) is the Euler-Mascheroni constant, which yields:

\[ n \approx e^{(\ln(n) + c_{\gamma})} \]

Therefore, in order even for event \( E_1 \) to happen with approximately \( 1 - e^{-1} \) probability for small \( \rho \) values, the expected number of iterations is exponential to the reciprocal of the success probability \( \rho \) of the MonteCarlo-Prop function. For efficiency purposes it is necessary to have methods where \( n \) does not depend exponentially to \( 1/\rho \).

VI. BestNear Selection: Improving the Convergence Rate

The computationally efficient alternative to Naïve Random-Tree for finding a path, if one exists, without a steering function is an approach such as RRT-Extend, which is probabilistically complete. RRT-Connect, however, has been shown to converge almost surely to a suboptimal solution. While RRT* provides asymptotic optimality, it requires a steering function. Nevertheless, it is possible to consider methods which do not require a steering function and for which this work argues desirable \( \delta \)-robust completeness and near-optimality properties. This variation is referred to here as RRT-BestNear (RRTBN) and it is based on ideas in the related literature [27]. The RRT-BestNear works like Naïve Random-Tree but switches line 3 in Alg. 2 with the procedure in Alg. 3.

Algorithm 3: BestNear(\( X_f, V, \delta_{BN} \))

1. \( x_{rand} \leftarrow \text{Sample State}(X_f); \)
2. \( x_{near} \leftarrow \text{Near}(V, x_{rand}, \delta_{BN}); \)
3. If \( x_{near} = \emptyset \) return Nearest(V, xrand);
4. Else return \( \text{arg min}_{x \in X_{near}} c_{st}(x); \)

Relative to RRT-Extend, this method also uses a predefined metric to find states near a randomly sampled point \( x_{rand} \) (line 1). In contrast to RRT family’s selection procedure that only returns the closest vertex to the sampled point, BestNear looks at a neighborhood around the sampled state (a hyperball of radius \( \delta_{BN} \) as in line 2) and examines each vertex in that neighborhood. The procedure will select the vertex that corresponds to the lowest trajectory cost from \( x_0 \), the root of the tree (line 4: \( c_{st}(x) \) corresponds to the cost of the path from root \( x_0 \) to node \( x \)). If this neighborhood is empty, then BestNear defaults to using the nearest neighbor to the random sample as in RRT (line 3).

Relative to RRT*, this method also uses a neighborhood and tries to propagate a node along the best path from the root. Nevertheless, RRT* propagates the closest node to \( x_{rand} \). Then, it attempts connections between all nodes in an \( X_{near} \) set of the newly generated state and the edge to the parent with the best path cost from the root is kept. These steps require a
steering function. Here a near-optimal node in a neighborhood of the random sample is directly selected for propagation.

The BestNear procedure was presented as a “heuristic” in previous work to improve the performance of the basic RRT \cite{RRT}. Here it is formally analyzed to show its mathematical guarantees in terms of path quality and convergence properties.

A. Completeness Properties of RRT-BestNear

An important observation from the complexity discussion for Naive Random-Tree is that the exponential term comes from the harmonic series expression, which fundamentally comes from the selection mechanism of the algorithm.

![Fig. 3. (left) Illustration of different trajectories generated by MonteCarlo-Prop inside the covering balls B(\(x^*, \delta, T\)). Many trajectories may enter these balls, but may not be δ-similar to the optimal one. (right) Sampling \(x_{rand}\) in the gray region guarantees that a node \(z_i \in B_\delta(x_i)\) is selected for propagation so that either \(z_i = y_i\) or \(\text{cost}(z_i) < \text{cost}(y_i)\).

Consider the selection mechanism BestNear procedure for RRT-BestNear, as illustrated in Fig. 3.

Lemma 2: Assuming uniform sampling in the Sample function BestNear, if \(\exists x\) s.t. \(x \in B_{\delta_BN}(x^*_i)\) at iteration \(n\), then the probability that BestNear selects for propagation a node \(x' \in B_\delta(x^*_i)\) can be lower bounded by a positive constant \(\gamma\) for every \(n' > n\).

Proof: The random sample \(x_{rand}\) is sampled at the intersection of a small ball of radius \(\theta = \delta - \delta_{BN}\), and of a \(\delta_{BN}\)-radius ball centered at \(y_i\), which exists with probability \(P(x_{rand}^{(n)})\).

In other words, if \(x_{rand} \in B_{\theta}(y_i) \cap B_{\delta_{BN}}(y_i)\), then \(y_i\) will be considered by BestNear. The small circle is defined so that the \(\delta_{BN}\) ball of \(x_{rand}\) can only reach states in \(B_\delta(x_i)\). It is also required that \(x_{rand}\) is in the \(\delta_{BN}\)-radius ball centered at \(y_i\), so that at least one node in \(B_\delta(x_i)\) is guaranteed to be returned. Thus, the second probability can be lower bounded by the following expression:

\[
\gamma = \mu(\{x_{rand}|x_{rand} \in B_{\theta}(y_i) \cap B_{\delta_{BN}}(x')\})
\]

Now with Thm. 1 and Lem. 2 both of the probabilities of selection and propagation are lower bounded, it is possible to argue the following completeness property.

Theorem 7: If \(\delta_{BN} < \delta\), then RRT-BestNear is probabilistically \(\delta\)-robustly complete.

Proof: Let \(A_k^{(n)}\) denote the event that at the \(n^{th}\) iteration, the algorithm generates one \(\delta\)-similar trajectory \(\pi = x_{k-1}^{n} \cdot x_{k}\) such that \(\pi(0) \in B_\delta(x_{k-1}^{n})\) and \(\pi(T_{rand}) \in B_{\delta_{BN}}(x_k)\). Let \(E_k^{(n)}\) denote the event that from iteration 1 to \(n\), the algorithm generates at least one such trajectory. Then, the event \(\neg E_k^{(n)}\) is the event the algorithm fails to generate any near-optimal trajectory inside \(B_{\delta_{BN}}(x_k^*)\) after \(n\) iterations, which only happens when all \(n\) iterations fail, i.e.,

\[
P(\neg E_k^{(n)}) = P(\neg A_k^{(1)}) \cdot P(\neg A_k^{(2)}|A_k^{(1)}) \cdot \ldots \cdot P(\neg A_k^{(n)}|\bigcap_{j=1}^{n-1} \neg A_k^{(j)})
\]

(17)

The probability that \(\neg A_k^{(n)}\) happens given \(\bigcap_{j=1}^{n-1} \neg A_k^{(j)}\) is equivalent to the probability of failing to generating a trajectory to the \(B_{\delta_{BN}}(x_{k-1}^*)\) plus the probability that a trajectory has been generated to \(B_{\delta_{BN}}(x_k^*)\), but fails to generate a new trajectory segment to \(B_{\delta_{BN}}(x_k^*)\), i.e.,

\[
P(\neg A_k^{(n)}) \geq 1 - \prod_{j=1}^{n} P(\neg A_k^{(j)})
\]

(18)

Therefore, using Eq. (17) and Eq. (18)

\[
P(E_k^{(n)}) \geq 1 - \prod_{j=1}^{n} (1 - P(E_k^{(j)}) \cdot \gamma_{\delta_{BN}})
\]

(19)

For the base case, \(P(E_0^{(j)}) = 1\) because \(x_0\) is always in \(B_{\delta_{BN}}(x_0)\). Then, consider event \(E_1\) from iteration 1 to \(n\) using Eq. (19)

\[
P(E_1^{(n)}) \geq 1 - \prod_{j=1}^{n} (1 - \gamma_{\delta_{BN}}) = 1 - (1 - \gamma_{\delta_{BN}})^n \lim_{n \to \infty} P(E_1^{(n)}) \geq 1 - \lim_{n \to \infty} (1 - \gamma_{\delta_{BN}})^n = 1 - 0 = 1.
\]

For the induction step, if \(\lim_{n \to \infty} P(E_k^{(j)}) = 1\), we need to show that the same will be true for \(E_{k+1}^{(n)}\). Set \(y_k^{(n)} = \prod_{j=1}^{n} (1 - P(E_k^{(j)}) \cdot \gamma_{\delta_{BN}})\). The logarithm of \(y_k^{(n)}\) behaves as follows,

\[
\log y_k^{(n)} = \log \prod_{j=1}^{n} (1 - P(E_{k-1}^{(j)})) \cdot \gamma_{\delta_{BN}}
\]

\[
= \sum_{j=1}^{n} \log(1 - P(E_{k-1}^{(j)})) \cdot \gamma_{\delta_{BN}}
\]

\[
< \sum_{j=1}^{n} -P(E_{k-1}^{(j)}) \cdot \gamma_{\delta_{BN}}
\]

\[
= -\gamma_{\delta_{BN}} \cdot \sum_{j=1}^{n} P(E_{k-1}^{(j)})
\]

(20)

From the inductive assumption that, \(P(E_{k}^{(j)})\) converges to 1 as \(j \to \infty\), then \(\lim_{n \to \infty} \sum_{j=1}^{n} P(E_{k}^{(j)}) = \infty\). Then,

\[
\lim_{n \to \infty} \log y_{k+1}^{(n)} < -\gamma_{\delta_{BN}}\cdot \lim_{n \to \infty} \sum_{j=1}^{n} P(E_{k}^{(j)}) = -\infty
\]

\[
\iff \lim_{n \to \infty} y_{k+1}^{(n)} = 0
\]

Using Eq. (19), with \(\lim_{n \to \infty} y_{k+1}^{(n)} = 0\), it can be shown that:

\[
\lim_{n \to \infty} P(E_{k+1}^{(n)}) = 1 - \lim_{n \to \infty} y_{k+1}^{(n)} = 1 - 0 = 1.
\]
B. Optimality Properties of RRT-BestNear

The proof of asymptotic $\delta$-robust near-optimality follows directly from Thm. 7 the Lipschitz continuity, additivity, and monotonicity of the cost function (Assumption 3). The completeness proof is already examining the generation of a near optimal trajectory, but the bound on the costs needs to be calculated. 

**Theorem 8:** If $\delta_{BN} < \delta$, then RRT-BestNear is asymptotically $\delta$-robustly near-optimal.

**Proof:** Let $x_{i-1}^r, x_i$ denote the $\delta$-similar trajectory segment generated by RRT-BestNear where $x_{i-1}^r \in B_\delta(x_{i-1}^*)$ of the optimal path and $x_i \in B_{\delta_{BN}}(x_i^*)$. Thm. 1 guarantees that the probability of generating it by MonteCarlo-Prop can be lower bounded as $\rho_\delta \rightarrow \delta_{BN}$. Then from the definition of $\delta$-similar trajectories and Lipschitz continuity of the cost function,

$$\text{cost}(x_{i-1}^r \rightarrow x_i) \leq \text{cost}(x_{i-1}^* \rightarrow x_i^*) + K_x \cdot \delta \quad (21)$$

Lem. 4 guarantees that when $x_i$ exists in $B_{\delta_{BN}}(x_i^*)$, then $x_i^r$ returned by the BestNear function with least bound $\gamma$, must have equal or less cost, i.e $x_i^r$ can be the same state as $x_i$ or a different state with smaller or equal cost:

$$\text{cost}(x_i^r) \leq \text{cost}(x_i) \quad (22)$$

Consider $B_\delta(x_i^*)$, as illustrated in Fig. 3 according to (21) and (22),

$$\text{cost}(x_0 \rightarrow x_1^*) \leq \text{cost}(x_0 \rightarrow x_1) \leq \text{cost}(x_0 \rightarrow x_1^*) + K_x \cdot \delta$$

Assume this is true for $k$ segments:

$$\text{cost}(x_0 \rightarrow x_k^*) \leq \text{cost}(x_0 \rightarrow x_k^*) + k \cdot K_x \cdot \delta$$

Consider the cost of the trajectory with $k+1$ segments:

$$\text{cost}(x_0 \rightarrow x_{k+1}^*) \leq \text{cost}(x_0 \rightarrow x_{k+1}) + K_x \cdot \delta$$

By induction, this holds for all $k$.

Since the largest $k = \frac{T^*}{K_x}$, and the cost of the trajectory is its duration,

$$\text{cost}(x_0 \rightarrow x_k^*) \leq (1 + \frac{K_x \cdot \delta}{T}) \cdot c_k^*$$

Recall from Theorem 7 event $E_k$ implies event $\{Y^{RRT_{BN}} \leq (1 + \alpha \delta)c_k^*\}$.

$$\mathbb{P}(E_k) = \mathbb{P}(\{Y^{SST} \leq (1 + \frac{K_x \delta}{T})c_k^*\})$$

As $n \rightarrow \infty$, Thm. 11 guarantees that if $\delta > \delta_n + 2\delta_s$, $E_k(\infty)$ almost surely happens.

$$\mathbb{P}\left(\{\limsup_{n \rightarrow \infty} Y^{SST}_n \leq (1 + \frac{K_x \delta}{T})c_k^*\}\right) = \lim_{n \rightarrow \infty} \mathbb{P}(E_k(n)) = 1 \quad \blacksquare$$

C. Expected Iterations of RRT-BestNear

Now consider the convergence rate for RRT-BestNear.

**Theorem 9:** For the worst case, the $k^{th}$ segments of the trajectory returned by RRT-BestNear converges linearly to the near optimal solution, e.g.,

$$\lim_{n \rightarrow \infty} \frac{|\mathbb{P}(E_k(n+1)) - 1|}{|\mathbb{P}(E_k(n)) - 1|} = (1 - \gamma \rho_\delta \rightarrow \delta_{BN}) \in (0,1)$$

**Proof:** Applying the boundary condition of Eq. 19 consider the ratio of the probabilities between iteration $k+1$ and $k$.

$$\frac{|\mathbb{P}(E_k(n+1)) - 1|}{|\mathbb{P}(E_k(n)) - 1|} = \prod_{j=1}^{n+1} (1 - \mathbb{P}(E_k^{(j)}) \cdot \gamma \rho_\delta \rightarrow \delta_{BN})$$

Taking $\lim_{n \rightarrow \infty}$, and given Thm. 7 such that

$$\lim_{n \rightarrow \infty} \frac{|\mathbb{P}(E_k(n+1)) - 1|}{|\mathbb{P}(E_k(n)) - 1|} = \lim_{n \rightarrow \infty} (1 - \mathbb{P}(E_k(n)) \cdot \gamma \rho_\delta \rightarrow \delta_{BN})$$

$$= 1 - \gamma \rho_\delta \rightarrow \delta_{BN} \in (0,1)$$

Thm. 9 states that RRT-BestNear approach converges linearly to near optimal solution. Recall that Thm. 6 indicates that Naive Random-Tree approach converges logarithmically (sub-linearly). This difference indicates that RRT-BestNear converges significantly faster than Naive Random-Tree.

Now consider the expected iteration, i.e. the iterations needed to return a near-optimal trajectory with a certain probability. Specifically, the convergence rate depends on the difficulty level of the kinodynamic planning problem which is measured by the probability $\rho_\delta \rightarrow \delta_{BN}$ of successfully generating a $\delta$-similar trajectory segment connecting two covering balls.

**Theorem 10:** For a $\delta$-robust optimal trajectory consisting of $k > 0$ segments, and a fixed $\rho_\delta \rightarrow \delta_{BN} > 0$, the iterations $N_{\rho_\delta \rightarrow \delta_{BN}}$ for the algorithm to generate a near-optimal solution with probability greater than $1 - e^{-1}$ can be bounded by:

$$1 - e^{-1} \cdot \gamma \rho_\delta \rightarrow \delta_{BN}$$

The above theorem argues that in order to achieve at least $1 - e^{-1} \approx 63.21\%$ probability for RRT-BestNear to generate a near-optimal trajectory, the needed iterations can be upper bounded. This bound is in the same order as the expected number of iterations for RRT to return a solution [19].
Proof: Firstly, consider $B_{δBN}(x'_1)$, and from Eq.(4),
\[
\mathbb{P}(E_{1}^{(n)}) \geq 1 - \prod_{i=1}^{n} (1 - (1 - \gamma \rho_{δ \to δBN})^{\rho_{δ \to δBN}}) = 1 - (1 - (\gamma \rho_{δ \to δBN})^{n})
\]
Clearly, if $n \geq \frac{1}{\gamma \rho_{δ \to δBN}}$, then $\mathbb{P}(E_{1}^{(n)}) > 1 - e^{-1}$.
Next, consider $B_{δBN}(x'_{k+1})$, and use $\mathbb{P}(E_{k}^{(n)}) > 1 - e^{-1}$ as the induction assumption, from Eq. (4),
\[
\mathbb{P}(E_{k+1}^{(n)}) \geq 1 - \prod_{i=1}^{n} (1 - (1 - \gamma \rho_{δ \to δBN})^{\rho_{δ \to δBN}}) = 1 - (1 - (\gamma \rho_{δ \to δBN})^{n})
\]
Similar to the reasoning for $\mathbb{P}(E_{1}^{(n)})$, if $\Delta n \geq \frac{1}{\gamma \rho_{δ \to δBN}}$, then $\mathbb{P}(E_{k+1}^{(n)}) > 1 - e^{-1}$, where $\Delta n = n - \frac{1}{\gamma \rho_{δ \to δBN}}$. Inductively, there is, if $n \geq \frac{1}{\gamma \rho_{δ \to δBN}}$, then $\mathbb{P}(E_{k+1}^{(n)}) > 1 - e^{-1}$.
Recall that the expected number of iterations for $E_{1}$ to succeed for Naïve Random-Tree was $n \approx e^{\gamma} \cdot e^{(\rho_{δ}^{-1})}$. While in the case of RRT-BestNear for event $E_{1}$, this bound of iterations is $\frac{1}{1 - e^{\gamma}} \cdot \frac{\rho_{δ}}{\rho_{δ \to δBN}}$. Clearly, this is a significant improvement already for event $E_{1}$ (though providing a weaker near-optimality guarantee). For the cases of $E_{k}$, ($k > 1$), the iteration bound for RRT-BestNear linearly depends on the length of the optimal trajectory. While for Naïve Random-Tree, it is already intractable even for the first ball.
On the other hand, in terms of “per iteration” computation time, RRT-BestNear is worse than RRT. The BestNear procedure requires a $δ_{BN}$-radius query operation which is computationally more expensive than the nearest neighbor query in RRT. Therefore, RRT-BestNear shall be increasingly slower than RRT. Nevertheless, the following section considers a variation for keeping the completeness, optimality, and convergence properties, while also achieving considerable improvement on the expected time complexity.

VII. HARD CORE POINT PROCESS: FURTHER IMPROVING THE TIME COMPLEXITY

This section discusses STABLE SPARSE-RRT(SST) which upgrades RRT-BestNear with a Matérn Type III point process[7].

Last section shows that employing BestNear to replace the naïve selection mechanism can significantly improve convergence rate, at the price of relaxing completeness and optimality to $δ$-robust counterparts. From previous discussions, it is clear that the relaxed completeness and optimality do not require keeping every state. Not all states are necessary to be kept for Thm. 7 to hold, since some states with high cost has little chance of being selected by BestNear. Therefore, it is possible to further reduce the time complexity and space requirement.

Algorithm 4: SST($\mathcal{X} \cup \{x_0\}, \tau_{prop}, N, δ_{s}, δ_{c}$)

1. $i \leftarrow 0$, $\mathcal{V}_{active} \leftarrow \{x_0\}$, $\mathcal{V}_{inactive} \leftarrow \emptyset$
2. $V \leftarrow \mathcal{V}_{active} \cup \mathcal{V}_{inactive}$, $E \leftarrow \emptyset$, $G \leftarrow \{V, E\}$
3. $s_0 \leftarrow x_0$, $s_0.rep \leftarrow x_0$, $S \leftarrow \{s_0\}$
4. while $i + + < N$ do
5. $s_{sample} \leftarrow \text{Sample}(\mathcal{X})$
6. $x_{nearest} \leftarrow \text{BestNear}(\mathcal{V}_{active}, s_{sample}, δ_{s})$
7. $x_{new} \leftarrow \text{MonteCarlo-Prop}(x_{nearest}, U, τ_{prop})$
8. if CollisionFree($x_{nearest} \rightarrow x_{new}$) then
9. $s_{new} \leftarrow \text{Nearest}(S, x_{new})$
10. if $\text{dist}(x_{new}, s_{new}) > δ_{s}$ then
11. $S \leftarrow S \cup \{x_{new}\}$
12. $s_{new} \leftarrow x_{new}$
13. $s_{new}.rep \leftarrow \text{NULL}$
14. $x_{peer} \leftarrow s_{new}.rep$
15. if $x_{peer} == \text{NULL}$ or $\text{cost}(x_{new}) < \text{cost}(x_{peer})$ then
16. $\mathcal{V}_{active} \leftarrow \mathcal{V}_{active} \setminus \{x_{peer}\}$
17. $\mathcal{V}_{inactive} \leftarrow \mathcal{V}_{inactive} \cup \{x_{peer}\}$
18. $s_{new}.rep \leftarrow x_{new}$
19. $\mathcal{V}_{active} \leftarrow \mathcal{V}_{active} \cup \{x_{new}\}$
20. $E \leftarrow E \cup \{x_{nearest} \rightarrow x_{new}\}$
21. while IsLeaf ($x_{peer}$) and $x_{peer} \in \mathcal{V}_{inactive}$ do
22. $x_{parent} \leftarrow \text{Parent}(x_{peer})$
23. $E \leftarrow E \setminus \{x_{parent} \rightarrow x_{peer}\}$
24. $\mathcal{V}_{inactive} \leftarrow \mathcal{V}_{inactive} \setminus \{x_{peer}\}$
25. $x_{peer} \leftarrow x_{parent}$
26. return $G$.

A. Properties of SST

As discussion in the last section, RRT-BestNear has relaxed completeness. SST is the algorithm that upgrades RRT-BestNear with a thinning process. This section argues that the introduction of the thinning process does not compromise the probabilistically $δ$-robustly completeness of the algorithm.

Consider the selection mechanism of SST.

Lemma 3: Let $δ_{c} = δ - δ_{s} - δ_{c} > 0$. If a state $x \in \mathcal{V}_{active}$ is generated at iteration $n$ s.t. $x \in \mathcal{B}_{δ_{s}}(x'_{n})$, then for every iteration $n' \geq n$, there is a state $x' \in \mathcal{V}_{active}$ so that $x' \in \mathcal{B}_{δ_{c}}(x'_{n'})$ and $\text{cost}(x') \leq \text{cost}(x)$.

Proof: Given $x$ is a node generated by SST, then it is guaranteed that a witness point $s$ is located near $x$. As illustrated in Fig. 5, the witness point $s$ can be located, in the worst case, at distance $δ_{s}$ away from the boundary of $\mathcal{B}_{δ_{s}}(x'_{n})$. However, if $x \neq x_{0}$, note that $x$ can be removed from $\mathcal{V}_{active}$ by SST in later iterations. In fact, $x$ almost surely will be removed, if $x \neq x_{0}$.
Next, consider the size of the intersection of $B_{\delta-\delta_v}(x^*_v)$ and a ball of radius $\delta_v$ that is entirely enclosed in $B_\delta(x^*_v)$. Let $x_v$ denote the center of this ball. This intersection, highlighted in Fig. 5B, represents the area that a sample can be generated so as to return a state from ball $B_{\delta-\delta_v}(x^*_v)$. In the worst case the center this ball $B_{\delta-\delta_v}(x^*_v)$ could be on the border of $B_{\delta-\delta_v}(x^*_v)$ as seen in Fig. 5B. Then, the probability of sampling a state in this region can be computed as:

$$\gamma = \inf \mathbb{P}\left( x' \text{ returned by BestNear : } x' \in B_{\delta}(x^*_v) \right) = \frac{\mu(B_{\delta-\delta_v}(x^*_v) \cap B_\delta(x^*_v))}{\mu(B_\delta(x^*))}.$$  

This is the smallest region that will guarantee selection of a node in $B_\delta(x^*_v)$.

Lemma 4 shows that the probability to selection a near optimal state within the covering ball sequence with a non-increasing cost can be lower bounded. It is almost identical to the selection mechanism of RRT-BestNear. Together with Thm. 1, The completeness of SST can be argued.

**Theorem 11:** If $\delta_v + 2\delta_s < \delta$, then STABLE SPARSE-RRT is probabilistically $\delta$-robustly complete. e.g.,

$$\liminf_{n \to \infty} \mathbb{P}( \exists \pi \in \Pi^{SST}_n : \pi \text{ solution to } (X_f, x_0, X_G, \delta) ) = 1$$

Next, asymptotically $\delta$-robustly near-optimality can also be argued for SST.

**Theorem 12:** If $\delta_v + 2\delta_s < \delta$, then STABLE SPARSE-RRT is asymptotically $\delta$-robustly near-optimal. e.g.

$$\mathbb{P}\left( \limsup_{n \to \infty} Y^{SST}_n \leq (1 + \frac{K}{T}) \cdot c^* \right) = 1$$

The proofs for Thm. 11 and Thm. 12 are almost identical to the proof of Thm. 7 and Thm. 8 respectively. And they are omitted here.

**B. Complexity of RRT-BestNear/RRT-Extend/SST**

The convergence rate and expected iterations for SST, again, are almost identical to that of RRT-BestNear. Briefly speaking, similar to RRT-BestNear, since both of the selection mechanism and propagation probability of SST can be bounded by constants. Proof of convergence rate and expected iterations for SST follows the same reasoning as of RRT-BestNear.

**Theorem 13:** For the worst case, the $k^{th}$ segments of the trajectory returned by SST converges linearly to the near
optimal solution. e.g.,
\[
\lim_{n \to \infty} \frac{|P(E^{(n+1)}_k) - 1|}{|P(E^{(n)}_k) - 1|} = (1 - \gamma \rho_\delta \to \delta_c) \in (0, 1)
\]

Theorem 14: For a \( \delta \)-robust optimal trajectory consisting of \( k > 0 \) segments, and a fixed \( \rho_\delta \to \delta_c > 0 \), the iterations \( N_{\rho_\delta \to \delta_c} \) for the algorithm to generate a near-optimal solution with probability greater than \( 1 - e^{-1} \) can be bounded by:
\[
\frac{1}{1-e^{-1}} \cdot \frac{k}{\gamma \rho_\delta \to \delta_c}
\]
The benefit of SST is that the per iteration complexity ends up being smaller than RRT-BestNear. It worths mentioning that the most expensive operations for the family of algorithms that discussed in this paper asymptotically is the near neighbor queries. Therefore, SST delivers noticeable improvement over RRT-BestNear. The rest of this section examines the influence of the sparse data structure which is brought by the “hard-core” point process in SST.

In contrast to RRT* which employs a steering function, the proposed algorithm involves no such functions. All operations for the proposed algorithm are well understood. Therefore, it is possible to evaluate the overall computational cost needed for RRT-BestNear and SST.

Lemma 5: Among a set of size \( n \) points, the average time complexity for the nearest neighbor query is \( O(\log n) \). The average time complexity of the range query for near neighbors is \( O(n) \), since the result is a fixed proportional subset of the whole set.

Lemma 8 estimates the optimal time complexity of two types of near neighbors queries that is used in RRT-BestNear and SST. Note that performance of implementations usually deteriorates as dimension or size of the set increases.

Now it is possible to estimate the overall asymptotical time complexities for RRT-BestNear, RRT-Extend, and SST to return near-optimal solutions with probability at least \( 1 - e^{-1} \approx 63.21\% \).

Lemma 6: For a \( k \) segment optimal trajectory with \( \delta \) clearance, the expected running time for RRT-BestNear to return a near-optimal solution with \( 1 - e^{-1} \) probability can be evaluated as, \( O(\frac{k}{\gamma \rho_\delta \to \delta_c}) \)

Proof: Let \( N_p \) denote \( \frac{k}{(1-e^{-1})\gamma \rho_\delta \to \delta_c} \). Since Lemma 5 and Lemma 3 the total time computation after \( N_p \) iterations can be evaluated as,
\[
O\left( \sum_{i=1}^{N_p} c \cdot \delta^{-d} + N_p \right) = O\left( \frac{k}{\gamma \rho_\delta \to \delta_c} \right)
\]

Note that the second term \( N_p \) describes the worst case of deletion of nodes in \( V_{active} \) corresponding to Lines 21-25 of Alg. 4. For \( N_p \) iterations, in the worst case, the algorithm can delete at most \( N_p \) nodes.

In sum, RRT-BestNear and SST have the same forms of completeness and optimality guarantees, but SST has significantly better time complexity than RRT-BestNear, linear v.s. square, due to the hard-core distribution in \( S \) and the sparse distribution in the state space. This is verified by the experimental results in the experiment section. It is also important to know that RRT-BestNear and SST heavily depend on STLMA assumption, which guarantees that \( \rho_\delta \to \delta_c > 0 \) around the optimal trajectory. RRT-Extend, on the other hand, is a relatively “imbalanced” algorithm. It can guarantee “strongest” completeness, but non-optimality at all due to its greediness. And for this reason, it is hard to design a “star” version of RRT-Extend. Next section discusses how to build the “star” version of RRT-BestNear and SST. It is also interesting to note that, RRT-Extend has slightly worse time complexity than SST in theory, but experimentally it is not obvious in the early stage of the execution.

VIII. RESTORING ASYMPTOTIC OPTIMALITY

The probabilistical \( \delta \)-robust completeness and near-optimality are relaxed concepts of probabilistic complete and asymptotical optimality. Strickly speaking, RRT-BestNear and SST cannot solve motion planning problem as stand-alone
algorithms. They can only solve motion planning problems of minimum clearance greater than or equals to \( \delta \). Never the less, they are strong forms of mathematical guarantees, which only quantitatively differ from the original concepts. And most importantly, they do not require solving BVP problems. This section discusses designing BVP solution independent sampling-based algorithms, employing \( \delta \)-robust counterparts as building blocks, to achieve uncompromised probabilistical completeness and asymptotical optimality.

A. From SST to SST\(^*\)

**Algorithm 5:** SST* \((X, U, x_0, T_{prop}, \delta_s, v, 0)\)

1. \(j \leftarrow 0; K \leftarrow k_0;\)
2. \(\delta_s \leftarrow \delta_{s,0}; \delta_v \leftarrow \delta_{v,0};\)
3. **while** true **do**
   4. \(SST(X, U, x_0, T_{prop}, \delta_v, \delta_s);\)
   5. \(\delta_s \leftarrow \xi \cdot \delta_s; \delta_v \leftarrow \xi \cdot \delta_v; j \leftarrow j + 1;\)
   6. \(K \leftarrow (1 + \log j) \cdot \xi^{-\delta_w} \cdot k_0;\)

Recall that in RRT-BestNear and SST, for a given \( \delta \) and \( \delta_{BN} \) value, \( \gamma \) and \( \delta_{BN} \) are two constants describing the probability of selecting a near optimal state for propagation and of successfully propagating to the next ball region (the smaller core regions). First of all, the rates of changing with respect to the chosen clearance values are needed.

**Theorem 15:** Given \( \delta > 0 \), and \( \delta_{BN} > 0 \), for a scale \( \alpha \in (0, 1) \), let \( \delta' = \alpha \delta \) and \( \delta'_{BN} = \alpha \delta_{BN} \), there is

\[
\frac{\gamma'}{\gamma} = \alpha^d
\]

**Theorem 16:** For a \( B_1 \) of radius \( \delta \) and a ball \( B'_1 \) with radius \( \delta' \), such that \( \delta'/\delta = \alpha \), where \( \alpha \in (0, 1) \), there is

\[
\frac{\delta'}{\delta} = \alpha^{w+1}
\]

The proofs for Thm. 15 and 16 can be found in Section XII. Then it is possible to show that RRT\(_{BN}^* \) and SST\(^*\) are probabilistically complete and asymptotically optimal.

**Theorem 17:** SST\(^*\) is probabilistically complete, i.e.,
\[
\lim_{j \to \infty} \mathbb{P}(\{\exists x_{goal} \in (V_n^{SST^*} \cap X_G)\}) = 1
\]

**Proof:** Let \( E^{(i)}_{k,j} (k \geq 1) \) denote the event \( E_k \) at sprint \( j \), after \( i \) iterations within the sub-function SST. There is,

\[
P(E_{k,j}) = 1 - \prod_{i=1}^{K(j)} (1 - \mathbb{P}(E^{(i)}_{k-1,j} \gamma(j) \rho(j)))
\]

Let \( c \) be a constant \( 1 \leq c \leq K(j) \) and \( p_c \) be the value of \( \mathbb{P}(E^{(c)}_{k-1,j}) \) for a given \( j \). Note that within the same sprint \( j \), Eq. 23 equivalents to Eq. 19. Let \( P_c \) be a constant such that

\[
P_c = \prod_{i=1}^{c} (1 - \mathbb{P}(E^{(i)}_{k-1,j} \gamma(j) \rho(j)))
\]

Then, Eq. 23 becomes the following,

\[
\mathbb{P}(E_{k,j}) = 1 - P_c \prod_{i=c+1}^{K(j)} (1 - \mathbb{P}(E^{(i)}_{k-1,j} \gamma(j) \rho(j))
\]

And clearly, any \( \mathbb{P}(E^{(i)}_{k-1,j}) (k \geq 0) \) is strictly positive and non-decreasing as \( i \) increases. Then Eq. 24 becomes the following,

\[
P(E_{k,j}) = 1 - P_c \prod_{i=c+1}^{K(j)} (1 - \mathbb{P}(E^{(i)}_{k-1,j} \gamma(j) \rho(j)))
\]

Since the inequality \((1 - \frac{2}{x})^x < e^{-\alpha}\) for all \( x > 1 \) and \( \alpha > 0 \). Then Eq. 25 becomes,

\[
P(E_{k,j}) \geq 1 - P_c \left[ (1 - \mathbb{P}(\gamma(j) \rho(j))) \right] \prod_{i=c+1}^{K(j)} (1 - \mathbb{P}(E^{(i)}_{k-1,j} \gamma(j) \rho(j)))
\]

Since Thm. 15 and let \( \beta = k_0 \cdot \gamma(0) \rho(0) \), Eq. 26 becomes,

\[
P(E_{k,j}) > 1 - P_c (e^{-\beta})^{(i) \gamma(j) \rho(j)} \prod_{i=c+1}^{K(j)} (1 - \mathbb{P}(E^{(i)}_{k-1,j} \gamma(j) \rho(j)))
\]

Clearly, as \( j \) increases to infinity, the following holds,

\[
\lim_{j \to \infty} P(E_{k,j}) = \lim_{j \to \infty} 1 - P_c (e^{-\beta})^{(i) \gamma(j) \rho(j)}
\]

Since the limit exists, therefore it is true that,

\[
\lim_{j \to \infty} P(E_{k,j}) = 1
\]

Next, the argument regarding asymptotic optimality.

**Theorem 18:** SST\(^*\) is asymptotically optimal, i.e.,

\[
\mathbb{P}(\{\limsup_{j \to \infty} Y_j^{SST^*} = c^*\}) = 1
\]

**Proof:** Since event \( E_{k,j} \) implies event \( \{Y_n^{SST^*} \leq (1 + c_0 \delta) \cdot c^*\} \), therefore at the end of the \( j^{th} \) sprint, since Eq. 27
\[ \mathbb{P}(\{ Y_j^{\text{SST}} \leq (1 + c_n \delta(j)) \cdot c^* \}) = \mathbb{P}(E_{k,j}) \]
\[ > 1 - \frac{P_c \cdot (e^{-p_c \cdot \beta} (1 + \log j))}{e^{\gamma(n) \rho(j) \cdot c}} \]

As \( j \to \infty \), clearly \( \lim_{j \to \infty} \delta(j) = 0 \), \( \lim_{j \to \infty} \gamma(j) = 0 \), and \( \lim_{j \to \infty} \rho(j) = 0 \). Then, there is,
\[ \mathbb{P}(\{ \lim_{j \to \infty} Y_j^{\text{SST}} \leq \lim_{j \to \infty} (1 + c_n \delta(j)) \cdot c^* \}) = \mathbb{P}(\{ \lim_{j \to \infty} Y_j^{\text{SST}} \leq c^* \}) > \lim_{j \to \infty} \left[ 1 - \frac{P_c \cdot (e^{-p_c \cdot \beta (1 + \log j))}{e^{\gamma(n) \rho(j) \cdot c}} \right] = 1 - 0 = 1 \]

\section*{B. Comparing SST* to RRT*}

Comparing SST* and RRT* provides useful insights of sampling-based algorithms.

Alg. 5 describes a process that gradually relaxing the “sparsification”, (in fact, RRT-BestNear unintentionally employs the “sparsification” idea, without explicitly removing states from the tree), which increasingly adds active states. On a high level perspective, SST* employs the same idea implicitly. Recall that RRT* also allows adding states as the algorithm proceeds. The difference is that RRT* also adds one state per iteration, while SST* adds a batch of states per batch iterations. Generally speaking, all sampling-based algorithms need increasingly adding states to cover the space.

With this approach, sampling-based algorithms avoid knowing the minimum clearance parameter \( \delta \).

Briefly speaking, PRM* (RRT* is achieved by trimming edges from PRM*), is a process of generating random vertices (and connecting them with steering function in the state space while ensuring the states are “sufficiently connected”. “Sufficiently connected” is modelled and analysed by the theory of random geometric graph. In another word, a “finite percolated graph” is increasingly built and it eventually becomes an infinite percolated graph, which contains the optimal path.

On the other hand, SST* follows a complete different but simpler process. The data structure is always a tree, meaning that, at any moment there are \( n \) edges and \( n + 1 \) vertices. The system accessibility property (Ball-Box theorem) guarantees that it is possible to extend horizontal edges from one ball region to the next. And the “infinite monky theory” guarantees that it shall happen eventually and almost surely. The Lipschitz continuity of cost function guarantees a near-optimal bound on the extended trajectories. BestNear and hard-core mechanisms simply make the above guarantees efficient.

It worths mentioning that Alg. 5 is probabilistically complete and asymptotically optimal. However, compared to RRT*, this version of SST* provides a slightly weaker form of guarantee. Recall that the completeness guarantee for RRT* is
\[ \mathbb{P}(\lim_{j \to \infty} A_n) = 1 \]
\[ \text{where } A_n \text{ in } \text{[14]} \text{ denotes the event of generating a near-optimal trajectory which is collectively covered by a sequence of balls, ) meaning that RRT* guarantees to deliver a near-optimal trajectory in sufficiently large finite time. While this completeness guarantee for the particular version of SST* shown above is not as strong as RRT*. Thm. [7] shows that \( \lim \inf_{j \to \infty} \mathbb{P}(E_{k,j}) = 1 \). Although this difference is minimal practically, it is interesting to investigate the theoretical properties between the two algorithms.

**Corollary 2**: For small \( k_0 \), to almost surely return a near-optimal trajectory by Alg. 5 only happens at time infinity. e.g., \( \lim \inf_{j \to \infty} \mathbb{P}(E_{k,j}) = 1 \). If \( k_0 \) is chosen sufficiently large, Alg. 5 has guarantee to return a near-optimal trajectory in sufficiently large finite time. e.g., \( \mathbb{P}(\lim \inf_{j \to \infty} E_{k,j}) = 1 \).

**Proof**: Let \( \neg E_{k,j} \) be the negation of \( E_{k,j} \). Then since Eq. 27 there is,
\[ \mathbb{P}(\neg E_{k,j}) < 1 - 1 + P_c \cdot (e^{-p_c \cdot \beta (1 + \log j)) - \gamma(j) \cdot \rho(j) \cdot c} \]
\[ = P_c \cdot (e^{-p_c \cdot \beta} \cdot (e^{-p_c \cdot \beta \cdot \log j} \cdot (e^{-p_c \cdot \gamma(j) \cdot \rho(j)}) \] (28)

Clearly, \( p_c \cdot c > 0 \), so \( 0 < e^{-p_c \cdot c} < 1 \). And \( \gamma(j) \rho(j) \to 0 \) as \( j \to \infty \). Therefore for the third term in Eq. 28 there is \( (e^{-p_c \cdot \gamma(j) \cdot \rho(j)} < 1 \). Eq. 28 becomes the following,
\[ \mathbb{P}(\neg E_{k,j}) < P_c \cdot (e^{-p_c \cdot \beta} \cdot (e^{-p_c \cdot \beta \cdot \log j}) \]

To sum up the probabilities of \( j \) from \( 1 \to \infty \), there is
\[ \sum_{j=1}^{\infty} \mathbb{P}(\neg E_{k,j}) < P_c \cdot (e^{-p_c \cdot \beta}) \sum_{j=1}^{\infty} (e^{-p_c \cdot \beta \cdot \log j}) \]
\[ = P_c \cdot (e^{-p_c \cdot \beta}) \sum_{j=1}^{\infty} \frac{1}{(j) \cdot \log j \cdot \rho(j) \cdot \beta} \] (29)

Clearly when \( p_c \cdot \beta > 1 \) Eq. 29 is Riemann zeta function times a constant. It converges when the real part \( p_c \cdot \beta > 1 \) and diverges otherwise. Recall that \( \beta = k_0 \gamma(0) \rho(0) \), since \( \gamma(0) \rho(0) \) is unknown, \( p_c \cdot \beta \) is unknown.

Therefore, when \( k_0 \) is chosen sufficiently large, Eq. 29 converges \( \sum_{j=1}^{\infty} \mathbb{P}(\neg E_{k,j}) < \infty \). According to Borel-Cantelli lemma, there is
\[ \mathbb{P}\left( \lim \sup_{j \to \infty} \neg E_{k,j} \right) = 0 \iff \mathbb{P}\left( \lim \inf_{j \to \infty} E_{k,j} \right) = 1 \]
It implies that for some large number \( j_c \), \( \mathbb{P}(E_{k,j}) = 1 \), for all \( j \geq j_c \). Therefore, for all finite \( k > 0 \), there is
\[ \mathbb{P}\left( \lim \inf_{j \to \infty} E_{k,j} \right) = 1 \]

In fact, the framework of SST* is rather flexible. By tuning the expression of \( K(j) \), the guarantee can be as strong as RRT* at the cost of more computation. Consider a new version of Alg. 5 where switch line 6 to be
\[ K \leftarrow j \cdot \xi^{-(d+w+1)} \cdot k_0 \]
Corollary 3: The version of $SST^*$ where $K \leftarrow j \cdot \xi^{-(d+w+1)} \cdot k_0$ guarantees to return a near-optimal trajectory in sufficiently large finite time, e.g.
\[
\mathbb{P} \left( \lim_{j \to \infty} E_{k,j} \right) = 1
\]

The proof is rather simple and very similar to the proof of Corollary 2. The difference is that Eq. 29 becomes
\[
\sum_{j=1}^{\infty} \mathbb{P}(\neg E_{k,j}) < P_c \cdot \sum_{j=1}^{\infty} (e^{-p_c \beta})^j \tag{30}
\]
It is a geometric series instead of Riemann zeta function. For all possible $k_0 > 0$, Eq. 30 converges.

IX. EXPERIMENTAL EVALUATION

In order to evaluate the proposed method, a set of experiments involving several different systems have been conducted. The proposed algorithm, $SST$, is compared against $RRT$ as a baseline and also with another common algorithm: (a) if a steering function is available, a comparison with $RRT^*$ is conducted, (b) if $RRT^*$ cannot be used, a comparison with a heuristic alternative based on a “shooting” function is utilized [12]. The shooting function is numerically approximating a steering function but doesn’t connect two states exactly. To alleviate this problem, when a rewire is performed, entire subtrees are resimulated with the new end state that is close to the original state. The overall results show that $SST$ can provide consistently improving path quality given more iterations as $RRT^*$ does for kinematic systems, achieve running time equivalent (if not better) than $RRT$, and maintain a small number of nodes, all while using a very simple random propagation primitive.

Evaluation was also conducted on several pendulum-like systems which include a single link pendulum, a two-link passive-active acrobot system, and a cart-pole system. In addition, a quadrotor system is considered. These systems have simple state update equations, but are nonlinear. No steering function is used in these experiments.

One of the more interesting applications of $SST$ is in the domain of planning for physically-simulated systems [11]. $SST$ is able to provide improving path quality given enough time and keeps the number of forward propagations to one per iteration. In this setup, the computational cost of propagation overtakes the cost of nearest neighbor queries. Nearest neighbor queries become the bottleneck in problems like the kinematic point where propagation and collision checking are cheap. In this respect, $SST$ is specially suited to plan for physically-simulated systems. In Fig. 6 the different experimental setups. From left to right and top to bottom, a kinematic point system, a simple pendulum, a cart-pole among obstacles, a passive-active acrobot, a 12-dim quadrotor, and physically-simulated car-like system. Each experiment is averaged over 50 runs of each algorithm. A. Quality of Solution Trajectories

In Fig. 7 the average solution quality to nodes in each tree is shown. This average is a measure of the quality of trajectories that have explored the space being searched. In every case, $SST$ is able to improve quality over time, even in the case of the physically-simulated car. $RRT$ will increases this average over time because it chooses suboptimal nodes and further propagates them.

B. Time Efficiency

Fig. 9 shows time vs. iterations plots for each of the systems. The graphs show the amount of time it took to achieve a number of iterations. The running time of $SST$ is always comparable or better than $RRT$. $RRT^*$ has a higher time cost per iteration as expected. Initially $SST$ is slightly slower than $RRT$ for the kinematic point, but becomes increasingly more efficient later on. This is explained by Lem. 9 and Lem. 7 since $SST$ has better running time than $RRT$ given the sparse data structure. For physically-simulated systems, the computational cost is dominated by the forward propagation, where both $RRT$ and $SST$ perform the same amount.
Fig. 7. The average cost to each node in the tree for each algorithm (RRT, RRT$^*$ or the shooting approach, and SST).

Fig. 8. The average cost to each node in the tree for each algorithm (RRT, RRT$^*$ or the shooting approach, and SST).

C. Space Efficiency

One of the major gains of using SST is in the smaller number of nodes that are needed in the data structure. Fig. [1] shows the number of nodes stored by each of the algorithms. The number of nodes is significantly lower in SST, even when also considering the witness set S. The sparse data structure of SST makes the memory requirements quite small, in contrast to RRT and RRT$^*$ which don’t perform any pruning operations. In the case using shooting, sometimes the inaccuracy of shooting will cause collisions to occur in resimulated trees, pruning them from the tree. This can lead to losing solution trajectories however.

X. DISCUSSION AND CONCLUSION

Recently, the focus in sampling-based motion planning has been on providing optimality guarantees. Achieving this objective for systems with dynamics has generally required specialized steering functions. SST is a method that does not require such processes and still provides trajectory quality guarantee. Experiments and analysis also find that the running time and space requirements of SST are better even than RRT, which can quickly provide feasible trajectories.

By removing the requirement of the steering function, SST is well suited to research in other areas where steering functions are difficult to construct. One of these areas is planning under uncertainty, where planning is performed in belief space. It is impossible to find a steering function in this domain, but forward propagation can update the probability distributions. It is also important to evaluate the effectiveness of the approach on real systems with significant dynamics, high-dimensional state spaces, and in cluttered spaces.

REFERENCES

[1] Bullet Physics Engine. http://bulletphysics.org.
[2] M. Branicky, S. LaValle, K. Olson, and L. Yang. Quasi-Randomized Path Planning. In ICRA, pages 1481–1487, Seoul, Korea, May 2001.
[3] T. Brooks, R. Lozano-Perez. A subdivision algorithm in configuration space for findpath with rotation. In International Joint Conference on Artificial intelligence, 1983.
[4] J. Canny. The Complexity of Robot Motion Planning. The MIT Press, 1988.
[5] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor,
Fig. 9. The amount of time needed for each algorithm (RRT, RRT* or the shooting approach, and SST).

Fig. 10. The amount of time needed for each algorithm (RRT, RRT* or the shooting approach, and SST).

W. Burgard, L. E. Kavraki, and S. Thrun. *Principles of Robot Motion*. The MIT Press, 2005.

[6] A. Dobson and K.E. Bekris. Sparse Roadmap Spanners for Asymptotically Near-Optimal Motion Planning, 2013.

[7] D. Ferguson and A. Stentz. Anytime RRTs. In *IROS*, pages 5369–5375, October 2006.

[8] E. Frazzoli, M. A. Dahleh, and E. Feron. Real-Time Motion Planning for Agile Autonomous Vehicles. *Journal of Guidance, Control and Dynamics*, 25(1):116–129, 2002.

[9] G. Grimmett and D. Stirzaker. *Probability and Random Processes*. Oxford University Press, third edition, 2001.

[10] D. Hsu, R. Kindel, J. C. Latombe, and S. Rock. Randomized Kinodynamic Motion Planning with Moving Obstacles. *IJRR*, 21(3):233–255, March 2002.

[11] Lucas Janson and Marco Pavone. Fast Marching Tree: a Fast Marching Sampling-Based Method for Optimal Motion Planning in Many Dimensions. http://arxiv.org/pdf/1306.3532.pdf, (submitted: 30 April 2014).

[12] J.-H. Jeon, S. Karaman, and E. Frazzoli. Anytime Computation of Time-Optimal Off-Road Vehicle Maneuvers using the RRT*. In *CDC*, 2011.

[13] J.-H. Jeon, R.V. Cowlagi, S.C. Peters, S. Karaman, E. Frazzoli, P. Tsiotras, and K Iagnemma. Optimal Motion Planning with the Half-Car Dynamical Model for Autonomous High-Speed Driving. In *ACC*, 2013.

[14] S. Karaman and E. Frazzoli. Sampling-based Algorithms for Optimal Motion Planning. *IJRR*, 30(7):846–894, June 2011.

[15] S. Karaman and E. Frazzoli. Sampling-Based Optimal Motion Planning for Non-holonomic Dynamical Systems. In *ICRA*, 2013.

[16] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars. Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces. *IEEE TRA*, 12(4):566–580, 1996.

[17] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *International Journal of Robotics Research*, 5(1):90-98,1986.

[18] J. Kuffner and S. LaValle. An efficient approach to single-query path planning. In *ICRA*, 2000.

[19] S. M. LaValle and J. J. Kuffner. Randomized Kinody-
Fig. 11. Number of nodes in the tree (RRT, RRT* or the shooting approach, and SST).

[20] Y. Li and K. E. Bekris. Learning Approximate Cost-to-Go Metrics To Improve Sampling-based Motion Planning. In IEEE Intl. Conf. on Robotics and Automation (ICRA), Shanghai, China, 9-13 May 2011.

[21] Z. Littlefield, Y. Li, and K.E. Bekris. Efficient Sampling-based Motion Planning with Asymptotic Near-Optimality Guarantees with Dynamics. In IROS, 2013.

[22] J.Levine M. S. Branicky, M. M. Curtis and S. Morgan. Sampling-based planning, control, and verification of hybrid systems. In IEEE Proc. Control Theory and Applications. 153(5):575-590, Sept. 2006.

[23] J. J. Kuffner M. Zucker and M. S. Branicky. Multiple RRTs for Rapid Replanning in Dynamic Environments. In IEEE Conference on Robotics and Automation, 2007.

[24] G. Papadopoulos, H Kurniawati, and N.M. Patrikalakis. Analysis of Asymptotically Optimal Sampling-based Motion Planning Algorithms for Lipschitz Continuous Dynamical Systems. http://arxiv.org/abs/1405.2872, (submitted: 12 May 2014).

[25] A. Perez, R. Platt, G. Konidaris, L. Kaelbling, and T. Lozano-Perez. LQR-RRT*: Optimal Sampling-based Motion Planning with Automatically Derived Extension Heuristic. In IEEE Intl. Conf. on Robotics and Automation (ICRA), 2012.

[26] J. Reif. Complexity of the mover’s problem and generalizations. In the IEEE Symposium on Foundations of Computer Science, 1979.

[27] C. Urmson and R. Simmons. Approaches for Heuristically Biasing RRT Growth. In IROS, pages 1178–1183, 2003.

[28] D. Webb and J. van Den Berg. Kinodynamic RRT*: Asymptotically Optimal Motion Planning for Robots with Linear Differential Constraints. In ICRA, 2013.