Asymptotically Near-Optimal Methods for Kinodynamic Planning with Initial State Uncertainty

Kaiwen Liu, Yang Zhang, Andrew Dobson, and Dmitry Berenson

***Abstract*—This paper focuses on the problem of planning robust trajectories for system with initial state uncertainty. While asymptotically-optimal methods have been proposed for many motion planning applications, there is no prior method which is able to guarantee asymptotic (near-)optimality for planning with initial state uncertainty with non-trivial dynamics and no steering function. In this paper we define a cost function to evaluate state divergence for kinodynamic planning. We prove properties of this function, our system dynamics, and our planners, which allow asymptotically near-optimal planning without a steering function. We then evaluate our two proposed planners, one that uses random restarts, and another that encourages sparsity, in several experiments. Our results suggest that we are able to improve both the trajectory and end state divergence by about half as compared to a previous method, which is not asymptotically near-optimal.**

***Index Terms*—Motion and Path Planning, Nonholonomic Motion Planning**

I. INTRODUCTION

CONSIDER a robot which is capable of precise control, but is not localized well with respect to the environment. This is a common situation for mobile manipulators because robot arm control is precise, but the localization errors of the base with respect to obstacles can be significant. The localization error at the initial state can lead to a large divergence in the possible final states of the system after executing a trajectory (see Fig. 1), which can easily lead to task failure. However, in some scenarios, such as hill climbing [1] or robot arm pushing operations [2], many actions are inherently uncertainty-decreasing. To plan trajectories that are robust to initial state uncertainty we wish to bias a planner to choose such uncertainty-reducing actions when possible.

These kinds of uncertainty-reducing actions have been considered in contraction analysis [3], which proves global exponential convergence over a contraction region. The contraction region is a subset of the state space where all states converge to a single trajectory with a series of controls. Based on contraction analysis, three divergence metrics were proposed along with motion planners based on RRT [4]. The metrics and planners were able to decrease the uncertainty and find low-divergence plans. However, [4] does not show that planning with these divergence metrics is efficient, probabilistically complete, or asymptotically optimal.

Manuscript received: September, 10, 2018; Revised December, 04, 2018; Accepted February, 09, 2019.

This paper was recommended for publication by Nancy Amato upon evaluation of the Associate Editor and Reviewers' comments. This work was supported in part by NSF grant IIS-1750489 and ONR grant N000141712050.

The authors are with University of Michigan, Ann Arbor, MI 48105, USA. kwliu@umich.edu; youngzh@umich.edu; chupleskai@gmail.com; berenson@eecs.umich.edu

Digital Object Identifier (DOI): see top of this page.

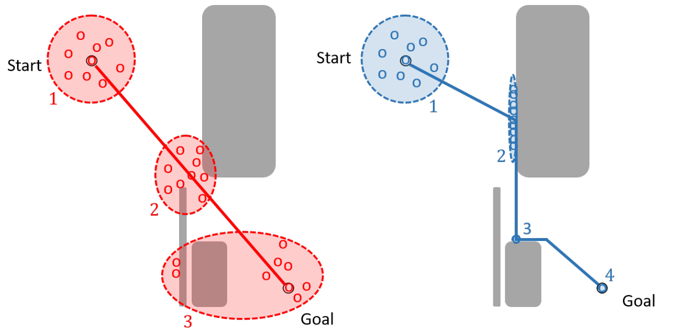

Fig. 1: Illustration of the effect of initial state uncertainty for a point robot that can slide on obstacles. Left: Moving directly to the goal leads to a large divergence at the final state. Right: There may be some actions that decrease uncertainty and by executing these actions we can reduce uncertainty at the final state.

RRT can be used for kinodynamic systems because it does not rely on a steering function. However, standard RRT does not guarantee asymptotic optimality and it reaches a sub-optimal solution almost surely. One method to empirically improve RRT's performance is to restart the planner multiple times [5] within the given time limit. RRT* guarantees asymptotic optimality but requires a steering function when applied to kinodynamic problems [6]. Steering functions can be obtained for some system dynamics but for many systems, such functions are not available and linearizations are only valid locally. Stable Sparse RRT (SST) is a sampling-based motion planner with a sparse data structure [7]. By pruning the tree and maintaining a small number of nodes, the nearest neighbor queries can be performed much more efficiently, which leads to an improvement in both space efficiency and time efficiency. SST has been shown to be probabilistic complete and asymptotically near-optimal [7].

Thus, inspired by [4], we define a cost metric based on contraction analysis [3] to evaluate system divergence. We show properties of this function and our two types of system dynamics which allow us to guarantee asymptotically near-optimal planning. We then use the cost function in multi-restart RRT and SST and show that these methods are far more efficient at reducing trajectory divergence than using RRT (as is used in [4]) in experiments on a mobile robot climbing a hill, a 2D point and 3D gripper that can slide in contact. The central contribution of this paper is thus planning methods for systems with initial state uncertainty that are proven to be asymptotically near-optimal and outperform previous work in terms of computation time and plan quality. The key contribution of our proof is an extension of the proof for SST [7] to systems where initial state uncertainty is represented as a set of particles. Despite only planning for the mean particle, we can nevertheless show that the cost (which considers all particles) of the trajectory produced by our planner is less than a constant multiple of optimal.

II. RELATED WORK

This paper contributes to the field of planning with uncertainty. In the general case, planning under uncertainty can be framed as a Markov Decision Process (MDP) or Partially Observable Markov Decision Process (POMDP) [8] and belief-space planning methods can be applied [9], [10], [11]. Although efficient POMDP solvers with optimality guarantees have been investigated [12], [13], POMDP solvers have difficulty when the state and action spaces are continuous and high-dimensional.

Another type of approach solves this problem by maximizing the probability of success given expected uncertainties [14], [15]. The trajectories produced by these approaches exhibit robustness to uncertainty, however explicit noise models and failure checking mechanism are required, which we do not assume are available.

The most similar method to ours is Convergent Planning [4], which uses cost functions based on contraction analysis [3] to avoid actions which cause divergence in the distribution of states. It biases action selection for Kinodynamic RRT [16] to improve generated trajectories' robustness to initial state uncertainty. However, the method provides no completeness or optimality guarantees. We compare to this kind of approach in our experiments.

We seek to make a more efficient planner, as well as showing completeness and optimality properties. Improving the efficiency of kinodynamic planning has been studied extensively, including minimizing calls to integration/physics engines [17], [18], [19] and reducing the number of vertices stored [20], [21]. We build on Stable Sparse RRT (SST) [7], which provides a proof for probabilistic completeness and asymptotic near-optimality as well as faster convergence to high-quality paths. Asymptotically-optimal planners based on RRT* [22] also exist. However, RRT* requires a steering function to rewire the tree to improve solution quality, which may not always be available for kinodynamic planning.

III. PROBLEM STATEMENT AND NOTATION

Consider a time-invariant system with derivative

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

where $\mathbb{X}$ is the state space and $\mathbb{U}$ is the control space. Both $\mathbb{X}$ and $\mathbb{U}$ are assumed to be compact, bounded sets, with $\mathbb{X}^{free} \subset \mathbb{X}$ denoting the set of valid states of the system, and $\mathbb{X}^{inv} = \mathbb{X} \setminus \mathbb{X}^{free}$ is the set of invalid states (often referred to as a collision set). The system is simulated in discrete time, propagating for some small fixed timestep $\Delta t > 0$.

This work assumes that a configuration of the system q is not represented as a single vector, but rather is a collection of m particles $q = \{x_1, x_2, \ldots, x_m\}$. This work will assume state vectors $x_i \ \forall i \in [1, m-1]$ are generated from some initial state distribution (i.e. a localization estimate), and $\bar{x} = x_m = \frac{1}{m-1}\sum_{i=1}^{m-1} x_i$ is the m^{th} particle, which is initialized to the sample mean of all other particles and used as the *representative* for this configuration. $\bar{x}$ will be used during planning for nearest neighbors checks, though it is understood $\bar{x}$ may deviate from the sample mean when the system moves. We denote the $\bar{x}$ of a q as $q.\bar{x}$.

Each particle evolves according to the system dynamics. Define a trajectory $\pi(t)$ as a function $\pi(t) : [0, t_d] \to \mathbb{X}^m$ defined over a domain of duration t_d, and with a control function $\Upsilon(t) : [0, t_d] \to \mathbb{U}$ such that $\pi(t)$ is the result of applying $\Upsilon(t)$ from the initial state $\pi(0) = q_{init} = \{x_1(0), x_2(0), \ldots, x_m(0)\}$.

Furthermore, in this work, we consider two distance metrics: one considers only the representative state $\bar{x}$, $d(q, q') = \|q.\bar{x} - q'.\bar{x}\|$; and the other, with subscript f, considers the full state, $d_f(q, q') = \|q.\bar{x} - q'.\bar{x}\| + |D(q) - D(q')|$, where $D()$ is a measure of dispersion of all the particles and will be defined in Section IV. Then, given the system dynamics, an initial configuration q_{init}, and a goal region $\mathbb{X}_{goal}$, we seek to find a trajectory starting from q_{init} which brings the representative state to within $\mathbb{X}_{goal}$ while minimizing the cost function defined in Section IV.

IV. METHOD

A. Cost Metric

A *contraction region*, defined in contraction analysis [3], is a region in the state space such that any trajectory starting in that region will remain in the region and converge exponentially to a single trajectory. Being in a contraction region is ideal for systems with initial state uncertainty because, if the uncertainty is contained within the region, the system's state will be "funneled" to a single trajectory. However, these regions are difficult to compute in general. In order to find a contraction region and exploit its properties, [4] defines several metrics to quantify the divergence of the state and uses those metrics in a planner to reduce state uncertainty.

The expected divergence metric in [4], $D_{[4]}$, is defined as:

$$\begin{aligned} E[\|\delta x(t)\|] &= E[\|\delta x(t_0)\|] e^{\int_{t_0}^{t_f} D_{[4]}(x,u,t) d\tau} \\ D_{[4]}(x, u, t) &:= \frac{d}{dt} \ln E[\|\delta x(t)\|] \\ &\approx \frac{1}{\delta t} \ln \frac{\frac{1}{m-1}\sum_{i=0}^{m-1} \|x_i(t+\delta t) - \bar{x}(t+\delta t)\|}{\frac{1}{m-1}\sum_{i=0}^{m-1} \|x_i(t) - \bar{x}(t)\|} \end{aligned} \quad (2)$$

where δx represents a virtual displacement. If $D_{[4]}$ is negative, then the expected value of the virtual displacement around a given trajectory will decrease over time.

[4] uses $D_{[4]}$ as a cost function in an RRT. However, extending their planner to be asymptotically near-optimal requires changing the metric because $D_{[4]}$ is not Lipshitz continuous, additive, or non-degenerative (these properties are required for asymptotic near-optimality). [4] also proposes the function below to estimate trajectory quality:

$$E_e := e^{\int_0^t D_{[4]}(x,u,t) d\tau} \approx \frac{\frac{1}{m-1}\sum_{i=0}^{m-1} \|x_i(t) - \bar{x}(t)\|}{\frac{1}{m-1}\sum_{i=0}^{m-1} \|x_i(0) - \bar{x}(0)\|} \quad (3)$$

This function also cannot be used as a metric in our planner because it only considers divergence at the endpoints of the trajectory. Instead, we will modify it to obey the required properties while maintaining an estimate of convergence.

Since all trajectories start with the same initial particle distribution, the denominator in Eq. 3 is constant. Therefore a simplified metric can assess divergence at a configuration:

$$D_e(q) = \frac{1}{m-1}\sum_{i=0}^{m-1} \|q.x_i - q.\bar{x}\| \tag{4}$$

If a contraction region exists, by limiting $D_e(q)$ the planner could force all particles into a contraction region and subsequently converge to a single trajectory. We show examples of such convergence in Section VI. Note that assuming $q.\bar{x}$ is always the mean of the distribution can produce inaccuracy in estimating divergence. However, recomputing the mean at each q violates the required metric properties.

While $D_e(q)$ is straightforward, it misses a key issue that arises when obstacles are present. Obstacles can make $\mathbb{X}^{free}$ non-convex, so even if two particles are nearby, they can be separated by a thin obstacle, which can prevent subsequent convergence. To account for this, we introduce a penalty which accounts for obstacles:

$$D(q) = \frac{1}{m-1}\sum_{i=1}^{m-1}(1+\lambda_2\alpha(x_i,\bar{x}))||x_i(t)-\bar{x}(t)|| \tag{5}$$

where α is the proportion of the line between $\bar{x}(t)$ and $x_i(t)$ that is in collision and $\lambda_2 > 0$ is a weighting factor used to set the collision penalty. Thus this metric penalizes configurations where the particles are "split" by an obstacle. The difference between D_e and D is shown in Fig. 2.

Finally, we arrive at the cost function used in our planners:

$$cost(\pi) = \lambda_1(t_f - t_s) + \int_{t_s}^{t_f} D(\pi(t))dt \tag{6}$$

where $\lambda_1 > 0$ weighs duration vs. divergence and t_s and t_f are the start time and end time of π. Unlike Eq. 3 from [4], this metric considers dispersion along the entire trajectory, not only at the endpoints. The integral is implemented as a discrete sum with a small time step.

B. Convergent Planning Methods

In order to respect the dynamics of the system, our proposed methods are designed based on Kinodynamic RRT [16]. Convergent RRT (C-RRT), similar to the RRT used in [4], biases the action selection by our cost function, and we use it as a baseline for comparison. Convergent Multiple-restart RRT (C-MRRT) restarts C-RRT once a solution is found and records the best solution found so far. Finally, we incorporate a sparse data structure [7] into C-RRT to arrive at Convergent SST (C-SST).

Note that RRT* is not considered and compared to the proposed algorithms. We are assuming an explicit steering function is not available for the systems which forbids applying the rewire step of RRT*. Even if a steering function or a "shooting" function [23] can be found for the system, RRT* is still not suitable for our context. The basic assumption of RRT* is that if the cost of a new node is smaller than the cost of a node in the neighbourhood, rewiring to the new node will reduce the cost of the whole subtree starting from that

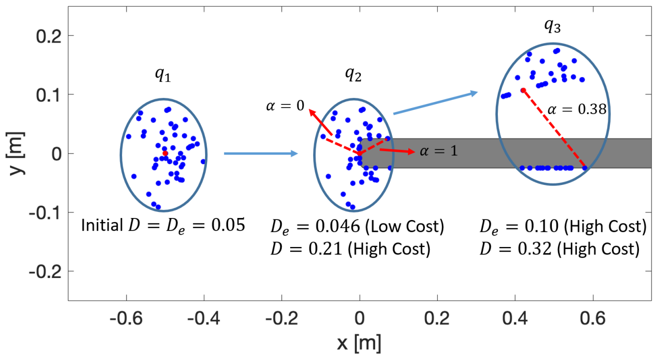

Fig. 2: Illustration of cost computation and differences between D_e and D ($\lambda_2 = 10$). The red particle represents $\bar{x}$. q_2 leads to a large divergence at q_3. Using D instead of D_e correctly labels q_2 to be high cost, and thus q_2 is likely to be avoided by the planner.

node. However, this assumption is not always true for our problem, because each node is represented by m particles and rewiring may change the positions of the particles. As a result, we need to re-evaluate the entire sub-tree when re-wiring in RRT*, which is extremely inefficient.

1) Convergent RRT (C-RRT): A random state is sampled at each iteration, and the nearest neighbor (using metric d) in C-RRT is found. `MonteCarlo-Prop` [7], which forward-simulates the system using a random piecewise-constant control for a random duration, is run H times from the nearest node to generate candidate actions. 50% of the time, the candidate whose endpoint is closest (using metric d) to the sampled point is selected. The other 50% of the time, the candidate with the lowest cost will be selected. The new node is then added to the C-RRT. New nodes will continue being added to C-RRT until the allowed iteration or time limit is reached. With probability P_{goal}, a $x \in \mathbb{X}_{goal}$ is sampled instead of a random state. By having the chance to select more convergent actions, C-RRT is able to improve the trajectory's robustness to initial state uncertainty.

2) Convergent Multiple-restart RRT (C-MRRT): Instead of continuing to grow the tree, C-MRRT restarts C-RRT once a solution is found and records the best solution according to our cost function. By restarting the tree after a solution is found, C-MRRT has the chance to quickly find alternate routes which the previous tree did not explore.

3) Convergent Stable Sparse RRT (C-SST): C-SST uses SST's [7] sparse data structure to speed up C-RRT and provide asymptotic near-optimality. The overall data structure consists of three sets of nodes: the witness set S, the active vertices set V_{active} and the inactive vertices set $V_{inactive}$. Including S allows for regions in the state space to have a representative node, regardless of which nodes are pruned. SST maintains the invariant that within a neighborhood of radius δ_s around any witness $s \in S$, there is always one state in V_{active}. This state in V_{active} is called the representative of the witness s, denoted $s.rep$, and is the node terminating the least-cost path to s found so far. Representatives can change over time but their costs from the root only decrease. V_{active} is the set of $s.rep$ nodes for all $s \in S$. $V_{inactive}$ consists of nodes that no longer lie on the best path from the initial node to the nearest $s.rep$ node but may provide connectivity to their child nodes in V_{active}. The union of V_{active} and $V_{inactive}$ is the set of tree nodes.

Algorithm 1 Convergent SST (C-SST)

```
 1: Initialize $V_{active}, V_{inactive}, S$ to empty
 2: Add $q_{init}$ to $V_{active}$ and create a representative in $S$
 3: while TimeRemaining() do
 4:     Sample $x \in \mathbb{X}$ (sample $x \in \mathbb{X}_{goal}$ with prob. $P_{goal}$)
        $q_{near} \leftarrow \texttt{BestNear}(V_{active}, x)$
        $q_{new} \leftarrow \texttt{MonteCarlo-Prop}(q_{near})$
 5:     Find nearest (using metric $d$) witness $s \in S$ to $q_{new}$
 6:     if $d(s, q_{new}) \leq \delta_s$ and $cost(q_{new}) < cost(s.rep)$ then
 7:         Add $q_{new}$ to $V_{active}$; move $s.rep$ to $V_{inactive}$
 8:         $s.rep = q_{new}$; Prune all leaf nodes in $V_{inactive}$
 9:     else if $d(s, q_{new}) > \delta_s$ then
10:         Add $q_{new}$ to $V_{active}$; create $s_{new}$ at $q_{new}.\bar{x}$
11:         $s_{new}.rep = q_{new}$
12:     end if
13: end while
```

To conduct a fair comparison between the planners, the propagation steps for all three use `MonteCarlo-Prop` [7]. Nearest neighbour search uses `BestNear` [7], which returns the lowest-cost node within a radius δ_{BN} around the sampled state if there exists a node within the radius, otherwise returning the nearest node to the sampled state.

V. Analysis of Systems, Costs, and Optimality

A δ-robustly feasible trajectory is one with strong δ-clearance from state space obstacles. Our proofs require this clearance for $q.\bar{x}$. We show that C-MRRT and C-SST provide near-optimality guarantees for the following problem.

Definition 1 (δ-robust feasible motion planning problem): Given the partition $\mathbb{X} = \mathbb{X}^{free} \cup \mathbb{X}^{inv}$ and query $(q_{init}, \mathbb{X}_{goal})$, a δ-robustly feasible motion planning problem is one which yields a δ-robust solution trajectory $\pi_\delta : [0, t_f] \to \mathbb{X}^{free,m}$ with $\pi_\delta(0) = q_{init}$ and $\pi_\delta(t_f).\bar{x} \in \mathbb{X}_{goal}$ for some $\delta > 0$.

This section first shows required properties of the proposed systems and cost metrics as prescribed by [7]. Each system is shown to be 1) Small-Time Locally Accessible (STLA) [24] and 2) Lipschitz continuous in state and control. We then show that our metric satisfies 1) Lipschitz continuity, 2) monotonicity $\big(cost(\pi_1) \leq cost([\pi_1, \pi_2])\big)$, 3) additivity $\big(cost([\pi_1, \pi_2]) = cost(\pi_1) + cost(\pi_2)\big)$, and 4) non-degenerativity $\big(t_2 - t_1 \leq M_c \cdot |cost(\pi_1) + cost(\pi_2)|\big)$. Then, the main contribution of the analysis is to show that C-MRRT and C-SST are asymptotically δ_f-robustly near-optimal by extending the proof in [7] to systems where initial state uncertainty is represented by particles. The key is to show that although the planner considers only distances between $q.\bar{x}$s, we can still bound the cost difference (which considers all particles) between a trajectory produced by the planners and the optimal.

A. Proving System Controllability and Continuity

This section proves the tested systems to be Small-Time Locally Controllable (STLC) [24] and Lipschitz continuous, which satisfies some of the required properties for showing asymptotic near-optimality. The systems are assumed to use Euler integration, and equations describe a single configuration, where each particle x is updated independently.

1) Kinematic Sliding Robot System: Consider a 2D or 3D kinematic rigid body model for this system, which translates but does not rotate, and can slide along obstacle surfaces. The 3D system has state $x = [x_x,\ x_y,\ x_z]$ and control $u = [u_x,\ u_y,\ u_z]$, with derivative $\dot{x}(x, u, \Delta t) = \Delta t \cdot u$. We assume the simulation enforces that each particle cannot move into invalid states, enforced by a projection operator $\dot{x}_{rect} = P(x, \dot{x})$, which is physically actualized as a manipulator system employing force-torque feedback control to maintain contact with and slide along surfaces rather than driving into them.

The analysis leverages two assumptions: first that the surfaces of obstacles are smooth (continuously differentiable), and second that the projector operator $P(x, \dot{x})$ always projects to states whose distance is within a bound of the contact point, and does not prevent "sliding", i.e. translation perpendicular to the contact normal.

a) STLC of Kinematic Sliding Robot: Given bounds $a, b, c > 0$ such that $u_x \in [-a,\ a]$, $u_y \in [-b,\ b]$, $u_z \in [-c,\ c]$, define control vectors $u_a = [a, 0, 0]$, $u_b = [0, b, 0]$, $u_c = [0, 0, c]$, which induce state delta vectors $\dot{x}_a = [\Delta t \cdot a, 0, 0]$, $\dot{x}_b = [0, \Delta t \cdot b, 0]$, $\dot{x}_c = [0, 0, \Delta t \cdot c]$. For any open subset around each particle x, the system can instantaneously follow any of $\dot{x}_a$, $\dot{x}_b$, and $\dot{x}_c$, which are linearly independent vectors forming a basis in $\mathbb{R}^3$, implying STLC.

b) Lipschitz Continuity of Kinematic Sliding Robot: In the absence of the projector $P(x, \dot{x})$, the system is linear, and each particle updates according to $\bar{x}' = \bar{x} + \Delta t \dot{x}_{rect}$. Substituting this update equation into the Lipschitz inequalities for control, we get

$$\|(x_0 + \Delta t \cdot u_0) - (x_0 + \Delta t \cdot u_1)\| \leq K_u \|u_0 - u_1\|$$

$$\Delta t \|u_0 - u_1\| \leq K_u \|u_0 - u_1\|$$

which holds for $K_u \geq \Delta t$. Doing the same for state yields

$$\|(x_0 + \Delta t \cdot u_0) - (x_1 + \Delta t \cdot u_0)\| \leq K_x \|x_0 - x_1\|$$

$$\|x_0 - x_1\| \leq K_x \|x_0 - x_1\|$$

which holds for $K_x \geq 1$. These relationships do not necessarily hold when, due to projection, $\dot{x}_{rect} \neq \dot{x}$, which is addressed by the assumptions above. For instance, the system violates Lipschitz continuity in control when the projector prevents "sliding" motion (this causes $\|u_0 - u_1\|$ to tend toward 0 while $\|f(x_0, u_0) - f(x_0, u_1)\|$ remains constant). Continuity in state and control are violated when the system propagates near sharp corners (if obstacles are not smooth).

2) Hill-Climbing Robot: This system represents a robot travelling over height-varying terrain with height $h(x_x,\ x_y)$, which has bounded first and second derivative ($\frac{\partial h}{\partial x} < \infty$ and $\frac{\partial^2 h}{\partial x^2} < \infty$). The system state vector $x = [x_x,\ x_y]$ and control vector $u = [u_v,\ u_\theta]$ yield an instantaneous height derivative

$$\Delta h(h, x, u) = \frac{\partial h(x_x, x_y)}{\partial x_x} \cos(u_\theta) + \frac{\partial h(x_x, x_y)}{\partial x_y} \sin(u_\theta)$$

This yields the state derivative function

$$\dot{x}(x, u, \Delta t) = \Delta t[u_v \cdot p \cdot \cos(u_\theta),\ u_v \cdot p \cdot \sin(u_\theta)]$$

where $p = -\frac{2}{\pi}\arctan(\Delta h(h, x, u)) + 1$.

a) STLC of Hill-Climber: Given bounds $u_v \in [0,\ a],\ u_\theta \in [-\pi, \pi]$, define control vectors $u_a = [1, 0]$ and $u_b = [1, \frac{\pi}{2}]$ which induce linearly independent state delta vectors $\dot{x}_a = [\Delta t \cdot p, 0]$ and $\dot{x}_b = [0, \Delta t \cdot p]$, forming a basis in $\mathbb{R}^2$, implying STLC of the system.

b) Lipschitz Continuity of Hill-Climber: We argue the Lipschitz continuity of the hill-climber follows from the boundedness of the partial derivatives of $f(x, u)$. Full derivations are omitted here for brevity, but are given as

$$\frac{\partial f}{\partial x_x} = \Big[1 + \Delta t \cdot u_v \cos(\theta)\frac{\partial p}{\partial x_x},\ \Delta t \cdot u_v \sin(\theta)\frac{\partial p}{\partial x_x}\Big]$$

$$\frac{\partial f}{\partial x_y} = \Big[\Delta t \cdot u_v \cos(\theta)\frac{\partial p}{\partial x_y},\ 1 + \Delta t \cdot u_v \sin(\theta)\frac{\partial p}{\partial x_y}\Big]$$

$$\frac{\partial f}{\partial u_v} = \Big[-\Delta t \cdot u_v \sin(\theta)\frac{\partial p}{\partial u_\theta},\ \Delta t \cdot u_v \cos(\theta)\frac{\partial p}{\partial u_\theta}\Big]$$

$$\frac{\partial f}{\partial u_\theta} = \Big[\Delta t \cdot p \cdot \cos(\theta),\ \Delta t \cdot p \cdot \sin(\theta)\Big], \text{ where}$$

$$\frac{\partial p}{\partial x_x} = \gamma \cdot \frac{\partial \Delta h}{\partial x_x},\ \frac{\partial p}{\partial x_y} = \gamma \cdot \frac{\partial \Delta h}{\partial x_y}$$

$$\frac{\partial p}{\partial u_\theta} = \gamma \cdot \frac{\partial \Delta h}{\partial u_\theta},\ \gamma = -\frac{2}{\pi}\left(\frac{1}{(\Delta h)^2 + 1}\right)$$

The magnitude of these derivatives is bounded over the whole domain (i.e. $\|\frac{\partial f}{\partial x_x}\|, \|\frac{\partial f}{\partial x_y}\|, \|\frac{\partial f}{\partial u_v}\|, \|\frac{\partial f}{\partial u_\theta}\| < \infty$), meaning the system exhibits Lipschitz continuity.

B. Proving Metric Properties

We now show the properties of the metric required for asymptotic optimality. Assume three non-degenerate trajectories $\pi_0 : [0, t_1] \rightarrow \mathbb{X}^{free,m}$, $\pi_1 : [0, t_1] \rightarrow \mathbb{X}^{free,m}$ and $\pi_2 : [t_1, t_2] \rightarrow \mathbb{X}^{free,m}$, where $t_1 > 0$, $t_2 > t_1$, $\pi_0(0) = \pi_1(0)$, and $\pi_1(t_1) = \pi_2(t_1)$.

1) Lipschitz Continuity of Metric: We will first extend the definition of Lipschitz continuity to this problem domain, and then show the proposed metric satisfies this constraint.

In prior work [7], the authors state that for Lipschitz continuity of a cost metric, the following must hold for two arbitrary trajectories π and π':

$$|cost(\pi) - cost(\pi')| \leq K_c \cdot \sup_{t \in [0, t_f]} \|\pi(t) - \pi'(t)\|$$

Intuitively, the cost difference between trajectories defined over the same time domain must be bounded by their maximum separation at time t. The proposed metric does not strictly satisfy this, since two trajectories can have zero separation (i.e. identical representative trajectories) but different distributions, resulting in cost difference $\Delta c > 0$.

Instead, we directly incorporate the effects of the particle distribution on the cost function into the continuity constraint, which is essentially using the d_f distance metric instead of the d distance metric:

$$|cost(\pi) - cost(\pi')| \leq K_c \cdot \big(\sup_{t \in [0, t_f]} \|\pi(t) - \pi'(t)\| + \sup_{t \in [0, t_f]} |D(\pi(t)) - D(\pi'(t))|\big) \tag{7}$$

Lipschitz continuity follows directly from the definition of the cost function.

2) Additivity of Metric: An additive cost function obeys $cost([\pi_1, \pi_2]) = cost(\pi_1) + cost(\pi_2)$. Thus, it must be $\lambda_1 t_2 + \int_0^{t_2} D([\pi_1, \pi_2](t))dt = \lambda_1 t_1 + \int_0^{t_1} D(\pi_1(t))dt + \lambda_1(t_2 - t_1) + \int_{t_1}^{t_2} D(\pi_2(t))dt$. All of the λ_1 terms above cancel out, leaving us with $\int_0^{t_2} D([\pi_1, \pi_2](t))dt = \int_0^{t_1} D(\pi_1(t))dt + \int_{t_1}^{t_2} D(\pi_2(t))dt$, which is true by the definition of an integral, and that $[\pi_1, \pi_2]$ is the concatenation of π_1 and π_2.

3) Monotonicity of Metric: A monotonic metric obeys $cost(\pi_1) \leq cost([\pi_1, \pi_2])$. Substituting the additivity property proven above, we only need show $cost(\pi_1) \leq cost(\pi_1) + cost(\pi_2)$, which is clearly true since $cost(\pi) \geq 0$.

4) Non-Degeneritivity of Metric: Non-degenerativity follows, when π_1, π_2 satisfy the given assumptions, and $cost(\pi_2) > 0$, which is trivially true since π_2 is defined over $[t_1, t_2]$ where $t_2 > t_1$, and thus $cost(\pi_2) \geq \lambda_1(t_2 - t_1) > 0$.

C. Asymptotically δ_f-robustly Near-optimality

We now show the asymptotic δ_f-robust near-optimality of C-MRRT and C-SST. Let $\mathcal{B}_\delta(x)$ be a state space ball centered at x with radius δ under metric d. $\mathcal{B}_\delta(q)$ is a configuration space ball centered at q with radius δ using metric d_f. We then define a measure of similarity for trajectories:

Definition 2 (δ-similar trajectory): Trajectories π, π' are δ-similar if for a continuous, non-decreasing scaling function $\sigma : [0, T_\pi] \rightarrow [0, T_{\pi'}]$, it is true that $\pi'(\sigma(t)) \in \mathcal{B}_\delta(\pi(t))$.

1) C-MRRT: The near-optimality of C-MRRT requires that the algorithm eventually generates a δ_f-similar trajectory to any optimal trajectory π^* with initial state q_0^*, where the subscript f refers to the distance metric d_f. Following from [7, Thm. 24], this can be shown if $q_0 \in \mathcal{B}_{\delta_{BN}}(q_0^*)$ (which is true since all trajectories start at the same initial configuration) and $\gamma_{\text{c-mrrt}}$ is nonzero. $\gamma_{\text{c-mrrt}}$ is the probability of selecting $q' \in \mathcal{B}_{\delta_f}(q_i^*)$ by `BestNear` given $\exists q \in \mathcal{B}_{\delta_{BN}}(q_i^*)$. Here, q_i^* is a configuration in π^*, and δ_{BN} is the radius used by `BestNear`.

[7, Lem. 23] shows that $\gamma_{\text{c-mrrt}} > 0$ when using a configuration space distance metric (in our context, the d_f metric). C-MRRT selects nodes with `BestNear` considering representative states $\bar{x}$ under metric d; however, since $d(q, q') = \|q.\bar{x} - q'.\bar{x}\| \leq d_f(q, q')$, if $q' \in \mathcal{B}_{\delta_{BN}}(q)$, meaning $d(q, q') \leq d_f(q, q') \leq \delta_{BN}$, then $q'.\bar{x} \in \mathcal{B}_{\delta_{BN}}(q.\bar{x})$. Leveraging [7, Lem. 23], it follows that $\gamma_{\text{c-mrrt}} > 0$.

Thus, C-MRRT eventually generates a δ_f-similar trajectory to any optimal trajectory, implying probabilistic δ_f-robust completeness [7, Def. 10]. Similar to [7, Thm. 25], we now prove C-MRRT is asymptotically δ_f-robustly near-optimal [7, Def. 12].

We consider a covering ball sequence [7, Def. 14] of radius δ_f around an optimal trajectory of cost C^* such that each segment between ball centers has cost C_Δ, and yields $\frac{C^*}{C_\Delta}$ segments. Let $\pi^* : [0, t_f] \rightarrow \mathbb{X}^{free,m}$ denote the optimal trajectory and $\overline{q_{i-1}^* \rightarrow q_i^*} : [t_{i-1}, t_i] \rightarrow \mathbb{X}^{free,m}$ denote the ith segment of π^*. As shown in Fig. 3, define $\overline{q_{i-1}' \rightarrow q_i}$ to be a segment of the δ_f-similar trajectory generated by the algorithm, where $q_{i-1}' \in \mathcal{B}_{\delta_f}(q_{i-1}^*)$ and $q_i \in \mathcal{B}_{\delta_{BN}}(q_i^*)$. By

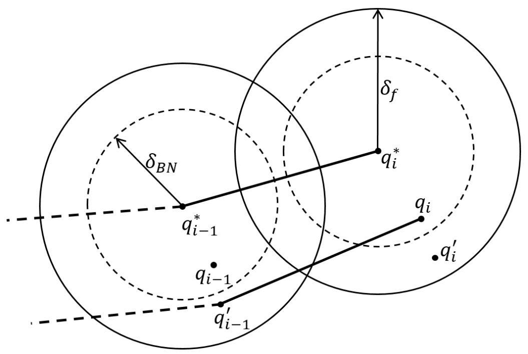

Fig. 3: Two hyper-balls of a covering ball sequence around the optimal trajectory. $\overline{q_{i-1}' \to q_i}$ is a δ_f-similar trajectory segment to the optimal trajectory segment. Nodes q_{i-1}' and q_i' are returned by `BestNear` with probability $\gamma_{\text{c-mrrt}}$, which could be the same as q_{i-1} and q_i.

using `MonteCarlo-Prop`, the probability of generating this segment is strictly non-zero [7, Thm. 17]. Then by definition of δ_f-similar trajectories and Eq. 7, the segment cost is bounded:

$$cost(\overline{q_{i-1}' \to q_i}) \leq cost(\overline{q_{i-1}^* \to q_i^*}) + K_c\delta_f \qquad (8)$$

Then let $q_i \in \mathcal{B}_{\delta_{BN}}(q_i^*)$ be a node in the tree and $q_i' \in \mathcal{B}_{\delta_f}(q_i^*)$ be a node returned by `BestNear`. [7, Lem. 23] shows that there is a positive probability $\gamma_{\text{c-mrrt}} > 0$ to sample a state x_{rand} such that $q_i' = \texttt{BestNear}(x_{rand})$, and the following holds: $cost(\overline{q_0 \to q_i'}) \leq cost(\overline{q_0 \to q_i})$.

We will now prove that the cost of a δ_f-similar trajectory is less than a constant multiple of the optimal cost by induction. Consider the first segment, according to Eq. 8, $cost(\overline{q_0 \to q_1'}) \leq cost(\overline{q_0 \to q_1^*}) + K_c\delta_f$. Assume the following is true for k segments,

$$cost(\overline{q_0 \to q_k'}) \leq cost(\overline{q_0 \to q_k^*}) + k \cdot K_c\delta_f$$

Then the cost of $k+1$ segments is,

$$\begin{aligned} cost(\overline{q_0 \to q_{k+1}'}) &\leq cost(\overline{q_0 \to q_{k+1}}) \\ &\leq cost(\overline{q_0 \to q_k'}) + cost(\overline{q_k' \to q_{k+1}}) \\ &\leq cost(\overline{q_0 \to q_{k+1}^*}) + (k+1)K_c\delta_f \end{aligned}$$

By induction, this holds for all k. Since the optimal path π^* has cost C^* and each segment has cost C_Δ, k will be at most $\frac{C^*}{C_\Delta}$. The cost bound for the entire trajectory is then

$$cost(\overline{q_0 \to q_k'}) \leq (1 + \frac{K_c\delta_f}{C_\Delta}) \cdot C^*$$

If our algorithm has returned a solution of a cost less than this bound, it must be the case that we have generated all of the k δ_f-similar segments over the optimal path up until this point and thus have also generated the final segment. We have previously argued that C-MRRT is able to generate a δ_f-similar trajectory to the optimal trajectory eventually. Hence, a δ_f-similar trajectory to the kth segment of π^* is almost surely generated as $n \to \infty$. Let $Y_n^{\text{C-MRRT}}$ represent the minimum cost among all trajectories generated by C-MRRT at iteration n. Then, asymptotic δ_f-robustly near-optimality [7, Def. 12] is guaranteed:

$$\mathbb{P}\Big(\{\lim_{n\to\infty} \sup_n \ Y_n^{\text{C-MRRT}} \leq (1 + \frac{K_c\delta_f}{C_\Delta}) \cdot C^*\}\Big) = 1$$

TABLE I: Experiment parameters

Experiment	λ_1	λ_2	δ_{BN}	δ_s	P_{goal}	H	m
Hill-Climbing Robot	0.1	-	0.4	0.05	0.02	10	51
2D Point Robot	0.1	1000	0.02	0.06	0.02	5	51
3D Gripper	0.1	1000	0.04	0.12	0.02	5	51

2) C-SST: Proving that C-SST is probabilistically δ_f-robustly complete is almost identical to the proof for C-MRRT, needing only to show that $\gamma_{\text{c-sst}}$ is nonzero. This follows by the definition of d_f. Given a constant radius δ, if a node exists in $\mathcal{B}_\delta(q_i^*)$, then its representative must exist in $\mathcal{B}_\delta(q_i^*.\bar{x})$. Then from [7, Lem. 27] and [7, Lem. 28], C-SST's pruning process does not affect completeness. The asymptotic δ_f-robustly near-optimality of C-SST can then be shown with the same proof as C-MRRT.

VI. Experiment and Results

We compare C-RRT, C-MRRT, and C-SST in three experiments. The first experiment is a mobile robot driving in hilly terrain. The second and third experiments are kinematic systems. The second one is a point robot in 2D space that can slide on obstacles, and the third one is a gripper in a 3D environment that can also slide. Parameters are shown in Table I. All experiments were run in C++ on an Intel Xeon CPU E5-2690 v4 @ 2.60GHz CPU and 251GB RAM.

A. Hill-Climbing Robot

For the hill-climbing robot, we model the hill as a height function $z = h(x, y) := 3y + \sin(x + xy)$ similar to [4] over $(x, y) \in [-1.5, 1, 5] \times [-1.5, 1.5]$. The robot's dynamics are described in Section V-A2. Intuitively, the robot's dynamics is correlated to the local gradient, and since different particles are at different locations with different gradients, there exists some actions in moving in certain directions that will actually reduce the dispersion of the particles.

The resulting solution paths of C-SST over different iterations are shown in Fig. 4(a) (where n refers to iterations). 30 trials are performed with this experiment, and the average solution cost over time is shown in Fig. 4(b). From Fig. 4(b), we can observe that C-SST is able to find a better solution with less time. C-RRT is able to improve the solution cost because we take the lowest cost trajectory that reaches the goal region at the given time. C-MRRT is able to improve the solution quality over time, but is slower than C-SST in finding a lower-cost solution path. Example solutions from C-RRT and C-SST are shown in Fig. 4(c)-(d). Note the difference in the divergence at the end configuration.

The results including nodes, solution costs, and final divergence of the solution are shown in Table II. Note that the nodes for C-MRRT is the number of nodes in the tree at the end of the run instead of all the nodes generated. Our cost requires that we not only want to have low divergence at the goal position, but also want to maintain low dispersion everywhere during the path. Thus, a low cost solution may not directly lead to a low divergence end configuration. In Table II, the final state divergence of each algorithm's solution are shown. With lower-cost solutions, the end point divergence on average goes down.

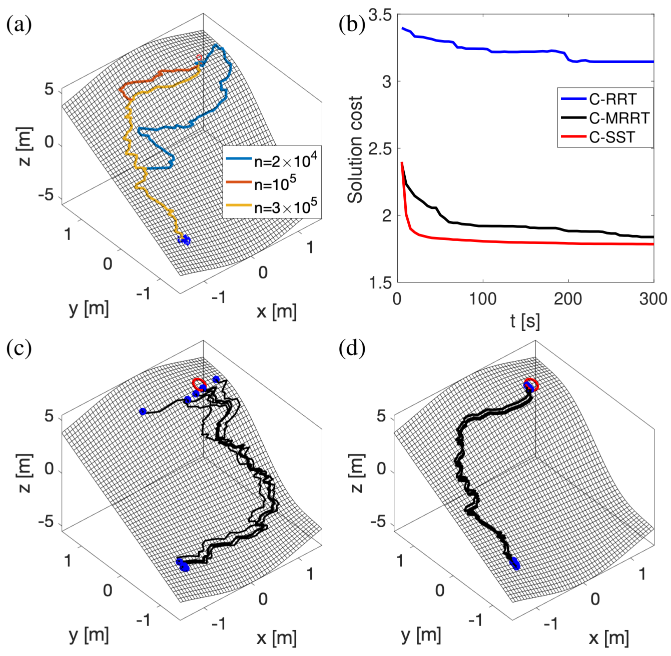

Fig. 4: *Top:* (a) Hill-climbing robot C-SST solution paths (blue dots are the initial particles) and (b) comparison results. *Bottom:* Two hill-climbing examples. (c) A C-RRT solution: cost is 5.16, end $D(q) = 0.58$, and 5% of rollouts reaching the goal. (d) A C-SST solution: cost is 1.79, end $D(q) = 0.09$, all rollouts reach the goal.

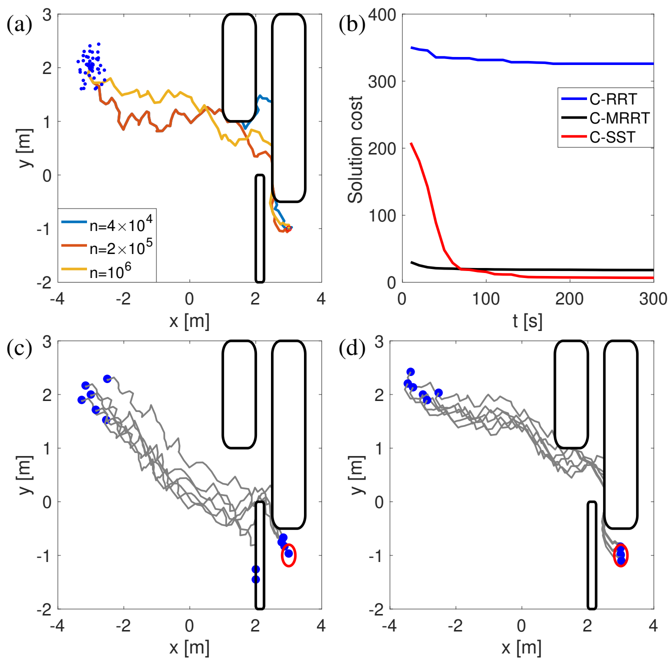

Fig. 5: *Top:* (a) 2D point robot C-SST solution paths (blue dots are the initial particles) and (b) comparison results. *Bottom:* Two point robot examples. (c) A C-RRT solution: cost is 320.32, end $D(q) = 111.06$, 10% of rollouts reach the goal. (d) A C-SST solution: cost is 6.45, end $D(q) = 0.13$, all rollouts reach the goal.

TABLE II: Hill climbing results. Avg. over 30 trials, 300s each

Test	Time[s]	Nodes[10^3]	Solution Cost	End $D(q)$
C-RRT	5	8.35 ± 0.12	3.40 ± 0.75	0.20 ± 0.08
	150	151 ± 6	3.22 ± 0.75	0.17 ± 0.06
	300	265 ± 7	3.11 ± 0.75	0.16 ± 0.06
C-MRRT	5	0.22 ± 0.13	2.40 ± 0.24	0.14 ± 0.06
	150	0.30 ± 0.22	1.92 ± 0.11	0.10 ± 0.02
	300	0.29 ± 0.16	1.84 ± 0.10	0.10 ± 0.02
C-SST	5	1.86 ± 0.16	2.40 ± 0.56	0.15 ± 0.09
	150	3.57 ± 0.10	1.80 ± 0.19	0.09 ± 0.01
	300	3.78 ± 0.10	1.80 ± 0.18	0.09 ± 0.01

TABLE III: 2D point robot results. Avg. over 30 trials, 300s each

Test	Time[s]	Nodes[10^3]	Solution Cost	End $D(q)$
C-RRT	10	21.7 ± 0.5	350 ± 359	38.8 ± 49.1
	150	199 ± 5	328 ± 332	36.3 ± 44.6
	300	333 ± 6	328 ± 337	36.3 ± 44.5
C-MRRT	10	0.57 ± 0.46	30.0 ± 11.5	0.21 ± 0.05
	150	0.60 ± 0.47	18.8 ± 1.47	0.20 ± 0.05
	300	0.56 ± 0.39	18.2 ± 0.91	0.19 ± 0.06
C-SST	10	5.20 ± 0.13	208 ± 188	36.3 ± 44.0
	150	9.31 ± 0.22	7.80 ± 0.83	0.17 ± 0.06
	300	9.54 ± 0.20	6.65 ± 0.48	0.13 ± 0.06

We also see that solution paths can be jerky. This is an artifact of using a small step-size and not including smoothness in the cost function. Trajectories generated by our methods could be improved by post-processing, but that is not within the scope of this work.

B. 2D Point Robot with Sliding

Further, we consider a 2D point robot system that follows the dynamics equations described in Section V-A1. The 2D point robot setup is shown in Fig. 5(a). $\mathbb{X}$ is designed to be $[-4, 4] \times [-2, 3]$. The initial uncertainty radius is set as 0.5 m in this example. The thin obstacle on the lower end is specially design to cause splitting, and zero-friction sliding on the obstacle surfaces is allowed. This experiment is specially designed to check if our planner is able to avoid splitting given that we incorporate splitting penalties in our cost function. Also, by using a 2D point robot, the collision checking can be done very quickly.

The resulting solution paths of C-SST over different iterations are shown in Fig. 5(a). By moving along the boundary of the obstacles, all the particles are able to condense to one line to pass the narrow passage, which is expected. The solution path generated at 4×10^4 iterations has high cost due to particle splitting in the middle of the path (particles and representative state lie on different sides of the obstacle). Fig. 5(b) shows the cost over time plot for C-RRT, C-MRRT, and C-SST. The result shows that C-SST is not as good as C-MRRT at first. However, given some time for C-SST, it is able to find a better solution than C-MRRT. Example solutions from C-RRT and C-SST are shown in Fig. 5(c)-(d). Note the difference in the divergence at the end configuration.

Table III shows the number of nodes at the end of the trial, solution costs and end configuration divergence. The divergence of the initial state is around 0.25. However, the divergence of the end state of C-RRT solutions are much larger than this value, which conveys that C-RRT is not effective at avoiding splits. For C-MRRT and C-SST, the divergence of the end state is much smaller, meaning more convergent end states. The results show that C-MRRT is able to get a low cost solution quicker than C-SST, but C-SST can reach better solutions given some time.

C. 3D Gripper with Sliding

Similar to the 2D point robot experiment, we now moved to 3D with a 4-finger gripper. The environment is set up

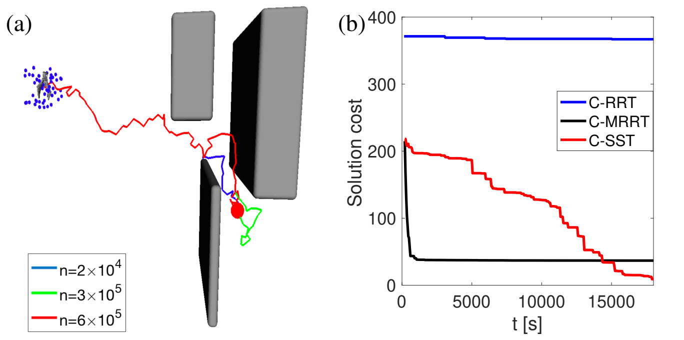

Fig. 6: 3D gripper C-SST solution trajectories and comparison results (blue dots are the initial particles).

in OpenRAVE shown in Fig. 6(a). Due to the "sliding" requirement of the system, the ODE collision checker is used for checking collisions and determining the sliding conditions, which requires significant computation time. The environment setup is also similar to the 2D case, where we have a narrow passage that the gripper needs to pass. This experiment is designed to test with a more realistic scenario, and compare the performance of the algorithms when the system propagation dominates the computation.

The low cost solutions condense all the particles into a single line before passing the narrow passage, and hence avoid splittings of the particles. The average solution cost of 30 trials over time is shown in Fig. 6(b). Since finding a solution in this experiment takes much longer than the other two experiments, we are able to take a closer look at the earlier behaviours of all methods. C-RRT is not able to improve the cost much over time. C-MRRT gets to a low cost solution in a very short amount of time, but it get stuck in this solution and is not able to find a better path. In contrast, C-SST is not able to improve the initial guess as quickly as C-MRRT, which may be because C-SST is replacing and pruning nodes in the space instead of restarting with another tree. However, with sufficient time, C-SST is able to find better solutions than C-MRRT, as in the previous experiment.

Combining the results from the experiments, we can get more insight to C-MRRT and C-SST. The main advantage of the sparse data structure in C-SST is fewer number of nodes which reduces the nearest neighbour querying time. However, C-MRRT generally has even less nodes than C-SST, and C-MRRT does not need to worry about pruning, which leads to much faster execution of C-MRRT than C-SST. This explains why C-MRRT can quickly improve the solution at the beginning. On the other hand, thanks to the pruning, improving and keeping the whole tree, C-SST shows more potential in finding the optimal solution. Although both C-MRRT and C-SST have been proved to be aymptotically δ_f-robustly near-optimal, the convergence rate is not known. From the experiments, C-MRRT improves quickly in the beginning, but after finding a relatively good solution, it is hard for C-MRRT to improve further, unlike C-SST.

VII. Conclusion

We propose near-optimal methods for efficient kinodynamic planning with uncertain initial states. We introduce a divergence cost metric to evaluate state uncertainty, which is used by planners to avoid divergent trajectories. We prove optimality and completeness for both the C-MRRT and C-SST planners and evaluate these planners vs. a baseline on three experiments. Our results suggest that these planners show a large improvement in efficiency and solution quality over the baseline. In future work we seek to find a method that locally improves path quality without a steering function to be used as a post-processing step.

References

[1] A. M. Johnson, M. T. Hale, G. C. Haynes, and D. E. Koditschek, “Autonomous legged hill and stairwell ascent,” in *SSRR*, Nov. 2011.

[2] M. C. Koval, J. E. King, N. S. Pollard, and S. S. Srinivasa, “Robust trajectory selection for rearrangement planning as a multi-armed bandit problem,” in *IROS*, Sept. 2015.

[3] W. Lohmiller and J.-J. E. Slotine, “On contraction analysis for non-linear systems,” *Automatica*, vol. 34, no. 6, pp. 683 – 696, 1998.

[4] A. M. Johnson, J. King, and S. Srinivasa, “Convergent planning,” *RAL*, vol. 1, no. 2, pp. 1044–1051, 2016.

[5] J. Luo and K. Hauser, “An empirical study of optimal motion planning,” in *IROS*, Sept. 2014.

[6] J. hwan 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*, June 2013.

[7] Y. Li, Z. Littlefield, and K. E. Bekris, “Asymptotically optimal sampling-based kinodynamic planning,” *IJRR*, vol. 35, no. 5, pp. 528–564, 2016.

[8] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, “Planning and acting in partially observable stochastic domains,” *AIJ*, vol. 101, no. 1-2, pp. 99–134, 1998.

[9] A. Bry and N. Roy, “Rapidly-exploring random belief trees for motion planning under uncertainty,” in *ICRA*, May 2011.

[10] A. Agha-mohammadi, S. Chakravorty, and N. M. Amato, “Firm: Sampling-based feedback motion-planning under motion uncertainty and imperfect measurements,” *IJRR*, vol. 33, no. 2, pp. 268–304, 2014.

[11] C. Phillips-Grafflin and D. Berenson, “Planning and resilient execution of policies for manipulation in contact with actuation uncertainty,” in *WAFR*, Dec. 2016.

[12] D. Silver and J. Veness, “Monte-carlo planning in large pomdps,” in *NIPS*, Dec. 2010.

[13] A. Somani, N. Ye, D. Hsu, and W. S. Lee, “Despot: Online pomdp planning with regularization,” in *NIPS*, Dec. 2013.

[14] A. Cimatti and M. Roveri, “Conformant planning via symbolic model checking,” *JAIR*, vol. 13, no. 1, pp. 305–338, 2000.

[15] M. Levihn, J. Scholz, and M. Stilman, “Planning with movable obstacles in continuous environments with uncertain dynamics,” in *ICRA*, May 2013.

[16] S. M. LaValle and J. James J. Kuffner, “Randomized kinodynamic planning,” *IJRR*, vol. 20, no. 5, pp. 378–400, 2001.

[17] B. Akgun and M. Stilman, “Sampling heuristics for optimal motion planning in high dimensions,” in *IROS*, Sept. 2011.

[18] O. Arslan and P. Tsiotras, “Use of relaxation methods in sampling-based algorithms for optimal motion planning,” in *ICRA*, May 2013.

[19] F. Islam, J. Nasir, U. Malik, Y. Ayaz, and O. Hasan, “Rrt#-smart: Rapid convergence implementation of rrt# towards optimal solution,” in *ICMA*, Aug. 2012.

[20] A. J. Dobson and K. E. Bekris, “Sparse roadmap spanners for asymptotically near-optimal motion planning,” *IJRR*, vol. 33, pp. 18–47, 2014.

[21] J. D. Marble and K. E. Bekris, “Asymptotically near-optimal planning with probabilistic roadmap spanners,” *T-RO*, vol. 29, no. 2, pp. 432–444, 2013.

[22] S. Karaman and E. Frazzoli, “Optimal kinodynamic motion planning using incremental sampling-based methods,” in *CDC*, Dec. 2010.

[23] J. h. Jeon, S. Karaman, and E. Frazzoli, “Anytime computation of time-optimal off-road vehicle maneuvers using the rrt*,” in *CDC and ECC*, Dec. 2011.

[24] H. Choset, K. M. Lynch, S. Hutchinson, G. A. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. Cambridge, MA: The MIT Press, 2005.