Model Error Propagation via Learned Contraction Metrics for Safe Feedback Motion Planning of Unknown Systems

Glen Chou, Necmiye Ozay, and Dmitry Berenson[1]

***Abstract*— We present a method for contraction-based feedback motion planning of locally incrementally exponentially stabilizable systems with unknown dynamics that provides probabilistic safety and reachability guarantees. Given a dynamics dataset, our method learns a deep control-affine approximation of the dynamics. To find a trusted domain where this model can be used for planning, we obtain an estimate of the Lipschitz constant of the model error, which is valid with a given probability, in a region around the training data, providing a local, spatially-varying model error bound. We derive a trajectory tracking error bound for a contraction-based controller that is subjected to this model error, and then learn a controller that optimizes this tracking bound. With a given probability, we verify the correctness of the controller and tracking error bound in the trusted domain. We then use the trajectory error bound together with the trusted domain to guide a sampling-based planner to return trajectories that can be robustly tracked in execution. We show results on a 4D car, a 6D quadrotor, and a 22D deformable object manipulation task, showing our method plans safely with learned models of high-dimensional underactuated systems, while baselines that plan without considering the tracking error bound or the trusted domain can fail to stabilize the system and become unsafe.**

I. Introduction

Provably safe motion planning algorithms for unknown systems are critical for deploying robots in the real world. Planners are reliable when the system dynamics are known exactly, but this is rarely the case. To address this, data-driven methods (e.g. model-based reinforcement learning) plan with dynamics models learned from data. However, such methods can be unsafe, since the planner can and will exploit errors in the learned model to return plans that cannot be tracked on the real system, leading to unreliable, unsafe behavior in execution. Thus, to guarantee safety, it is of major interest to bound the error that the true system may see when tracking a trajectory planned with the learned dynamics, and to use it to guide the planning of robustly-trackable trajectories.

One key property of learned dynamics models is their nonuniform error: they are accurate near the training data, and that accuracy degrades when moving away from it. Thus, the model error seen in execution depends on the domain visited, which also depends on the tracking controller, e.g. a poor controller will lead to the system visiting a larger set of possible states, and thus experiencing a larger possible model error. To analyze this, we need a bound on the trajectory tracking error for a given disturbance bound (i.e. a tracking tube). In this paper, we consider tracking controllers based on contraction theory. Introduced in [1] and extended to the control-affine case in [2], control contraction theory studies incremental stabilizability, making it uniquely suited for obtaining tracking tubes under disturbance. While tracking tubes have been derived for contraction-based controllers under simple disturbance bounds [3] (e.g. a UAV subject to wind with a known uniform upper bound), these assumptions are ill-suited for learned dynamics: a uniform bound can be highly conservative, as the large errors far from the data would lead to enormous tracking tubes, rendering planning infeasible. It is also difficult to bound the model error, as its values are only known on the training data.

To this end, we develop a method for safe contraction-based motion planning compatible with learned high-dimensional neural network (NN) dynamics models, which probabilistically guarantees safety and goal reachability for the true system. Our core insights are 1) that we can derive a tracking error bound for a contraction-based controller under a spatially-varying model error bound, and 2) that this error bound can be used to bias planning towards regions where plans can be more robustly tracked. Our contributions are:

- A trajectory tracking error bound for contraction-based controllers subjected to a spatially-varying, Lipschitz constant-based model error bound that accurately reflects the error in the learned dynamics model.
- A deep learning framework for joint learning of dynamics, control contraction metrics (CCMs), and contracting controllers that are approximately optimized for planning performance under this model error description.
- A sampling-based planner that returns plans which can be safely tracked under the learned dynamics/controller.
- Evaluation of our method on learned dynamics up to 22D, and demonstrating that it outperforms baselines.

II. Related Work

Our work is related to contraction-based control of uncertain systems: [3] applies contraction to feedback motion planning for systems with a known disturbance bound, while [4], [5] apply contraction to adaptive control under known model uncertainty structure, i.e. the uncertainty lies in the range of known basis functions. In this paper, the uncertainty arises from the error between the true dynamics and a learned NN approximation, which lacks such structure. It is also only known at certain points (the training data), making the disturbance bound *a priori* unknown and nontrivial to obtain. These methods use sum-of-squares (SoS) optimization to find CCMs for moderate-dimensional polynomial systems [3], and cannot be used for NN models. Thus, [6], [7] model the CCM as an NN and learn it from data, assuming known dynamics with disturbance of known uniform upper bound.

[1]Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI, {gchou, necmiye, dmitryb}@umich.edu

Our method differs by learning the dynamics and CCM together to optimize planning performance under model error. Also related is [8], which learns a model jointly with a CCM but does not consider how model error affects tracking.

Our work is also related to safe learning-based control. Many methods learn stability certificates for a single equilbirium point [9], [10], but this is insufficient for point-to-point motion planning. Other methods use Gaussian processes to bound the reachable tube of a trajectory [11] or safely explore a set [12], [13], but these methods assume a feedback controller is provided; we do not, as we learn a CCM-based controller. [14] learns tracking tubes around trajectories, but it is unclear how close plans must stay to the training data for the guarantees to hold. Perhaps most relevant is [15], which plans safely with learned models by enforcing that tubes around plans remain in a "trusted domain". A key assumption of [15] is that the unknown system has as many controls as states; we remove this assumption, requiring fundamental advancements from [15], e.g. in deriving a new tracking bound, controller, trusted domain, and planner.

III. Preliminaries and Problem Statement

We consider deterministic unknown continuous-time nonlinear systems $\dot{x} = h(x,u)$, where $h : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$, $\mathcal{X} \subseteq \mathbb{R}^{n_x}$, and $\mathcal{U} \subseteq \mathbb{R}^{n_u}$. We define $g : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$ to be a control-affine approximation of the true dynamics:

$$g(x,u) = f(x) + B(x)u. \tag{1}$$

While we do not assume that the true dynamics are control-affine, we do assume that they are locally incrementally exponentially stabilizable (IES), that is, there exists a β, $\lambda > 0$, and feedback controller such that $\|x^*(t) - x(t)\| \leq \beta e^{-\lambda t}\|x^*(0) - x(0)\|$ for all solutions $x(t)$ in a domain. Many underactuated systems satisfy this, and it is much weaker than requiring $n_x = n_u$, as in [15]. Also, this only needs to hold in a task-relevant domain D, defined later.

For a function η, a Lipschitz constant over a domain $\mathcal{Z}$ is any L such that for all $z_1, z_2 \in \mathcal{Z}$, $\|\eta(z_1) - \eta(z_2)\| \leq L\|z_1 - z_2\|$. Norms $\|\cdot\|$ are always the 2-norm. We define L_{h-g} as the smallest Lipschitz constant of the error $h - g$. The argument of $h - g$ is a state-control pair (x,u) and its value is a state. We define a ball $\mathcal{B}_r(x)$ as $\{y \mid \|y - x\| < r\}$, also referred to as a r-ball about x. We suppose the state space $\mathcal{X}$ is partitioned into safe $\mathcal{X}_{\text{safe}}$ and unsafe $\mathcal{X}_{\text{unsafe}}$ sets (e.g., collision states). We denote $\widehat{Q} \doteq Q + Q^\top$ as a symmetrization operation on matrix Q, and $\bar{\lambda}(\widehat{Q})$ and $\underline{\lambda}(\widehat{Q})$ as its maximum and minimum eigenvalues, respectively. We overload notation when $Q(x)$ is a matrix-valued function, denoting $\bar{\lambda}_{\mathcal{Q}}(Q) \doteq \sup_{x \in \mathcal{Q}} \bar{\lambda}(Q(x))$ and $\underline{\lambda}_{\mathcal{Q}}(Q) \doteq \inf_{x \in \mathcal{Q}} \underline{\lambda}(Q(x))$. Let $\mathbf{I}_n$ be the identity matrix of size $n \times n$. Let $\mathbb{S}_n^{>0}$ denote the set of symmetric, positive definite $n \times n$ matrices. Let the Lie derivative of a matrix-valued function $Q(x) \in \mathbb{R}^{n \times n}$ along a vector $y \in \mathbb{R}^n$ be denoted as $\partial_y Q(x) \doteq \sum_{i=1}^n y^i \frac{\partial Q}{\partial x^i}$. Let x^i denote the ith element of vector x. Let the notation $Q_\perp(x)$ refer to a basis for the null-space of matrix $Q(x)$.

Finally, we introduce the needed terminology from differential geometry. For a smooth manifold $\mathcal{X}$, a Riemannian metric tensor $M : \mathcal{X} \to \mathbb{S}_{n_x}^{>0}$ equips the tangent space $T_x\mathcal{X}$ at each element x with an inner product $\delta_x^\top M(x)\delta_x$, providing a local length measure. Then, the length $l(c)$ of a curve $c : [0,1] \to \mathcal{X}$ between points $c(0)$, $c(1)$ can be computed by integrating the local lengths along the curve: $l(c) \doteq \int_0^1 \sqrt{V(c(s), c_s(s))}ds$, where for brevity $V(c(s), c_s(s)) \doteq c_s(s)^\top M(c(s))c_s(s)$, and $c_s(s) \doteq \partial c(s)/\partial s$. Then, we can define the Riemannian distance between two points $p, q \in \mathcal{X}$ as $\text{dist}(p,q) \doteq \inf_{c \in \mathcal{C}(p,q)} l(c)$, where $\mathcal{C}(p,q)$ is the set of all smooth curves connecting p and q. Finally, we define the Riemannian energy between p and q as $\mathcal{E}(p,q) \doteq \text{dist}^2(p,q)$.

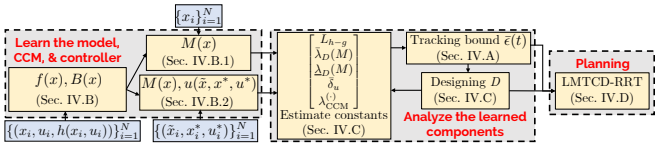

Fig. 1. Method. **L**: We learn a model/contracting controller (Prob. 1). **C**: We verify the controller and bound model/tracking error in D (Prob. 2). **R**: We use the tracking bound to find safely-trackable plans (Prob. 3).

A. Control contraction metrics (CCMs)

Contraction theory studies incremental stability by measuring the distances between system trajectories via a contraction metric $M(x) : \mathcal{X} \to \mathbb{S}_{n_x}^{>0}$, quantifying if the differential distances between trajectories $V(x, \delta_x) = \delta_x^\top M(x)\delta_x$ shrink with time. Control contraction metrics (CCMs) adapt this analysis to control-affine systems (1). For dynamics of the form (1), the differential dynamics can be written as $\dot{\delta}_x = (\frac{\partial f}{\partial x} + \sum_{i=1}^{n_u} u^i \frac{\partial B^i}{\partial x})\delta_x + B(x)\delta_u$ [3], where $B^i(x)$ is the ith column of $B(x)$. Then, we call $M(x) : \mathcal{X} \to \mathbb{S}_{n_x}^{>0}$ a CCM if there exists a differential controller δ_u such that the closed-loop system satisfies $\dot{V}(x, \delta_x) < 0$, for all x, δ_x.

How do we find a CCM $M(x)$ ensuring the existence of δ_u? First, define the dual metric $W(x) \doteq M^{-1}(x)$. Then, two sufficient conditions for contraction are (2)-(3) [3], [6]:

$$B_\perp(x)^\top \Big(-\partial_f W(x) + \widehat{\tfrac{\partial f(x)}{\partial x} W(x)} + 2\lambda W(x) \Big) B_\perp(x) \preceq 0 \tag{2a}$$

$$B_\perp(x)^\top \Big(\partial_{B^j} W(x) - \widehat{\tfrac{\partial B^j(x)}{\partial x} W(x)} \Big) B_\perp(x) = 0, j = 1 \ldots n_u \tag{2b}$$

$$\dot{M}(x) + \widehat{M(x)(A(x) + B(x)K(\tilde{x}, x^*, u^*))} + 2\lambda M(x) \prec 0 \tag{3}$$

where $A \doteq \frac{\partial f}{\partial x} + \sum_{i=1}^{n_u} u^i \frac{\partial B^i}{\partial x}$ and $K = \frac{\partial u(\tilde{x}, x^*, u^*)}{\partial x}$, where $u : \mathcal{X} \times \mathcal{X} \times \mathcal{U} \to \mathcal{U}$ is a feedback controller which takes as input the tracking deviation $\tilde{x}(t) \doteq x(t) - x^*(t)$ from a nominal state $x^*(t)$, as well as a state/control $x^*(t)$, $u^*(t)$ on the nominal state/control trajectory that is being tracked $x^* : [0,T] \to \mathcal{X}$, $u^* : [0,T] \to \mathcal{U}$. We refer to the LHSs of (2a) and (3) as $C^s(x)$ and $C^w(\tilde{x}, x^*, u^*)$, respectively. Intuitively, (2a) is a contraction condition simplified by the orthogonality condition (2b), which together imply that all directions where the differential dynamics lack controllability must be naturally contracting at rate λ. The conditions (2) are stronger than (3), which does not assume orthogonality.

How do we recover a tracking feedback controller $u(\tilde{x}, x^*, u^*)$ for (1) from (2) and (3)? For (2), the controller is implicit in the dual metric $W(x)$, and can be computed by solving a nonlinear optimization problem at runtime [3], [16]. In (3), $u(\tilde{x}, x^*, u^*)$ is directly involved as the function defining K; as a consequence, $M(x)$ and $u(\tilde{x}, x^*, u^*)$ both need to be found. The benefit of using (2) is that there are

fewer parameters to learn. However, as some systems may not satisfy the properties needed to apply (2), we resort to using (3) in these cases (see Sec. IV-B.2). Finally, for a given CCM $M(x)$ and controller $u(\tilde{x}, x^*, u^*)$ satisfying (2) or (3), the tracking error of any nominal trajectory $x^*(t)$ satisfies $\|x(t)-x^*(t)\| \leq \beta\|x(0)-x^*(0)\|e^{-\lambda t}$ for overshoot constant β. If the system is subjected to bounded perturbations, it is instead guaranteed to remain in a tube around $x^*(t)$.

B. Problem statement

Our method has three major components. First, we learn a model (1) and a CCM $M(x)$ and/or controller $u(\tilde{x}, x^*, u^*)$ for (1). Next, we analyze the learned (1), $M(x)$, and/or $u(\tilde{x}, x^*, u^*)$ to determine a trusted domain $D \subseteq \mathcal{X}\times\mathcal{U}$ where trajectories can be robustly tracked. Finally, we design a planner that connects states in D, such that under the tracking controller $u(\tilde{x}, x^*, u^*)$, the system remains safe in execution and reaches the goal. In this paper, we represent the approximate dynamics $g(x, u)$ with an NN, though our method is agnostic to its structure. Let $\mathcal{S} = \{(x_i, u_i, h(x_i, u_i))\}_{i=1}^N$ be the training data for g obtained by any means (e.g. sampling, demonstrations, etc.), and let $\Psi = \{(x_j, u_j, h(x_j, u_j))\}_{j=1}^M$ be a set of independent, identically distributed (i.i.d.) samples collected near $\mathcal{S}$. Then, our method solves the following:

Problem 1 (Learning)**.** *Given $\mathcal{S}$, learn a control-affine model g, a contraction metric $M(x)$, and find a contraction-based controller $u(\tilde{x}, x^*, u^*)$ that satisfies* (2) *or* (3) *over $\mathcal{S}$.*

Problem 2 (Analysis)**.** *Given Ψ, g, $M(x)$, and $u(\tilde{x}, x^*, u^*)$, design a trusted domain D. In D, find a model error bound $\|h(x, u) - g(x, u)\| \leq e(x, u)$, for all $(x, u) \in D$, and verify if for all $x \in D$, M and u are valid, i.e. satisfying* (2)*/*(3)*.*

Problem 3 (Planning)**.** *Given g, $M(x)$, $u(\tilde{x}, x^*, u^*)$, start x_I, goal x_G, goal tolerance μ, maximum tracking error tolerance $\hat{\mu}$, trusted domain D, and $\mathcal{X}_{\text{safe}}$, plan a nominal trajectory $x^* : [0, T] \to \mathcal{X}$, $u^* : [0, T] \to \mathcal{U}$ under the learned dynamics g such that $x(0) = x_I$, $\dot{x} = g(x, u)$, $\|x(T) - x_G\| \leq \mu$, and $x(t)$, $u(t)$ remains in $D \cap \mathcal{X}_{\text{safe}}$ for all $t \in [0, T]$. Also, guarantee that in tracking $(x^*(t), u^*(t))$ under the true dynamics h with $u(\tilde{x}, x^*, u^*)$, the system remains in $D \cap \mathcal{X}_{\text{safe}}$ and reaches $\mathcal{B}_{\hat{\mu}+\mu}(x_G)$.*

IV. Method

We first derive a tracking error bound for a CCM-based controller under a Lipschitz constant-based model error bound (Sec. IV-A). We show how to learn a dynamics model, CCM, and controller to optimize the tracking error bound (Sec. IV-B). Then we show how to design a trusted domain D and verify the validity of the controller and model/tracking error bounds inside D (Sec. IV-C). Finally, we show how the tracking bound can bias a planner to return safely-trackable plans (Sec. IV-D). We summarize our method in Fig. 1. Omitted proofs can be found in the extended version [17].

A. CCM-based tracking tubes under Lipschitz model error

We first establish a spatially-varying bound on model error within a trusted domain D which can be estimated from the model error evaluated at training points. For a single training point $(\bar{x}, \bar{u})$ and a novel point (x, u), we can bound the error between the true and learned dynamics at (x, u) using the triangle inequality and Lipschitz constant of the error L_{h-g}:

$$\begin{aligned}&\|h(x, u) - g(x, u)\| \\ &\quad\leq L_{h-g}\|(x, u) - (\bar{x}, \bar{u})\| + \|h(\bar{x}, \bar{u}) - g(\bar{x}, \bar{u})\|.\end{aligned} \tag{4}$$

As this holds between the novel point and all training points, the following (possibly) tighter bound can be applied:

$$\begin{aligned}\|h(x, u) - g(x, u)\| \leq \min_{1\leq i\leq N}\Big\{&L_{h-g}\|(x, u) - (x_i, u_i)\| \\ &+ \|h(x_i, u_i) - g(x_i, u_i)\|\Big\}.\end{aligned} \tag{5}$$

To exploit higher model accuracy near the training data, we define D as the union of r-balls around $\mathcal{S}$, where $r < \infty$:

$$D = \textstyle\bigcup_{i=1}^N \mathcal{B}_r(x_i, u_i). \tag{6}$$

For these bounds to hold, L_{h-g} must be a valid Lipschitz constant over D. In Sec. IV-C, we discuss how to obtain a probabilistically-valid estimate of L_{h-g} and how to choose r. We now derive an upper bound $\bar{\epsilon}(t)$ on the Euclidean tracking error $\epsilon(t)$ around a nominal trajectory $(x^*(t), u^*(t)) \subseteq D$ for a given metric $M(x)$ and feedback controller $u(\tilde{x}, x^*, u^*)$, such that the executed and nominal trajectories $x(t)$ and $x^*(t)$ satisfy $\|x(t) - x^*(t)\| \leq \bar{\epsilon}(t)$, for all $t \in [0, T]$, when subjected to the model error description (5). In Sec. IV-B, we discuss how M and u can be learned from data.

In [3], it is shown that by using a controller which is contracting with rate λ according to metric $M(x)$ for the *nominal* dynamics (1), the Riemannian energy $\mathcal{E}(t)$ of a *perturbed* control-affine system $\dot{x}(t) = f(x(t)) + B(x(t))u(t) + d(t)$ is bounded by the following differential inequality:

$$D^+\mathcal{E}(t) \leq -2\lambda\mathcal{E}(t) + 2\sqrt{\mathcal{E}(t)\bar{\lambda}_D(M)}\|d(t)\|, \tag{7}$$

where $\bar{\lambda}_D(M) = \sup_{x\in D}\bar{\lambda}(M(x))$ and $D^+(\cdot)$ is the upper Dini derivative of $(\cdot)$. Here, the energy $\mathcal{E}(t) = \mathcal{E}(x^*(t), x(t))$ is the squared trajectory tracking error according to the metric $M(x)$ at a given time t, and $d(t)$ is an external disturbance. Suppose that the only disturbance to the system comes from the discrepancy between the learned and true dynamics, i.e. $d(t) = h(x(t), u(t)) - g(x(t), u(t))$[1]. For short, let $e_i \doteq \|h(x_i, u_i) - g(x_i, u_i)\|$ be the training error of the ith data-point. In this case, we can use (5) to write:

$$\|d(t)\| \leq \min_{1\leq i\leq N}\left\{L_{h-g}\left\|\begin{bmatrix}x(t)\\u(t)\end{bmatrix} - \begin{bmatrix}x_i\\u_i\end{bmatrix}\right\| + e_i\right\}. \tag{8}$$

As (8) is spatially-varying, it suggests that in solving Prob. 3, plans should stay near low-error regions to encourage low error in execution. However, (8) is only implicit in the plan, depending on the state visited and feedback control applied *in execution*: $x(t) = x^*(t) + \tilde{x}(t)$ and $u(\tilde{x}(t), x^*(t), u^*(t)) = u^*(t) + u_{\text{fb}}(t)$. To derive a tracking bound that can directly inform planning, we first introduce the following lemma:

Lemma 1. *The Riemannian energy $\mathcal{E}(t)$ of the perturbed system $\dot{x}(t) = f(x(t)) + B(x(t))u(t) + d(t)$, where $\|d(t)\|$ satisfies* (8)*, satisfies the differential inequality* (11)*, where $\underline{\lambda}_D(M) = \inf_{x\in D}\underline{\lambda}(M(x))$, $\bar{u}_{\text{fb}}(t)$ is a time-varying upper bound on the feedback control $\|u(t) - u^*(t)\|$, and $i^*(t)$ achieves the minimum in* (8)*.*

[1] In addition to model error, we can also handle *runtime* external disturbances with a known upper bound; we assume the training data is noiseless.

Proof sketch. We use the triangle inequality to simplify (8):

$$\|d(t)\| \leq \min_{1\leq i\leq N}\left\{L_{h-g}\left(\left\|\begin{bmatrix}x^*(t)\\u^*(t)\end{bmatrix}-\begin{bmatrix}x_i\\u_i\end{bmatrix}\right\|+\left\|\begin{bmatrix}\tilde{x}(t)\\u_{\text{fb}}(t)\end{bmatrix}\right\|\right)+e_i\right\}$$
$$\leq L_{h-g}\left\|\begin{bmatrix}\tilde{x}(t)\\u_{\text{fb}}(t)\end{bmatrix}\right\|+\min_{1\leq i\leq N}\left\{L_{h-g}\left\|\begin{bmatrix}x^*(t)\\u^*(t)\end{bmatrix}-\begin{bmatrix}x_i\\u_i\end{bmatrix}\right\|+e_i\right\}.$$

Note that as $\|d(t)\|$ depends on $\tilde{x}(t)$, the disturbance bound itself depends on $\epsilon(t)$. To make this explicit, we use $\|\tilde{x}(t)\| = \epsilon(t)$ and $\|u_{\text{fb}}(t)\| \leq \bar{u}_{\text{fb}}(t)$ to obtain

$$\|d(t)\| \leq \; L_{h-g}\big(\epsilon(t)+\bar{u}_{\text{fb}}(t)\big)+ \min_{1\leq i\leq N}\left\{L_{h-g}\left\|\begin{bmatrix}x^*(t)\\u^*(t)\end{bmatrix}-\begin{bmatrix}x_i\\u_i\end{bmatrix}\right\|+e_i\right\}. \quad (9)$$

To obtain $\bar{u}_{\text{fb}}(t)$, if we use CCM conditions (2), we can use the optimization-based controller in [3] (cf. Sec. III-A), which admits the upper bound [3, p.28]:

$$\|u_{\text{fb}}(t)\| \leq \epsilon(t)\sup_{x\in D}\frac{\bar{\lambda}(L(x)^{-\top}F(x)L(x)^{-1})}{2\underline{\sigma}_{>0}(B^\top(x)L(x)^{-1})} \doteq \epsilon(t)\bar{\delta}_u, \quad (10)$$

where $W(x) = L(x)^\top L(x)$, $F(x) = -\partial_f W(x) + \widehat{\frac{\partial f(x)}{\partial x}W(x)} + 2\lambda W(x)$, and $\underline{\sigma}_{>0}(\cdot)$ is the smallest positive singular value. If we instead use condition (3), we must estimate $\bar{u}_{\text{fb}}(t)$ for the learned controller (cf. Sec. IV-C).

To obtain the result, we plug (9) into (7) after relating $\epsilon(t)$ with $\mathcal{E}(t)$. Since $\mathcal{E}(x^*(t),x(t)) = \text{dist}^2(x^*(t),x(t)) \geq \underline{\lambda}_D(M)\|x^*(t)-x(t)\|^2$, we have that $\epsilon(t) \leq \sqrt{\mathcal{E}(t)/\underline{\lambda}_D(M)}$. Finally, we can plug all of these components into (7) to obtain (11), where $i^*(t)$ denotes a minimizer of (8). □

For intuition, let us interpret (9). First, it depends on $\epsilon(t)$, which in turn relies on the disturbance magnitude: intuitively, with tighter tracking, the system visits a smaller set of states, thus experiencing lower worst-case model error. Second, it depends on $u_{\text{fb}}(t)$: if a large feedback is applied, the combined control $u(t) = u^*(t) + u_{\text{fb}}(t)$ can be far from the controls that the learned model is trained on, possibly leading to high error. Finally, it is driven by the model error and closeness to the training data (minimization term of (9)). We can also compare our tracking bound (11) with the tracking bound for a uniform disturbance bound (7). Notice that the "effective" contraction rate $\lambda - L_{h-g}\sqrt{\frac{\bar{\lambda}_D(M)}{\underline{\lambda}_D(M)}}$ shrinks with L_{h-g}, as the tracking error grows with model error. If the optimization-based controller [3] is used, the $\epsilon(t)$ dependence of (10) reduces this rate to $\lambda - L_{h-g}\sqrt{\frac{\bar{\lambda}_D(M)}{\underline{\lambda}_D(M)}}(1+\bar{\delta}_u)$. We note that a large model error can make this rate negative and cause rapid tube growth, restricting our planner to operate over a short-horizon. Now, we can derive the tracking bound:

Theorem 1 (Tracking bound under (8))**.** *Let $\mathcal{E}_{RHS}$ denote the RHS of* (11)*. Assuming that the perturbed system $\dot{x}(t) = f(x(t))+B(x(t))u(t)+d(t)$ satisfies $\mathcal{E}(t_1) \leq \mathcal{E}_{t_1}$ and $\|d(t)\|$ satisfies* (8)*. Then, $\bar{\epsilon}(t)$ is described at some $t_2 > t_1$ as:*

$$\bar{\epsilon}(t_2) = \sqrt{\big(\textstyle\int_{\tau=t_1}^{t_2}\mathcal{E}_{\text{RHS}}(t)d\tau\big)/\underline{\lambda}_D(M)}, \quad \mathcal{E}(t_1) = \mathcal{E}_{t_1}. \quad (12)$$

Note that Thm. 1 provides a Euclidean tracking error tube under the model error bound (8) for *any* nominal trajectory. Moreover, as (12) can be integrated incrementally in time, it is well-suited to guide planning in an RRT (Rapidly-exploring Random Tree [18]); see Sec. IV-D for more details.

B. Optimizing CCMs and controllers for the learned model

Having derived the tracking error bound, we discuss our solution to Prob. 1, i.e. how we learn the dynamics (1), a contraction metric $M(x)$, and (possibly) a stabilizing controller u in a way that minimizes (12). In this paper, we model $f(x)$, $B(x)$, $M(x)$, and $u(\tilde{x}, x^*, u^*)$ with NNs.

Ideally, we would learn the dynamics jointly with the contraction metric to minimize the size of the tracking tubes (12). In practice, this leads to poor learning (e.g. a valid CCM for inaccurate dynamics). Instead, we use a simple two step procedure: we first learn g, and then fix g and learn $M(x)$ and $u(\tilde{x}, x^*, u^*)$ for that model. While this is sufficient for our examples, in general alternating the learning may be helpful.

Dynamics learning. Inspecting (11), we note that the model-error related terms are the Lipschitz constant L_{h-g} and training error e_i, $i = 1,\ldots,N$. Thus, we train the dynamics using a loss on the mean squared error and a batch-wise estimate of the Lipschitz constant (which is finite provided each (x,u) in the batch is unique and the error is finite):

$$L_{\text{dyn}} = \frac{1}{N_b}\sum_{i=1}^{N_b}e_i^2 + \alpha_1 \max_{1\leq i\neq j\leq N_b}\left\{\frac{\|e_i-e_j\|}{\|(x_i,u_i)-(x_j,u_j)\|}\right\}, \quad (13)$$

where $e_i = \|g(x_i,u_i)-h(x_i,u_i)\|$, $N_b \leq N$ is the batch size, and α_1 trades off the objectives. Note that (13) promotes e_i to be small while remaining smooth over the training data, in order to encourage similar properties to hold over D.

CCM learning. We describe two variants of our learning approach, depending on if the stronger CCM conditions (2a) and (2b) or the weaker condition (3) is used.

1) Using (2a) *and* (2b)*:* We parameterize the dual metric as $W(x) = W_{\theta_w}(x)^\top W_{\theta_w}(x) + \underline{w}\mathbf{I}_{n\times n}$, where $W_{\theta_w}(x) \in \mathbb{R}^{n_x\times n_x}$, θ_w are the NN weights, and $\underline{w}$ is a minimum eigenvalue hyperparameter. This structure ensures that $W(x) \succ 0$ for all x. To enforce (2a), we follow [6], relaxing the matrix inequality to an penalty L^s_{NSD} over the training data, where:

$$L^{(\cdot)}_{\text{NSD}} = \max_{1\leq i\leq N_b}\bar{\lambda}\big(C^{(\cdot)}(x_i)\big). \quad (14)$$

As we ultimately wish (2a) to hold everywhere in D, we can use the continuity in x of the maximum eigenvalue $\bar{\lambda}(\lambda^s(x))$ to verify if (2a) holds over D (cf. Sec. IV-C). However, the equality constraints (2b) are problematic; by using unconstrained optimization, it is difficult to even satisfy (2b) on the training data, let alone on D. To address this, we follow [8] by restricting the dynamics learning to sparse-structured $B(x)$ of the form, where θ_B are NN parameters:

$$B(x) = [\mathbf{0}^\top_{n_x-n_u\times n_u}, \;\; B_{\theta_B}(x)^\top]^\top. \quad (15)$$

Restricting $B(x)$ to this form implies that to satisfy (2b), $W(x)$ must be a function of only the first $n_x - n_u$ states [8], which can be satisfied by construction. When this structural assumption does not hold, we use the method in Sec. IV-B.2

In addition to the CCM feasibility conditions, we introduce novel losses to optimize the tracking tube size (12). As (12) depends on the nominal trajectory, it is hard to optimize a tight upper bound on the tracking error independent of the plan. Instead, we maximize the effective contraction rate,

$$L^s_{\text{opt}} = \alpha_2 \max_{1\leq i\leq N_b}\Big(\lambda - L_{h-g}\sqrt{\frac{\bar{\lambda}(M(x_i))}{\underline{\lambda}(M(x_i))}}(1+\bar{\delta}_u(\tilde{x}_i))\Big), \quad (16)$$

$$D^+\mathcal{E}(t) \leq -2\left(\lambda - L_{h-g}\sqrt{\frac{\bar{\lambda}_D(M)}{\underline{\lambda}_D(M)}}\right)\mathcal{E}(t) + 2\sqrt{\mathcal{E}(t)\bar{\lambda}_D(M)}\left(L_{h-g}\left(\left\|\begin{bmatrix}x^*(t)\\u^*(t)\end{bmatrix} - \begin{bmatrix}x_{i^*(t)}\\u_{i^*(t)}\end{bmatrix}\right\| + \bar{u}_{\text{fb}}(t)\right) + e_{i^*(t)}\right) \tag{11}$$

where $\bar{\delta}_u(\tilde{x}_i)$ refers to the argument in the supremum in (10) and α_2 is a tuned parameter. Optimizing (16) while enforcing (2a) over the data is difficult for unconstrained NN optimizers. To ameliorate this, we use a linear penalty on constraint violation and switch to a logarithmic barrier [19] to maintain feasibility upon achieving it; let the combination of the linear and logarithmic penalties be denoted $\texttt{logb}(\cdot)$. Then, the full loss function can be written as $\texttt{logb}(-L^s_{\text{NSD}}) + L^s_{\text{opt}}$.

2) Using (3)*:* For systems that do not satisfy (15), we must use the weaker contraction conditions (3). In this case, we cannot use the optimization-based controllers proposed in [3], and we instead learn $u(\tilde{x}, x^*, u^*)$ in tandem with $M(x)$. As in (14), we enforce (3) by relaxing it to L^w_{NSD}. We represent $u(\tilde{x}, x^*, u^*)$ with the following structure:

$$u(\tilde{x}, x^*, u^*) = |\theta^u_1| \tanh\left(u_{\theta^u_2}(\tilde{x}, x^*)\tilde{x}\right) + u^*, \tag{17}$$

where θ^u_i are NN weights. Estimating $\bar{u}_{\text{fb}}$ for (17) is simple, as $\|u(\tilde{x}, x^*, u^*) - u^*\| < |\theta^u_1|$ for all x, x^*, u^*. We define L^w_{opt} as in (16), without the $\bar{\delta}_u$ term. Then, our full loss function is $\texttt{logb}(-L^w_{\text{NSD}}) + L^w_{\text{opt}} + \alpha_3|\theta^u_1|$. While local IES ensures that a CCM exists [1], [8], we still may not find a valid CCM due to local minima/poor hyperparameters. We found training reliability was improved by using the log-barrier and by increasing α_i, $i = 1, ..., 3$, with the training epoch, and that results are insensitive to $\bar{w}$. If we find a valid CCM on $\mathcal{S}$, we can check if it is also valid on D, as we discuss now.

C. Designing and verifying the trusted domain

Algorithm 1: Estimating the maximum of $\eta(z)$ over $\mathcal{Z}$

Input: N_s, N_b, ρ
for $j = 1, \ldots, N_s$ **do**
  generate i.i.d. samples $\{z^{i,j}\}_{i=1}^{N_b}$ over $\mathcal{Z}$
  compute $s_j = \max_{1 \leq i \leq N_b} \eta(z^{i,j})$
fit Weibull to $\{s_j\}$ to obtain $\hat{\gamma}$ and standard error ξ
validate fit using KS test with significance level 0.05
if validated **return** $\hat{\eta}_{\max} = \hat{\gamma} + \Phi^{-1}(\rho)\xi$ **else return** failure

The validity of the bound (12) requires overestimates of L_{h-g}, $\bar{\lambda}_D(M)$, and $\bar{\delta}_u$, an underestimate of $\underline{\lambda}_D(M)$, and for (2a)/(3) to hold in D. We describe how we solve Prob. 2, showing how to design D and estimate these constants in D.

For a given D, we can over/under-estimate the constants with a user-defined probability ρ via a stochastic approach from extreme value theory. We describe the general algorithm (Alg. 1) and refer to [15, p.3], [20] for more details. Alg. 1 estimates the maximum of a function $\eta(z)$ over a domain $\mathcal{Z}$ by taking N_s batches of samples over $\mathcal{Z}$ and computing the empirical maximum of $\eta(z)$ over each batch, s_j. If $\max_{z \in \mathcal{Z}} \eta(z)$ is finite and the distribution over s_j converges with increasing N_s, the Fisher-Tippett-Gnedenko (FTG) theorem [21] dictates that it must converge to a Weibull distribution. This can be empirically verified by fitting a Weibull distribution to the s_j, and validating the fit with a Kolmogorov-Smirnov (KS) goodness-of-fit test [22]. If the test passes, the location parameter $\hat{\gamma}$ of the fit distribution, adjusted with a confidence interval $\Phi^{-1}(\rho)\xi$ that scales with larger ρ, is an over-estimate of the maximum with probability ρ. Here, $\Phi^{-1}(\cdot)$ is the standard normal cumulative distribution function and ξ is the standard error of the fit $\hat{\gamma}$.

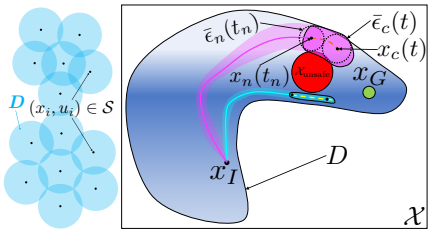

Fig. 2. **L**: example of D. **R**: LMTCD-RRT. Darker areas have smaller model error. The magenta extension is rejected (tube exits D/intersects unsafe set); the cyan extension is accepted: an example of bias towards low-error areas.

To estimate L_{h-g}, we follow [15] by running Alg. 1, setting $\mathcal{Z} = D \times D$ and $\eta(\cdot)$ as the slope between a pair of points drawn i.i.d. from D. Alg. 1 can also be used to estimate $\bar{\lambda}_D(M)$, $-\underline{\lambda}_D(M)$, and $\bar{\delta}_u$: here, $\mathcal{Z} = \text{proj}_x(D)$, where $\text{proj}_x(D) \doteq \bigcup_{\bar{x} \in \mathcal{S}} \mathcal{B}_r(\bar{x}) \supset \{x \mid \exists u, (x, u) \in D\}$, and $\eta(x) = \bar{\lambda}_D(M(x))$, $-\underline{\lambda}_D(M(x))$, and $\bar{\delta}_u(x)$ respectively. Since the eigenvalues of a continuously parameterized matrix function are continuous in the parameter [23] (here, the parameter is x) and D is bounded, these constants are finite, so by FTG, we can expect the samples s_j to be Weibull. Finally, FTG can also verify that (2a) and (3) are satisfied over D, since the verification is equivalent to ensuring $\sup_{x \in \text{proj}_x(D)} \bar{\lambda}(C^{(\cdot)}(x)) \leq \lambda^{(\cdot)}_{\text{CCM}}$ for some $\lambda^{(\cdot)}_{\text{CCM}} < 0$. λ^s_{CCM} can be estimated by setting $\mathcal{Z} = \text{proj}_x(D)$ and $\eta(x) = \bar{\lambda}(C^s(x))$. To estimate λ^w_{CCM}, we set $\mathcal{Z} = \mathcal{B}_{\epsilon_{\max}}(0) \times D$ and $\eta(\tilde{x}, x^*, u^*) = \bar{\lambda}(C^w(\tilde{x}, x^*, u^*))$, and sample $(x^*, u^*) \in D$ and $\tilde{x} \in \mathcal{B}_{\epsilon_{\max}}(0)$. Here, $\epsilon_{\max} \leq \hat{\mu}$ will upper-bound the allowable tracking tube size during planning (cf. Alg. 2, line 8); thus, to ensure that planning is minimally constrained, $\epsilon_{\max}$ should be chosen to be as large as possible while ensuring $\lambda^w_{\text{CCM}} < 0$. As all samples are i.i.d., the probability of (12) holding, and thus the overall safety probability of our method, is the product of the user-selected ρ for each constants. Note that other than for L_{h-g}, this estimation does not affect data-efficiency, as it queries the *learned dynamics* and requires no new data of the form $(x, u, h(x, u))$.

Finally, we discuss how to select r, which determines D (Fig. 2, left). An ideal r is maximally permissive for planning, where we must ensure that the tube around the plan remains within D (cf. Sec. IV-D). However, finding such an r is non-trivial and requires trading off many factors. Increasing r expands D; however, model error and L_{h-g} also increase with increased r, which may make $\bar{\epsilon}(t)$ and $\bar{u}_{\text{fb}}(t)$ grow, which in turn expands the tubes, making them harder to fit in D. Also, (2a)/(3) may not be satisfied over D for large r. For small r, the model error and L_{h-g} remain smaller due to the closeness to $\mathcal{S}$, leading to smaller tubes, but planning can be difficult, as D may be too small to contain even these smaller tubes. In particular, planning between two states in D can become infeasible if D becomes disconnected.

To trade off these factors, we propose the following solution for selecting r. We first find a minimum r, r_{connect}, such that D is fully-connected. Depending on how $\mathcal{S}$ is

collected, one may wish to first filter out outliers far from the bulk of the data. We calculate the connected component by considering the dataset as a graph, where an edge between $(x_i, u_i), (x_j, u_j) \in \mathcal{S}$ exists if $\|(x_i, u_i) - (x_j, u_j)\| \leq r$. We then check if the contraction condition (2a)/(3) is satisfied for $r = r_{\text{connect}}$, using the FTG-based procedure. If it is not satisfied, we decrement r until (2a)/(3) holds, and select r as the largest value for which (2a)/(3) are satisfied. Since $r < r_{\text{connect}}$ here, planning can only be feasible between starts/goals in each connected component; to rectify this, more data should be collected to train the CCM/controller. If the contraction condition is satisfied at $r = r_{\text{connect}}$, we incrementally increase r, starting from r_{connect}. In each iteration, we first determine if the contraction condition (2a)/(3) is still satisfied for the current r, using the FTG-based procedure. If the contraction condition is satisfied, we evaluate an approximate measure of planning permissiveness under "worst-case" conditions[2]: $r - \bar{\epsilon}(t) - \bar{u}_{\text{fb}}(t)$, evaluated at a fixed time $t = T_{\text{query}}$, where $\bar{\epsilon}(t)$ and $\bar{u}_{\text{fb}}(t)$ are computed assuming that for all $t \in [0, T_{\text{query}}]$, $\|(x^*(t), u^*(t)) - (x_{i^*(t)}, u_{i^*(t)})\| = \max_{1 \leq i \leq N} \min_{1 \leq j \leq N} \|(x_i, u_i) - (x_j, u_j)\|$, i.e. the dispersion of the training data, and experiences the worst training error (i.e. $e_{i^*(t)} = \max_{1 \leq i \leq N} e_i$, for all t). If (2a)/(3) is not satisfied, we terminate the search and select the r with the highest permissiveness, as measured by the above procedure.

D. Planning with the learned model and metric

Algorithm 2: LMTCD-RRT

Input: x_I, x_G, $\mathcal{S}$, $\{e_i\}_{i=1}^N$, estimated constants, μ, $\mathcal{E}_0$

$\mathcal{T} \leftarrow \{(x_I, \sqrt{\mathcal{E}_0/\underline{\lambda}_D(M)}, 0)\}$; $\mathcal{P} \leftarrow \{(\emptyset, \emptyset)\}$ // (state, energy, time)

while True **do**

  $(x_{\text{n}}, \bar{\epsilon}_{\text{n}}, t_{\text{n}}) \leftarrow$ `SampleNode` $(\mathcal{T})$

  $(u_{\text{c}}, t_{\text{c}}) \leftarrow$ `SampleCandidateControl` ()

  $(x_{\text{c}}^*(t), u_{\text{c}}^*(t)) \leftarrow$ `IntegrateLearnedDyn` $(x_{\text{n}}, u_{\text{c}}, t_{\text{c}})$

  $\bar{\epsilon}_c(t) \leftarrow$ `TrkErrBndEq12` $(\bar{\epsilon}_{\text{n}}, x_{\text{c}}^*(t), u_{\text{c}}^*(t), \mathcal{S}, \{e_i\}_{i=1}^N)$

  $D^1_{\text{chk}} \leftarrow (x^*(t), u^*(t)) \in D_{\bar{\epsilon}_c(t) - \bar{u}_{\text{fb}}(t)}, \forall t \in [t_{\text{n}}, t_{\text{n}} + t_{\text{c}})$

  if *controller learned* **then** $D^2_{\text{chk}} \leftarrow \bar{\epsilon}_c(t) \leq \epsilon_{\max}, \forall t \in [t_{\text{n}}, t_{\text{n}} + t_{\text{c}})$

  else $D^2_{\text{chk}} \leftarrow$ True

  $C \leftarrow$ `InCollision` $(x^*(t), u^*(t), \bar{\epsilon}_c(t))$

  if $D^1_{\text{chk}} \wedge D^2_{\text{chk}} \wedge \neg C$ **then**

    $\mathcal{T} \leftarrow \mathcal{T} \cup \{(x^*(t_{\text{n}} + t_{\text{c}}), \bar{\epsilon}_c(t_{\text{n}} + t_{\text{c}}), t_{\text{c}})\}$; $\mathcal{P} \leftarrow \mathcal{P} \cup \{(u_{\text{c}}, t_{\text{c}})\}$

  else continue

  if $\exists t, x_c^*(t) \in \mathcal{B}_\mu(x_G)$ **then** break; return plan

Finally, we discuss our solution to safely planning with the learned dynamics (Prob. 3). We develop an incremental sampling-based planner akin to a kinodynamic RRT [18], growing a search tree $\mathcal{T}$ by forward-propagating sampled controls held for sampled dwell-times, until the goal is reached. To ensure the system remains within D in execution (where the contraction condition and (12) are valid), we impose additional constraints on where $\mathcal{T}$ is allowed to grow.

Denote $D_q = D \ominus \mathcal{B}_q(0)$ as the state/controls which are at least distance q from the complement of D, where $\ominus$ refers to the Minkowski difference. Since (12) defines tracking error tubes for *any* given nominal trajectory, we can efficiently compute tracking tubes along any candidate edge of an RRT. Specifically, suppose that we wish to extend the RRT from a state on the planning tree $x^*_{\text{cand}}(t_1)$ with initial energy satisfying $\mathcal{E}_{\text{cand}}(t_1) \leq \mathcal{E}_{t_1}$ to a candidate state $x^*_{\text{cand}}(t_2)$ by applying control u over $[t_1, t_2)$. This information is supplied to (12), and we can obtain the tracking error $\bar{\epsilon}_{\text{cand}}(t)$, for all $t \in [t_1, t_2)$. Then, if we enforce that $(x^*(t), u^*(t)) \in D_{\bar{\epsilon}_{\text{cand}}(t) + \bar{u}_{\text{fb}}(t)}$ for all $t \in [t_1, t_2)$, we can ensure that the true system remains within D when tracked with a controller that satisfies $u_{\text{fb}}(t) \leq \bar{u}_{\text{fb}}(t)$ in execution. Otherwise, the extension is rejected and the sampling continues. When using a learned $u(\tilde{x}, x^*, u^*)$ with (3), an extra check that $\bar{\epsilon}_{\text{cand}}(t) \leq \epsilon_{\max}$ is needed to remain in $\mathcal{B}_{\epsilon_{\max}}(0) \times D$ (cf. Sec. IV-C). Since D is a union of balls, exactly checking $(x^*(t), u^*(t)) \in D_{\bar{\epsilon}(t) + \bar{u}_{\text{fb}}(t)}$ can be unwieldy. However, a conservative check can be efficiently performed by evaluating (18):

Theorem 2. *If* (18) *holds for some index* $1 \leq i \leq N$ *in* $\mathcal{S}$,

$$\|(x^*(t), u^*(t)) - (x_i, u_i)\| \leq r - \bar{\epsilon}(t) - \bar{u}_{\text{fb}}(t), \tag{18}$$

then $(x^*(t), u^*(t)) \in D_{\bar{\epsilon}(t) + \bar{u}_{\text{fb}}(t)}$.

We collision-check the tracking tubes and obstacles, which we assume are expanded for the robot geometry; this is simplified by the fact that (12) defines a sphere. We visualize our planner (Fig. 2, right), which we call **L**earned **M**odels in **T**rusted **C**ontracting **D**omains (LMTCD-RRT), and summarize it in Alg. 2. We note that the key idea of our planner (i.e. ensuring that the tracking tubes remain in $D \cap \mathcal{X}_{\text{safe}}$) can be applied to other sampling-based planners or trajectory optimizers, e.g. [24], [25]. However, enforcing the highly non-convex constraint of remaining in $D \cap \mathcal{X}_{\text{safe}}$ in a trajectory optimizer can be difficult, and is a direction for future work. We conclude with this correctness result:

Theorem 3 (LMTCD-RRT correctness)**.** *Assume that the estimated* L_{h-g}, $\bar{\lambda}_D(M)$, $\bar{u}_{\text{fb}}(t)$, *and* $\lambda^{(\cdot)}_{\text{CCM}}$ *overapproximate their true values and the estimated* $\underline{\lambda}_D(M)$ *underapproximates its true value. Then, when using a controller* $u(\tilde{x}, x^*, u^*)$ *derived from* (2)*, Alg. 2 returns a trajectory* $(x^*(t), u^*(t))$ *that remains within* D *in execution on the true system. Moreover, when using a controller* $u(\tilde{x}, x^*, u^*)$ *derived from* (3)*, Alg. 2 returns a trajectory* $(x^*(t), u^*(t))$ *such that* $(\tilde{x}^*(t), x^*(t), u^*(t))$ *remains in* $\mathcal{B}_{\epsilon_{\max}}(0) \times D$ *in execution on the true system.*

V. Results

We evaluate LMTCD-RRT on a 4D nonholonomic car, a 6D underactuated quadrotor, and a 22D rope manipulation task. We compare with four baselines to show the need for both using the bound (12) and remaining in D, where (12) is valid: B1) planning in D and assuming model error is uniformly bounded by the average training error $\|d(t)\| \leq \sum_{i=1}^N e_i/N$ to compute $\bar{\epsilon}(t)$, B2) planning in D and using the maximum training error $\|d(t)\| \leq \max_{1 \leq i \leq N} e_i$ as a uniform bound, B3) not remaining in D in planning and using the uniform maximum error bound $\|d(t)\| \leq \max_{1 \leq i \leq N} e_i$, and B4) not remaining in D and using our error bound (12). We note that B3-type assumptions are common in prior CCM work [3], [6]. In baselines that leave D, the space is unconstrained: $\mathcal{X} = \mathbb{R}^{n_x}, \mathcal{U} = \mathbb{R}^{n_u}$. We set $\rho = 0.975$ when estimating constants via FTG. See Table I for statistics and https://youtu.be/DYEyD5y-2po for results visualizations.

[2]Roughly, this compares the size of D to the tracking error tube size and feedback control bound, cf. Sec. IV-D and Thm. 2 for further justification.

	Avg. trk. error (Car)	Goal error (Car)	Avg. trk. error (Quadrotor)	Goal error (Quadrotor)	Avg. trk. error (Rope)	Goal error (Rope)
LMTCD-RRT	0.008 ± 0.004 (0.024)	0.009 ± 0.004 (0.023)	0.0046 ± 0.0038 (0.0186)	0.0062 ± 0.0115 (0.0873)	0.0131 ± 0.0063 (0.0278)	0.0125 ± 0.0095 (0.0352)
B1: Mean, in D	0.019 ± 0.012 (0.054)	0.023 ± 0.016 (0.078)	0.0052 ± 0.0051 (0.0311)	0.0104 ± 0.0161 (0.0735)	18.681 ± 55.917 (167.79)	42.307 ± 126.81 (380.45)
B2: Max, in D	0.02 ± 0.01 (0.05) [19/50]	0.019 ± 0.012 (0.062) [19/50]	— [65/65]	— [65/65]	17.539 ± 52.380 (157.22)	21.595 ± 64.295 (193.05)
B3: Max, $\notin D$	0.457 ± 0.699 (3.640)	1.190 ± 1.479 (7.434)	0.1368 ± 0.2792 (1.5408)	0.8432 ± 1.3927 (9.0958)	111.86 ± 39.830 (170.96)	236.34 ± 72.622 (331.83)
B4: Lip., $\notin D$	0.704 ± 2.274 (13.313)	2.246 ± 8.254 (58.32)	0.4136 ± 0.4321 (1.9466)	1.8429 ± 1.5260 (6.9859)	17.301 ± 49.215 (148.43)	36.147 ± 52.092 (147.76)

TABLE I

STATISTICS FOR THE CAR, QUADROTOR, AND ROPE. MEAN ± STANDARD DEVIATION (WORST CASE) [IF NONZERO, NUMBER OF FAILED TRIALS].

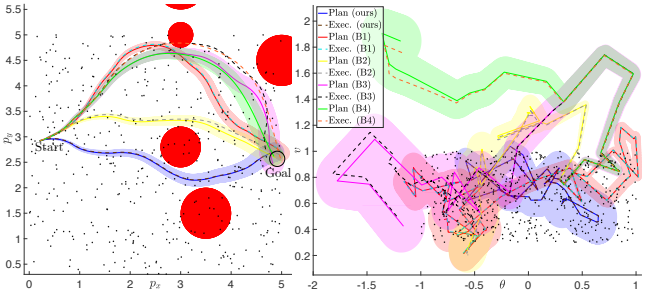

Fig. 3. 4D car; planned (solid) and executed (dotted); obstacles (red); tracking tubes (color-coded to the plan); black dots (subset of $\mathcal{S}$). We plot state space projections: **L**: x, y; **R**: θ, v. LMTCD-RRT, B1, and B2 remain in D in execution; B3 and B4 exit D and their tubes, crashing.

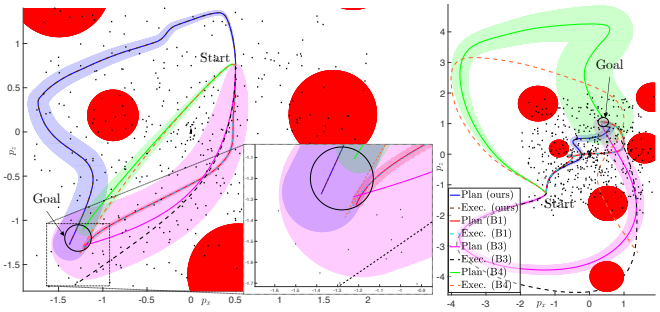

Fig. 4. 6D quadrotor; see Fig. 3 for color scheme. **L**: LMTCD-RRT remains in its tracking tube; all baselines exit their tubes (see inset). **R**: LMTCD-RRT remains within its tube; B3, B4 exit D and crash.

Nonholonomic car (4D): We use the model $[\dot{p}_x, \dot{p}_y, \dot{\theta}, \dot{v}] = [v\cos(\theta), v\sin(\theta), \omega, a]^\top$, where $u = [\omega, a]^\top$. As this model satisfies (15), we use the stronger CCM conditions (2). We use 50000 data-points uniformly sampled in $[0,5]\times[-5,5]\times[-1,1]\times[0.3,1]$ to train f, B, and M, where the states in $\mathcal{S}$ are used to train $M(x)$. We model f and B as NNs with one hidden layer of size 1024 and 16, respectively. We model $M(x)$ as an NN with two hidden layers, each of size 128. In training, we set $\underline{w} = 0.01$ and gradually increase α_1 and α_2 to 0.01 and 10, respectively. We select $r = 0.6$ by incrementally growing r as described in Sec. IV-C, collecting 5000 new datapoints for Ψ, giving us $\lambda = 0.09$, $L_{h-g} = 0.006$. $\bar{\delta}_u = 1.01$, $\bar{\lambda}_D(M) = 0.258$, and $\underline{\lambda}_D(M) = 0.01$.

We plan for 50 different start/goal states in D, taking on average 6 mins. We visualize one trial in Fig. 3. Over the trials, LMTCD-RRT and B2 never exit their tubes, while B1, B3, and B4 exit their tubes in 6, 48, and 43 of the 50 trials, respectively, which can lead to crashes (indeed, B3 and B4 crash in Fig. 3). This occurs as the baselines may underestimate the true model error seen in execution. Planning is also infeasible in 19/50 of B2's trials, since the large resulting tubes can invalidate all trajectories to the goal that remain in D, suggesting that a fine-grained bound like (9) is quite useful when D is highly constrained. Note that while B2 does not crash, using the maximum training error can still be unsafe (as seen later), as the true error can be higher in $D \setminus \mathcal{S}$. Finally, we note the tracking accuracy difference between LMTCD-RRT and B1/B2 reflects that our bound (12) steers LMTCD-RRT towards lower error regions in D. Overall, this example suggests that (12) is accurate, and using coarser bounds or exiting D can be unsafe.

Underactuated planar quadrotor (6D): We use the model in [3, p.20], where $x = [p_x, p_z, \phi, v_x, v_z, \dot{\phi}]$ models position and velocity, $u = [u_1, u_2]$ models thrust, and we use the constants $m = 0.486$, $l = 0.25$, and $J = 0.125$. This model also satisfies (15), so we use (2). We sample 245000 data-points in $[-2,2]\times[-2,2]\times[-\pi/3,\pi/3]\times[-1,1]\times[-1,1]\times[-\pi/4,\pi/4]$ to train f, B, and M. We model f and B as NNs with one hidden layer of size 1024 and 16. We model $M(x)$ as an NN with two hidden layers of size 128. In training, we set $\underline{w} = 0.01$ and gradually increase α_1 and α_2 to 0.001 and 0.33. We obtain $r = 1.0$ via Sec. IV-C, yielding $|\Psi| = 10000$. This gives us $\lambda = 0.09$, $L_{h-g} = 0.007$. $\bar{\delta}_u = 1.9631$, $\bar{\lambda}_D(M) = 4.786$, and $\underline{\lambda}_D(M) = 0.0909$.

We plan for 65 start/goal pairs in D, taking 1 min on average (see Fig. 4). B2 fails entirely, as the error bound is too large to feasibly plan in D for all trials. Over these trials, LMTCD-RRT never violates its bound in execution, while B1, B3, and B4 violate their bounds 14/65, 32/65, and 65/65 times, respectively, since they underestimate the true model error. From Table I, LMTCD-RRT obtains the lowest error and is closely matched by B1. However, LMTCD-RRT never violates its tubes, while B1 does (e.g. Fig. 4, left). B3-B4 perform poorly as the high model error causes crashes (Fig. 4, right). Overall, this example shows that LMTCD-RRT can plan safely on learned highly-underactuated systems.

10-link rope (22D): To show our method scales to high-dimensional, non-polynomial systems beyond the capabilities of SoS-based methods, we consider a planar rope manipulation task in Mujoco [26]. We model the rope with 10 elastic links (11 nodes), and the head of the rope (see Fig. 5(d)) is velocity-controlled. There are 22 states: 2 for the xy head position and 20 for the xy positions of the other nodes, relative to the head. There are two controls: the commanded xy head velocities. We wish to steer the rope's *tail* to an xy goal region while ensuring the rope does not collide in execution (cf. Fig. 5). This is difficult, as the tail is highly underactuated. We obtain three demonstrations to train the dynamics (see video): one steers the rope in a counterclockwise loop; the other two start vertically (horizontally) and move up (right). We also evaluate the dynamics at nearby state/control perturbations, giving $|\mathcal{S}| = 41000$. As the rope dynamics do not satisfy (15), we learn both $M(x)$ and $u(\tilde{x}, x^*, u^*)$. We model f and B as three-layer NNs of size 512. $M(x)$ has two hidden layers of size 128, and $u(\tilde{x}, x^*, u^*)$ has a single hidden layer of size 128. In training, we set $\underline{w} = 1.0$ and gradually increase α_1 and α_3 to 0.005 and 0.56. To ensure the CCM/controller are

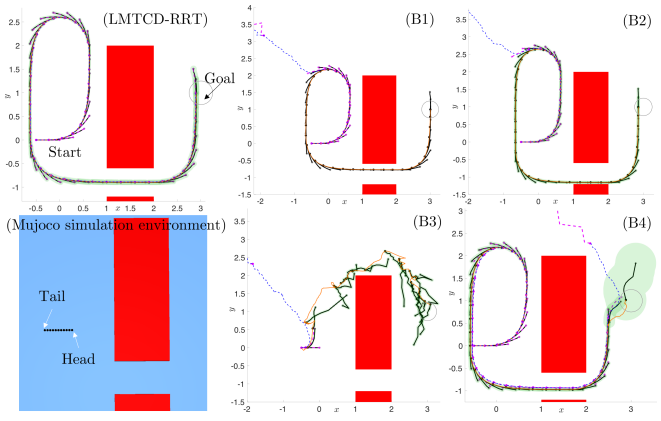

Fig. 5. 22D planar rope dragging task. Snapshots of planned/executed trajectory: black/magenta; tracking tubes: green. For each snapshot: we mark the rope head/tail with an asterisk/solid dot. Tail trajectory in the plan/in execution: orange/blue. Only LMTCD-RRT reaches the goal; all baselines destabilize. The original Mujoco environment is in the bottom left.

translation-invariant, we enforce $M(x)$ and $u(\tilde{x}, x^*, u^*)$ to not be a function of the head position. To simplify further, we hardcode the head dynamics as a single-integrator (as it is velocity-controlled) and learn the dynamics for the other 20 states. We find $\epsilon_{\text{max}} = 0.105$ and $r = 0.5$ (using the method in Sec. IV-C), resulting in $|\Psi| = 10000$, $\lambda = 0.0625$, $L_{h-g} = 0.023$, $\bar{u}_{\text{fb}} = 0.249$, $\bar{\lambda}_D(M) = 3.36$, and $\underline{\lambda}_D(M) = 1$.

We plan for 10 start/goal pairs in D, taking 9 min on average. Since we use (17), we change B1 and B2 to remain in $\mathcal{B}_{\epsilon_{\text{max}}}(0) \times D$, while B3 and B4 are unconstrained. We show one task in Fig. 5: the rope starts horizontally, with the head at $[0, 0]$, and needs to steer the tail to $[3, 0]$, within a 0.15 tolerance. LMTCD-RRT stays very close to the training data, reaching the goal with small tracking tubes. B1 and B2 also stay near the data, as they plan in D, but their bounds underestimate the true model error in D. In execution, the rope leaves D and destabilizes as the learned controller applies large inputs in an effort to return to the plan. B3 exploits model error to return an unrealistic plan; the rope immediately destabilizes in execution. This occurs as the maximum error severely underestimates model error outside D, and thus the tracking error. To move through the narrow passage, B4 is forced to remain near the training data at first. This is since the Lipschitz bound, while an underestimate outside of D, still grows quickly with distance from $\mathcal{S}$; attempting to plan a trajectory similar to B3 fails, since the tracking tube grows so large that it blocks off any paths to the goal. After getting through the narrow passage, B4 drifts from D and fails to be tracked beyond this point. Over 10 trials, LMTCD-RRT never violates its tracking bound, and B1, B2, B3, and B4 violate their bounds in 10, 6, 10, and 9 trials out of 10, respectively. Overall, this result suggests that contraction-based control can scale to very high-dimensional systems if one stays where the model/controller are good.

VI. Conclusion

We present a method for safe feedback motion planning with unknown dynamics. To achieve this, we jointly learn a dynamics model, a contraction metric, and contracting controller, and analyze the learned model error and trajectory tracking bounds under that model error description, all within a trusted domain. We then use these tracking bounds together with the trusted domain to guide the planning of probabilistically-safe trajectories; our results demonstrate that ignoring either component can lead to plan infeasibility or unsafe behavior. Future work involves extending our method to consider noisy training data and to plan safely with latent dynamics models learned from image observations.

Acknowledgements: We thank Craig Knuth for insightful feedback. This work is supported in part by NSF grants IIS-1750489, ECCS-1553873, ONR grants N00014-21-1-2118, N00014-18-1-2501, and an NDSEG fellowship.

References

[1] W. Lohmiller and J. E. Slotine, "On contraction analysis for non-linear systems," *Autom.*, vol. 34, no. 6, pp. 683–696, 1998.
[2] I. R. Manchester and J. E. Slotine, "Control contraction metrics: Convex and intrinsic criteria for nonlinear feedback design," *IEEE Trans. Autom. Control.*, vol. 62, no. 6, pp. 3046–3053, 2017.
[3] S. Singh, B. Landry, A. Majumdar, J. E. Slotine, and M. Pavone, "Robust feedback motion planning via contraction theory," 2019.
[4] A. Lakshmanan, A. Gahlawat, and N. Hovakimyan, "Safe feedback motion planning: A contraction theory and l_1-adaptive control based approach," *CDC*, 2020.
[5] B. T. Lopez, J. E. Slotine, and J. P. How, "Robust adaptive control barrier functions: An adaptive & data-driven approach to safety," *IEEE Control. Syst. Lett.*, vol. 5, no. 3, pp. 1031–1036, 2021.
[6] D. Sun, S. Jha, and C. Fan, "Learning certified control using contraction metric," *CoRL*, 2020.
[7] H. Tsukamoto and S. Chung, "Neural contraction metrics for robust estimation and control: A convex optimization approach," *IEEE Control. Syst. Lett.*, vol. 5, no. 1, pp. 211–216, 2021.
[8] S. Singh, S. M. Richards, V. Sindhwani, J. E. Slotine, and M. Pavone, "Learning stabilizable nonlinear dynamics with contraction-based regularization," *IJRR*, 2020.
[9] G. Manek and J. Z. Kolter, "Learning stable deep dynamics models," in *NeurIPS*, 2019, pp. 11 126–11 134.
[10] N. M. Boffi, S. Tu, N. Matni, J. E. Slotine, and V. Sindhwani, "Learning stability certificates from data," *CoRL*, 2020.
[11] T. Koller, F. Berkenkamp, M. Turchetta, and A. Krause, "Learning-based model predictive control for safe exploration," in *CDC*, 2018.
[12] A. K. Akametalu, J. F. Fisac, J. H. Gillula, S. Kaynama, M. N. Zeilinger, and C. J. Tomlin, "Reachability-based safe learning with gaussian processes," in *CDC*, 2014, pp. 1424–1431.
[13] F. Berkenkamp, R. Moriconi, A. P. Schoellig, and A. Krause, "Safe learning of regions of attraction for uncertain, nonlinear systems with gaussian processes," in *CDC*, 2016, pp. 4661–4666.
[14] D. D. Fan, A. Agha-mohammadi, and E. A. Theodorou, "Deep learning tubes for tube MPC," *RSS*, 2020.
[15] C. Knuth, G. Chou, N. Ozay, and D. Berenson, "Planning with learned dynamics: Probabilistic guarantees on safety and reachability via lipschitz constants," *IEEE RA-L*, 2021.
[16] K. Leung and I. R. Manchester, "Nonlinear stabilization via control contraction metrics: A pseudospectral approach for computing geodesics," in *ACC*. IEEE, 2017, pp. 1284–1289.
[17] G. Chou, N. Ozay, and D. Berenson, "Model error propagation via learned contraction metrics for safe feedback motion planning of unknown systems," *CoRR*, vol. abs/2104.08695, 2021. [Online]. Available: https://arxiv.org/abs/2104.08695
[18] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *IJRR*, vol. 20, no. 5, pp. 378–400, 2001.
[19] S. Boyd and L. Vandenberghe, *Convex Optimization*, 2004.
[20] T.-W. Weng, H. Zhang, P.-Y. Chen, J. Yi, D. Su, Y. Gao, C.-J. Hsieh, and L. Daniel, "Evaluating the robustness of neural networks: An extreme value theory approach," *ICLR*, 2018.
[21] L. De Haan and A. Ferreira, *Extreme value theory: an introduction*. Springer Science & Business Media, 2007.
[22] M. DeGroot and M. Schervish, *Probability & Statistics*. Pearson, 2013.
[23] P. D. Lax, *Linear Algebra and Its Applications*, 2007.
[24] K. Hauser and Y. Zhou, "Asymptotically optimal planning by feasible kinodynamic planning in a state-cost space," *T-RO*, 2016.
[25] R. Bonalli, A. Cauligi, A. Bylard, and M. Pavone, "Gusto: Guaranteed sequential trajectory optimization via sequential convex programming," in *ICRA*, 2019.
[26] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *IROS*, 2012, pp. 5026–5033.