Dynamic Tube MPC: Learning Tube Dynamics with Massively Parallel Simulation for Robust Safety in Practice

William D. Compton[1], Noel Csomay-Shanklin[1], Cole Johnson[2], Aaron D. Ames[1]

Abstract—Safe navigation of cluttered environments is a critical challenge in robotics. It is typically approached by separating the planning and tracking problems, with planning executed on a reduced order model to generate reference trajectories, and control techniques used to track these trajectories on the full order dynamics. Inevitable tracking error necessitates robustification of the nominal plan to ensure safety; in many cases, this is accomplished via worst-case bounding, which ignores the fact that some trajectories of the planning model may be easier to track than others. In this work, we present a novel method leveraging massively parallel simulation to learn a dynamic tube representation, which characterizes tracking performance as a function of actions taken by the planning model. Planning model trajectories are then optimized such that the dynamic tube lies in the free space, allowing a balance between performance and safety to be traded off in real time. The resulting *Dynamic Tube MPC* is applied to the 3D hopping robot ARCHER, enabling agile and performant navigation of cluttered environments, and safe collision-free traversal of narrow corridors.

I. INTRODUCTION

Safe autonomous navigation of cluttered environments is a challenging problem for robotic systems. Issues including non-convexity of the traversible environment, complexities of a robot's full-order dynamics, and the desire to simultaneously achieve dynamic and safe behaviors during execution motivate hierarchical solutions to this problem. Typical approaches decompose the problem into a *planning layer*, operating on a simplified model to find a collision-free (safe) path through the environment, and a *tracking layer*, which takes in the plan and produces control inputs to track it as accurately as possible on the full order system.

When decomposed in this way, the primary issue of interest becomes certifying how well the tracking controller can track plans, as error between the two systems can cause unsafe behaviors, i.e., collisions, if not accounted for properly. Most commonly, robotic practitioners will simply add a conservative, ad-hoc safety margin to the planning system; however significant work has been performed to develop theoretically grounded, less conservative methods. By setting up a pursuit-evasion game between the tracking and planning models, FaSTrack leverages numerical solutions to the HJB PDE to identify a tracking error bound, which upper bounds the deviation between the two models [1]. For systems of specific structure (for instance, feedback linearizability),
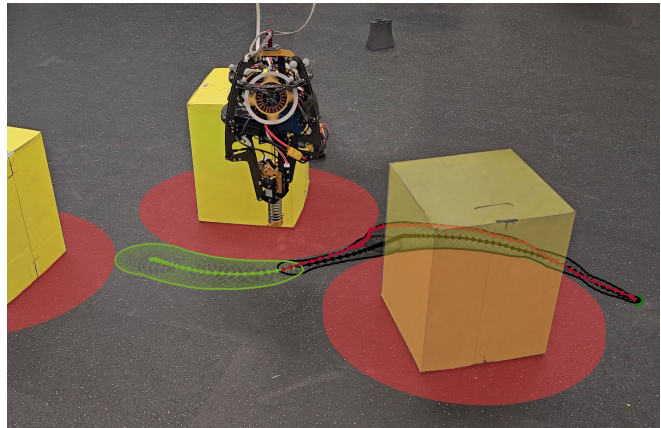
Fig. 1. ARCHER plans a collision-free path through a cluttered environment by jointly optimizing a path on a reduced order model and a dynamic tube, whose dynamics are learned from simulation data. The obstacles are tightly approximated by circles and buffered by the radius of the robot.

ISS Lyapunov functions can be identified, whose level-sets are forward invariant and thus give a tracking error bound [2], and input-to-state safe (ISSf) control barrier functions (CBFs) can be utilized with the reduced order dynamics to ensure safety [3] (see [4], [5] for recent surveys). In the case where a Lyapunov function cannot be identified analytically, a space of functions can be parameterized and searched, for instance via sum-of-squares programming [6] or learning [7], [8]. This problem has also been treated in the language of simulation functions [9], [10], and contraction theory [11].

Once a tracking invariant has been established, the navigation problem is simplified, and now requires finding a path for the planning model which, when buffered by the tracking invariant, is collision free. Tube Model Predictive Control (Tube MPC) solves this problem by minimizing a cost function while ensuring the tube-buffered trajectory lies in the free space [12], [13]. While computationally intensive, Tube MPC has been implemented in real-time to control several hardware systems, including [14], [15]. A notable drawback of Tube MPC formulations is that the fixed, worst case tubes can be quite conservative; to mitigate this, Planning Fast and Slow introduces a meta-planner to select what planning model (and thus tube size) to use depending on the environment [16]. Dynamic Tube MPC offers further flexibility by allowing tube sizes to vary with control decisions, as demonstrated through implementations with sliding mode control [17] and learned tube dynamics [18]. Alternatively, stochastic methods can be used to characterize dynamics uncertainty. Methods either rely on the mean and variance [19], [20], or the tail probabilities [21],

[22], [23] to plan paths which are highly likely to be collision free. Building on [18]'s use of quantile regression for tube learning, our approach significantly increases the available training data, incorporates error history information for more accurate tube prediction, and achieves real-time performance on a complex hardware platform.

A recent trend in robotics has been to leverage massively parallel simulation to learn desired behaviors through significant environmental interaction. Reinforcement learning, at the center of this approach, has seen significant success, including work in drones [24], legged robotics [25], and bipeds [26]. Recent work has even used parallel simulation for online planning [27]. While learning has been used in the past to generate certificates for dynamical systems, including Lyapunov/barrier functions, [7], [28], [29], reachable sets [8], and stabilizing policies, [30], there is an opportunity to extend these learning based certification schemes to significantly more complex and high dimensional environments by leveraging massively parallel simulation.

In this paper, we focus on the critical insight that for many systems, tracking performance varies with the signal being tracked. Very aggressive trajectories are difficult to track, resulting in large error bounds, while conservative trajectories can be tracked more closely. Informed by this concept, we learn a dynamic error tube that associates a planning model trajectory with a corresponding tube, within which the tracking model is expected to remain while following the trajectory. We condition the tube on a history of tracking errors, and demonstrate significant tube accuracy gains with increasing history length. Once the tube dynamics have been learned, we formulate a *Dynamic Tube MPC* problem, which optimizes trajectories of the planning model such that the dynamic tube remains in the free space.

Dynamic Tube MPC is deployed to control the 3D hopping robot ARCHER to safely navigate cluttered environments. We demonstrate problems in which fixed-size tube MPC leads to either infeasibility or highly conservative behavior, and demonstrate that our Dynamic Tube MPC achieves both performance and safety. Fig. 1 shows ARCHER running Dynamic Tube MPC in real time to dynamically and safely navigate a cluttered environment.

II. PRELIMINARIES

In this section, we introduce the mathematical concepts and nomenclature to be used throughout the paper. Discrete time variables will be indexed via subscript (i.e. $\mathbf{x}_k$), and sequences will be indicated via colon notation, $\mathbf{x}_{i:j} = \{\mathbf{x}_i, \mathbf{x}_{i+1}, \dots, \mathbf{x}_{j-1}\}$. We begin by defining the tracking and planning models, which form the foundation of our approach.

A. Tracking and Planning Models

Considering a system with n_x states and m_x inputs, with discrete dynamics governed by:

$$\mathbf{x}_{k+1} = \mathbf{f}_{\mathbf{x}}(\mathbf{x}_k, \mathbf{u}_k) \tag{1}$$

where $\mathbf{x}_k \in \mathcal{X} \triangleq \mathbb{R}^{n_x}$ is the system state, $\mathbf{u}_k \in \mathcal{U} \triangleq \mathbb{R}^{m_x}$ the system input, $\mathbf{f}_{\mathbf{x}} : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$ the tracking dynamics,

and $k \in \mathbb{Z}$ indexes time step. Note that while many robotic systems are naturally described by differential equations, we consider discrete time systems as they will be amenable to both the learning problem and the planning problem. For our purposes, these can be the discretized version of continuous dynamics (for instance, as discretized by a simulator).

To facilitate planning, we also define a planning model, with $n_z < n_x$ states and $m_z < m_x$ inputs,

$$\mathbf{z}_{k+1} = \mathbf{f}_{\mathbf{z}}(\mathbf{z}_k, \mathbf{v}_k) \tag{2}$$

where $\mathbf{z}_k \in \mathcal{Z} \triangleq \mathbb{R}^{n_z}$ is the planning state, $\mathbf{v}_k \in \mathcal{V} \subset \mathbb{R}^{m_z}$ the planning input, and $\mathbf{f}_{\mathbf{z}} : \mathcal{Z} \times \mathcal{V} \to \mathcal{Z}$ the planning dynamics. Additionally, we consider a tracking controller $\mathbf{k} : \mathcal{X} \times \mathcal{Z} \times \mathcal{V} \to \mathcal{U}$, which selects control inputs for the full order system to track the planning system.

The planning dynamics should include all states relevant to the planning problem; for instance, in a collision avoidance problem, the planning model should contain at least the position states, as these are relevant to the constraints. To this end, we define a projection operation $\mathbf{\Pi_z} : \mathcal{X} \to \mathcal{Z}$, which takes the tracking system state and projects it onto the planning model. From this, we can define the planning model error $e : \mathcal{X} \times \mathcal{Z} \to \mathbb{R}_{\geq 0}$:

$$e_k(\mathbf{x}_k, \mathbf{z}_k) = \|\mathbf{z}_k - \mathbf{\Pi_z}(\mathbf{x}_k)\| \tag{3}$$

More general definitions of the error can be taken; ours parameterizes the error as a scalar, and leads to spherical tubes for ease of planning (see Section III-B). Other parameterizations, for instance hypercubes or ellipsoids, lead to different structures of the planning problem.

Finally, we introduce the tube size, $w_k \in \mathbb{R}_{\geq 0}$. A dynamic tube is a sequence of tube sizes $\{w_j\}$; we define the tube to be *correct* when the closed loop execution of the full order model tracking the planning model yields $w_k \geq e_k$, i.e. the tracking system remains with the tube centered about the planning model. Ultimately, we will be concerned with learning tube dynamics, which will predict future tube sizes from a history of relevant data.

B. Nominal MPC for Planning

In the absence of tracking error, we can consider a nominal planning problem. Consider a environment with free space (collision-free subset of the planning model state space) denoted $\mathcal{C} \subset \mathcal{Z}$. For a given initial condition $\mathbf{z}_i$, goal state $\mathbf{z}_g$, and cost functional $J : \mathcal{Z}^{N+1} \times \mathcal{V}^N \to \mathbb{R}_{\geq 0}$, the nominal MPC problem is then given:

$$\underset{\mathbf{V}_{(\cdot)}, \mathbf{Z}_{(\cdot)}}{\text{minimize}} \quad J(\mathbf{z}_{(\cdot)}, \mathbf{v}_{(\cdot)}) \tag{4a}$$

subject to

$$\mathbf{z}_{j+1} = \mathbf{f}_{\mathbf{z}}(\mathbf{z}_j, \mathbf{v}_j) \quad j \in [0, N-1], \tag{4b}$$

$$\mathbf{z}_0 = \mathbf{z}_i, \tag{4c}$$

$$\mathbf{v}_j \in \mathcal{V} \qquad\qquad j \in [0, N-1], \tag{4d}$$

$$\mathbf{z}_j \in \mathcal{C} \qquad\qquad j \in [0, N] \tag{4e}$$

where $N \in \mathbb{Z}_{>0}$ is the length of the planning horizon. Under appropriate selection of the cost function, solving this

problem in closed loop by applying the first input in the solution trajectory and then resolving the MPC problem will result in stability of the closed loop system to the goal [31].

C. Application to ARCHER

The ARCHER 3D hopping robot, shown in Fig. 1, is the primary hardware focus of this paper. Stabilizing controllers for ARCHER have been previously studied [32], [33]. The system contains $n_x = 16$ states $\mathbf{x} = [\mathbf{p}^\top \; \mathbf{q}^\top \; \mathbf{v}^\top \; \boldsymbol{\omega}^\top \; \dot{\boldsymbol{\theta}}^\top]^\top$, where $\mathbf{p} \in \mathbb{R}^3$ is the global position, $\mathbf{q} \in \mathbb{S}^3$ a unit quaternion defining orientation, $\mathbf{v} \in \mathbb{R}^3$ the linear velocity, $\boldsymbol{\omega} \in \mathfrak{s}^3$ the angular velocity, and $\dot{\boldsymbol{\theta}} \in \mathbb{R}^3$ the velocities of the flywheels. The system takes $m_x = 3$ inputs, the torques applied to each of the flywheels. The foot subsystem is autonomous, and not considered in the design of our controllers; the foot spring is compressed to a preset distance during the flight phase, and released during the ground phase to maintain approximately constant hop height. More details regarding the hardware can be found here [33]. The system dynamics $\mathbf{f_x}$ will simply be steps of the robot in a given simulator.

In this work we will plan trajectories for the hopper using a two dimensional single integrator with $\mathbf{z}_k \in \mathbb{R}^2$ the positions and $\mathbf{v}_k \in \mathcal{V} \subset \mathbb{R}^2$ the velocities, which will be bounded to lie in a symmetric range, $-\bar{v} \le \mathbf{v}_k \le \bar{v}$, where $\bar{v} \in \mathbb{R}_{\ge 0}$ is the velocity bound. The projection operator will simply return the x and y positions of the hopper, $\mathbf{\Pi_z}(\mathbf{x}_k) = [p_{x,k} \; p_{y,k}]^\top$.

The tracking controller used for ARCHER in this work will be a Raibert Heuristic [34]. Given a set of gains $k_p, k_d, k_f \ge 0$, and clipping parameters, $c_p, c_v, c_f, c_a \ge 0$ we compute the desired roll, ϕ_d, and pitch, γ_d, as:

$$\mathbf{e}_p = \mathrm{clip}\,(\mathbf{z} - \mathbf{\Pi_z}(\mathbf{x}), c_p) \tag{5a}$$

$$\mathbf{e}_v = \mathrm{clip}\,(-\mathbf{\Pi_v}(\mathbf{x}), c_v) \tag{5b}$$

$$\mathbf{e}_f = \mathrm{clip}\,(\mathbf{v}, c_f) \tag{5c}$$

$$[\phi_d, \; -\gamma_d]^\top = \mathrm{clip}\,(-k_p\mathbf{e}_p - k_d\mathbf{e}_v + k_f\mathbf{e}_f, c_a) \tag{5d}$$

where $\mathbf{\Pi_v} : \mathcal{X} \to \mathbb{R}^2$ projects the hopper state to its x and y velocity coordinates. The clipping operations prevent large errors from requesting too large of orientations, which lead to falling. Finally, geometrically consistent PD control is used to track the desired orientation as outlined in [33].

III. METHODS

Our approach consists of two primary components: learning a dynamic representation of an error tube that characterizes the how tracking error bounds depend upon trajectories of the planning model, and employing Dynamic Tube MPC to plan collision-free paths within these bounds. This section details each of these components.

A. Learning Tube Dynamics

The primary goal of this work is to characterize how the actions taken by the planning model effect the size of the tracking invariant established by the tracking controller. To this end, consider the following problem. Take any feasible trajectory of the planning model, covering a length of at least $H + N$ time steps, where $H \in \mathbb{Z}_{\ge 0}$ is the history considered

by the model, and $N \in \mathbb{Z}_{>0}$ the horizon for the planning problem). Assume the full order system has been tracking the planning model for at least H time steps. Then at the current time k, we can compute the a history of tracking error as $e_{k-H:k}$ by applying (3) at each of the time steps. Now, given the state and input trajectory of the planning model, $\{\mathbf{z}_j\}_{j=k}^{k+N}, \{\mathbf{v}_j\}_{j=k}^{k+N}$, we would like to predict the tube, $\{w_j\}_{j=k}^{k+N}$, such that $w_j \ge e_j$ at each point along the trajectory when executing the tracking controller in closed loop. We consider two methods of prediction:

- One-Shot Tube Dynamics:

$$w_{k:k+N} = f_{w,os}(e_{k-H:k}, \mathbf{z}_{k-H:k+N}, \mathbf{v}_{k-H:k+N}) \tag{6}$$

  This predicts the tube size over the horizon as a single function of the error history and planning trajectory.

- Recursive Tube Dynamics:

$$w_{j+1} = f_{w,rec}(\tilde{e}_{j-H:j}, \mathbf{z}_{j-H:j}, \mathbf{v}_{j-H,j}, j) \tag{7}$$

$$\tilde{e}_{j-H:j} = \begin{cases} [e_{j-H:k} \; w_{k:k+H-j}] & j < k + H \\ w_{j-H:j} & j \ge k + H \end{cases}$$

  This is applied recursively from $j = k$ to $j = k + N$, predicting the next tube size as a function of the history of tube sizes and planning trajectory. Critically, for indices of $\tilde{e}_{j-H:j}$ up to k, we will use the error history, while after k we use the tube prediction. For this reason, j is also passed to the function, as this will help distinguish how much of $\tilde{e}$ is error history as opposed to tube predictions.

To learn the tube dynamics, we utilize massively parallel simulation to collect a large dataset of the form $\mathcal{D} = \{\mathbf{z}_{0:\bar{N}+1}, \mathbf{v}_{0:\bar{N}}, \mathbf{\Pi_z}(\mathbf{x}_{0:\bar{N}+1})\}$, with $\bar{N}$ the length of each trajectory in the dataset. The inputs to the planning model are randomly generated[1] and the planning model and tracking model initial conditions are sampled from an initial condition sets, $\mathbf{z}_0 \sim \mathcal{Z}_0 \subset \mathcal{Z}$, and $\mathbf{x}_0 \sim \mathcal{X}_0(\mathbf{z}_0) \subset \mathcal{X}$.

The tube dynamics are represented by a neural network, $f_w^{\boldsymbol{\theta}}$, with parameters $\boldsymbol{\theta}$. To train the model, as in [18], inspired by quantile regression [21], we use a check loss function:

$$\mathbf{w}(\boldsymbol{\theta}) = f_w^{\boldsymbol{\theta}}(\mathbf{e}_{k-H:k}, \mathbf{z}_{k-H:k+N}, \mathbf{v}_{k-H:k+N}) \tag{8a}$$

$$\mathbf{e} = \|\mathbf{z}_{k:k+N} - \mathbf{\Pi_z}(\mathbf{x}_{k:k+N})\| \tag{8b}$$

$$\mathbf{r}(\boldsymbol{\theta}) = \begin{cases} \alpha(\mathbf{w}(\boldsymbol{\theta}) - \mathbf{e}) & \mathbf{w}(\boldsymbol{\theta}) \ge \mathbf{e} \\ (1-\alpha)(\mathbf{e} - \mathbf{w}(\boldsymbol{\theta})) & \mathbf{w}(\boldsymbol{\theta}) < \mathbf{e} \end{cases} \tag{8c}$$

$$\mathcal{L}(\boldsymbol{\theta}) = \mathrm{Huber}(\|\mathbf{r}(\boldsymbol{\theta})\|_1) \tag{8d}$$

Given an index k, we evaluate the loss over the horizon length N on which the model predicts. (8a) evaluates the tube model (either recursive or one-shot) over the entire horizon. (8b) then evaluates the tracking error over the horizon. (8c) implements the check loss, where the $\mathbf{w} \ge \mathbf{e}$ condition holds when the tube prediction is correct, and incentives the tube to be tight around the error (by decreasing tube size). The

---

[1] The distribution of trajectories matters when deploying the tube model in an MPC problem, as discussed in Section V-A. See code [35] for implementation details.
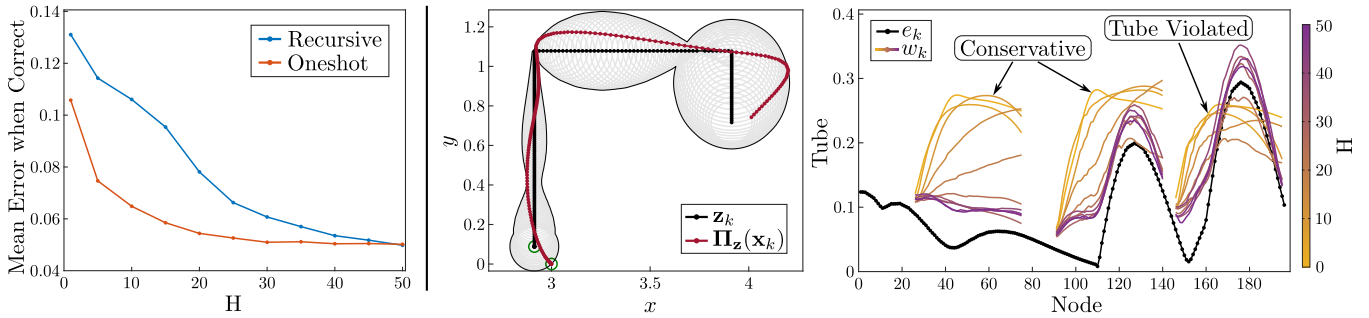
Fig. 2. Analysis of the impact of history, H, on tube prediction accuracy; tube dynamics are trained on the hopper dynamics for several values of H. (Left) Evaluation on a 20% holdout from training data, Mean Error when Correct is plotted against history length. (Middle) a portion of a square trajectory is executed in IsaacGym on ARCHER. (Right) The error signal, e_k, is plotted in black, with the tube predictions, w_k, plotted at three points in time, $k = 25, 75, 145$ over a horizon of $N = 50$ nodes (right pane). We see short horizon models being both overly conservative and violating the error tube.

$\mathbf{w} < \mathbf{e}$ conditional holds when the tube is incorrect, and incentivizes increasing the tube size. Since the output of (8c) is a vector (of length N), we apply a Huber loss [36] to the L1 penalty for a smooth penalty landscape.

The choice of the parameter α is directly related to the level of robustness achieved by the tube. Previous work [18] has demonstrated that minimizing the check loss (8c) corresponds to finding at tube which is an α-quantile representation of the error dynamics; the tube is expected to be correct an α proportion of the time.

B. Dynamic Tube MPC

Given a dynamic tube model $f_w^{\boldsymbol{\theta}}$, which maps an error history and planning model trajectory to a tube prediction (either via one-shot or recursive application), we can formulate the Dynamic Tube MPC problem as a variant of (4):

$$\underset{\mathbf{V}_{(\cdot)}, \mathbf{Z}_{(\cdot)}}{\text{minimize}} \quad J(\mathbf{z}_{(\cdot)}, \mathbf{v}_{(\cdot)}) \tag{9a}$$

subject to

$$\mathbf{z}_{j+1} = \mathbf{f_z}(\mathbf{z}_j, \mathbf{v}_j) \qquad j \in [0, N-1], \tag{9b}$$

$$w_{0:N} = f_w^{\boldsymbol{\theta}}(e_{\text{-}H:0}, \mathbf{z}_{\text{-}H:N}, \mathbf{v}_{\text{-}H:N}), \tag{9c}$$

$$\mathbf{z}_0 = \mathbf{z}_i, \tag{9d}$$

$$\mathbf{v}_j \in \mathcal{V} \qquad j \in [0, N-1], \tag{9e}$$

$$B_{w_j}(\mathbf{z}_j) \in \mathcal{C} \qquad j \in [0, N] \tag{9f}$$

where $B_r(\mathbf{z})$ is a ball of radius r centered at $\mathbf{z}$. The tube dynamics are enforced via (9c), and safety is enforced by (9f). Dynamic Tube MPC aims to minimize the given cost function while ensuring that the planned trajectory, when buffered by the error tube, remains within the free space.

In this work, we will consider circular obstacles, which can be encoded as a single constraint, such that the Dynamic Tube MPC problem can be formulated as a nonlinear program (NLP). Given the i'th obstacle, centered at $\mathbf{z}_{c,i}$ with radius r_i, the safety constraint (9f) can be expressed:

$$(\mathbf{z}_j - \mathbf{z}_{c,i})^\top (\mathbf{z}_j - \mathbf{z}_{c,i}) - (w_j + r_i)^2 \geq 0 \tag{10}$$

For closed loop control, we apply the first input in the Dynamic Tube MPC solution, then recompute the solution, with the next initial condition as the current state of the planning model. Note that this differs from some implementations of Tube MPC, where the projection of the tracking system must

be within the tube around the initial condition; we avoid this constraint, as it significantly complicates the interpretation and implementation of the error history, $e_{H:0}$.

IV. ANALYSIS

A critical differentiating factor between our method and previous ones, i.e. [18], is the incorporation of an error history, $\tilde{e}$ in (7), in the tube prediction. We analyse the impacts of the this error history on the size of the learned tube. Additionally, we highlight the improvements of Dynamic Tube MPC over classical tube MPC approaches.

A. Effect of Error History on Tube Size

Including a longer error history leads to reductions in the size of the learned tube. To demonstrate this, we train tube dynamics models with histories ranging from $H = 1$ to $H = 50$, with $\alpha = 0.9$, and predict on a horizon of length $N = 50$. On a 20% held out validation set, all models predict tubes which are correct at least 90% of the time, matching expectations from the check loss (8c). The primary metric used to quantify the size of a tube is Mean Error When Correct (MEC); given the errors, $e_{0:N}$, and tube predictions, $w_{0:N}$, the tube is correct on an index set $I_c = \{i \in 0, \ldots, N \mid w_i \geq e_i\}$. The MEC is then

$$\text{MEC} = \frac{1}{|I_c|} \sum_{i \in I_c} w_i - e_i \tag{11}$$

Note each element of this sum is non-negative, as we sum only over indices where the tube is correct. When the MEC is small, the tube tightly approximates the actual error, while when it is large, the tube over-approximates the error significantly. In Fig. 2 (left) we see that increasing H leads to reduction in the MEC. For large H, these returns diminish. Additionally, we note that the one-shot models tend to out-preform the recursive models at each history length, with this gap closing as H increases.

To further demonstrate this idea, we consider a case where the planning model executes part of a square, as shown in Fig. 2 (middle) by the black line. The hopper, when tracking this trajectory in simulation, incurs error, as shown by the magenta line. In Fig. 2 (right), we see three time steps, where the recursive tube predictions are evaluated, for various history lengths of the model. As an overall trend,
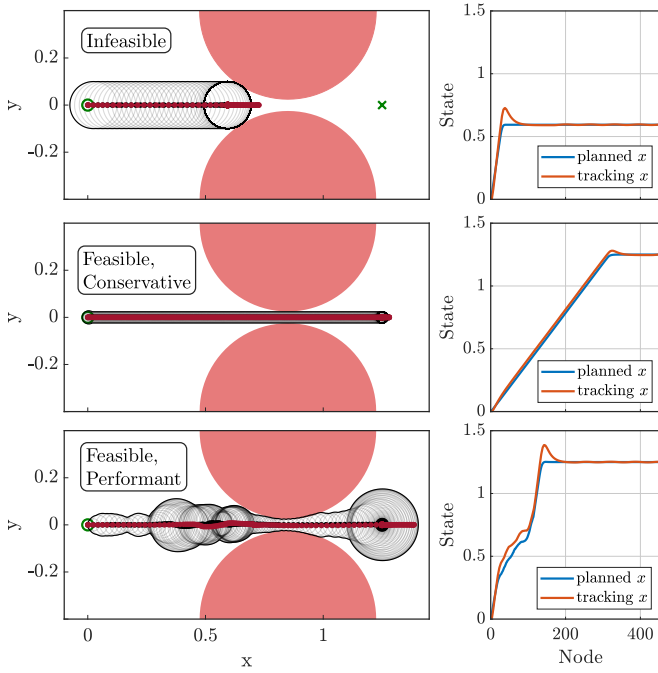
Fig. 3. Three Tube MPC variants, on a problem where the hopper must traverse a narrow gap. (Top) Tube MPC, where the tube size is fixed to be the 90% quantile in a dataset collected with $\bar{v} = 0.2$m/s. (Middle) Tube MPC, where the tube size is fixed to the 90% quantile, in a dataset collected with $\bar{v} = 0.04$m/s. (Bottom) Dynamic Tube MPC using recursive tube dynamics, which simultaneously achieves performance and safety.

we see that the short horizon models tend to be significantly more conservative than the long history models; additionally, the long history models seem to capture the shape of the error trajectory much more precisely. In the third set of tube predictions, we see a case where the short-horizon models under-approximate the resulting error, while the long horizon models retain solid performance.

The trends in Fig. 2 show a strong incentive to use longer error histories when predicting tube size, ideally $H \geq 25$ for this setup on ARCHER; however, performance levels off beyond a certain point. The incorporation of the error history is critical, as it allows the model to discriminate between states which have the same instantaneous error (for instance where the error may be increasing or decreasing, leading to very different tubes). We can draw parallels to ideas from filtering; the history of errors and planning trajectories gives the neural network the necessary information to learn a filter of the salient full order model states, and use these to more accurately predict tube sizes over the horizon. Beyond a certain point, including a longer history does not improve the quality of this implicit filter.

B. Dynamic Tube MPC

When employed in the Dynamic Tube MPC, the learned tube representation allows the optimizer to trade off aggressive behaviors and conservative behaviors when necessary to ensure safety. Fig. 3 demonstrates a critical benefit of Dynamic Tube MPC over a fixed size Tube MPC. A problem is set up for the hopper to traverse a narrow gap; the top and middle panes demonstrate solution of this problem with

a fixed tube size, while the bottom pane demonstrates the effectiveness of a recursive dynamic tube, trained on a dataset with $\bar{v} = 0.2$m/s. For the top plot, the tube size is the 90% quantile of the errors in the dataset used to train the tube dynamics (we do this as computation of a true robust invariant is quite difficult, and the quantile allows better comparison to the learned tube, which also uses quantile bounding). For this tube size, the planning model cannot reach the goal. In order to reach the goal with a fixed tube size, we are forced to limit the planning model to $\bar{v} = 0.04$m/s; collecting a dataset with these tightened input bounds, the 90% quantile bound fixed tube allows a very conservative solution to the problem, solving in around 400 nodes (40 seconds). The dynamic tube MPC travels quite quickly when away from the obstacles (note the slope of the state plot), while slowing to a speed similar to the conservative solution while between the obstacles. This allows a much more aggressive trajectory, solving the problem in only 200 Nodes (20 seconds). The ability to dynamically vary the aggressiveness of the planning model allows the system to balance performance with safety, and generates robust and performant behaviors difficult to achieve via other methods.

V. APPLICATION OF DYNAMIC TUBE MPC TO ARCHER

We deploy our Dynamic Tube MPC method on the ARCHER platform to generate collision free trajectories in cluttered environments, in real time.

A. Learning Tube Dynamics

We follow the tube learning process outlined in Section III-A, simulating the hopper with 8192 units in parallel in IsaacGym [37]. We collect a dataset consisting of 409,600 trajectories, each of length 20 seconds, under random inputs to the planning model. To avoid excessive distribution shift when deploying the learned tube in MPC, we did significant work to shape the random inputs such that the planning trajectories share characteristics with the solutions from MPC; this is detailed in the code [35]. Additionally, to facilitate transfer onto the hardware system, we implement domain randomization, inspired by results in RL [25], [38]. The ARCHER platform's tracking performance is extremely sensitive to the location of the center of mass, so we randomize this parameter within 5mm, as well as several other physical parameters of the robot (stiffness, damping, inertia, ect.). The domain randomization makes the tubes more conservative, improving the transfer to hardware. We choose a history of $H = 25$, a horizon of $N = 25$, and an error quantile of $\alpha = 0.9$. Training is conducted in PyTorch [39], using neural networks with two layers of 128 nodes, and Softplus ($\beta = 5$) activations, as these result in smooth networks which are easier than ReLU to optimize through in MPC. We choose a recursive tube representation; while the one-shot tube representation was more accurate, empirically the recursive tube yielded much smoother tube predictions, and was easier to optimize through, leading to more desirable closed loop behavior in the Dynamic Tube MPC. When training the recursive models, we stop gradients on the w
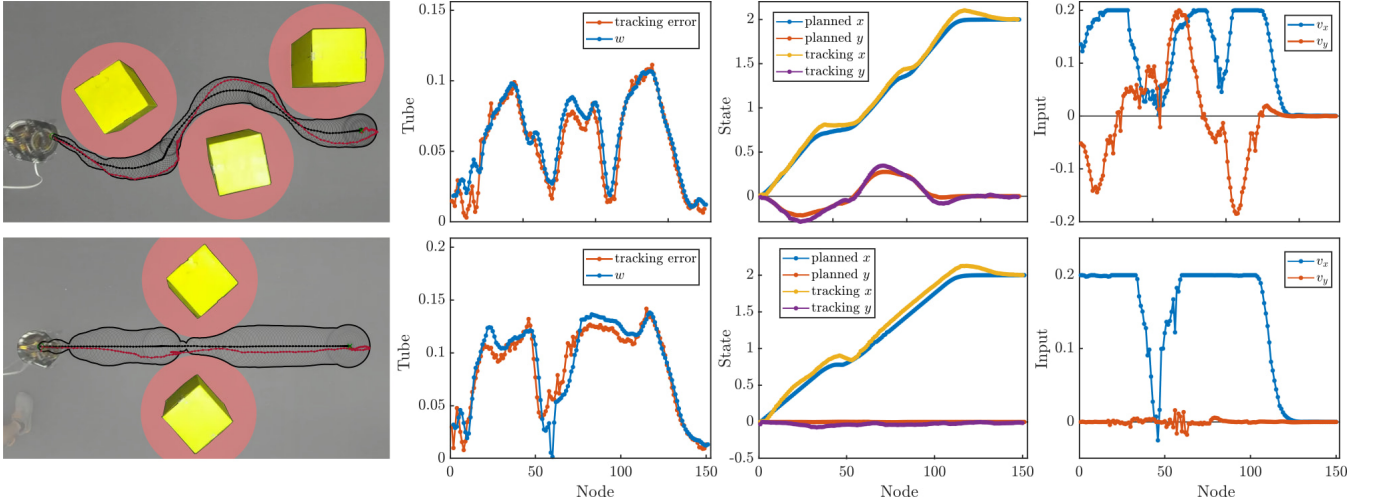
Fig. 4. Implementation of Dynamic Tube MPC on the ARCHER platform. Plotted are closed loop planned trajectory and tube (white), along with tracking performance (red), along with tube, state, and input trajectories over time. Note specifically how the planning model slows down when in narrow corridors between obstacles to improve tracking.

components of $\tilde{e}$ from (7), as these terms depend on $\boldsymbol{\theta}$; this stabilizes training by localizing gradient information.

B. Dynamic Tube MPC

Due to optimizing through the neural network constraint (9c), some extra effort is required to obtain smooth solutions at real-time rates to the dynamic tube MPC problem. For our nominal problem (4), we choose quadratic costs on the state and input. Defining the distance to the goal as $\mathbf{e}_k = \mathbf{z}_k - \mathbf{z}_g$, with $q, r, q_f > 0$, the cost functional is:

$$J(\mathbf{z}_{(\cdot)}, \mathbf{v}_{(\cdot)}) = q_f \mathbf{e}_k^\top \mathbf{e}_k + \sum_{j=0}^{N-1} q \mathbf{e}_k^\top \mathbf{e}_k + r \mathbf{v}_k^\top \mathbf{v}_k \quad (12)$$

In order to improve solutions to the Dynamic Tube MPC problem, (9), we first solve the nominal problem. Let $\bar{\mathbf{z}}_k, \bar{\mathbf{v}}_k$ be the solution to this problem; then the cost function for the full problem is defined as distance to the nominal problem solution. Additionally, to incentivize smooth inputs, we place a penalty on the input rate; putting these together, we have:

$$\mathbf{e}_{\mathbf{z},k} = \mathbf{z}_k - \bar{\mathbf{z}}_k \quad (13a)$$

$$\mathbf{e}_{\mathbf{v},k} = \mathbf{v}_k - \bar{\mathbf{v}}_k \quad (13b)$$

$$\boldsymbol{\delta}_{\mathbf{v},k} = \mathbf{v}_k - \mathbf{v}_{k-1} \quad (13c)$$

$$J(\mathbf{z}_{(\cdot)}, \mathbf{v}_{(\cdot)}) = \sum_{k=0}^{N-1} q \mathbf{e}_{\mathbf{z},k}^\top \mathbf{e}_{\mathbf{z},k} + r \mathbf{e}_{\mathbf{v},k}^\top \mathbf{e}_{\mathbf{v},k} \quad (13d)$$

$$+ q_f \mathbf{e}_{\mathbf{z},N}^\top \mathbf{e}_{\mathbf{z},N} + \sum_{k=1}^{N-1} r_r \boldsymbol{\delta}_{\mathbf{v},k}^\top \boldsymbol{\delta}_{\mathbf{v},k}$$

where (13a) and (13b) compute deviation from the nominal solution, and (13c) penalizes the input rate.

The MPC problem is solved in CasADi [40] via SNOPT [41], with the neural network constraint implemented via L4CasADi [42], [43]. We successfully deploy our algorithm on the ARCHER hardware in a number of experimental trials. To achieve real time rates, we allow the first Dynamic Tube MPC solve to run to convergence, and then

limit SNOPT to four SQP iterations per solve. Under these conditions, the Dynamic Tube MPC solves at 10Hz, which matches the $dt = 0.1$ of the planning model.

C. Results

Fig. 4 shows two experimental trials. In the bottom experiment, ARCHER is asked to navigate a straight line through a tight gap; we see the planning model operate at maximum speed, then slow to near zero speed which traversing the narrow gap, to ensure precise tracking, and finally speeding up again once free of the obstacles. In the top, similar behavior is observed, where high speeds are used when the robot is not confined between obstacles, and speeds drop in the narrow corridors. Critically, none of these maneuvers would have been possible with a classic Robust Tube MPC implementation, without setting the velocity limit $\bar{v}$ on the planning model very low.

D. Limitations

The main limitation of this work is that the training data and evaluation data come from different distributions, as training data is random. We carefully craft our randomized training trajectories to minimize effects of this distribution shift. This could be dramatically improved by solving planning problems in the training data collection, similar to [18], using a parallelizable planning method, such as [27].

VI. CONCLUSION

We proposed a method leveraging massively parallel simulation to learn a dynamic representation of a tracking error tube. We demonstrate that including a tracking error history in the tube dynamics dramatically improves prediction accuracy. We formulated a dynamic tube MPC, which optimized trajectories of the planning model such that the resulting dynamic tube lies in the free space, allowing real-time trade-offs between performant and conservative behaviors to ensure safety. We deploy this method on the ARCHER platform, achieving agile and collision free navigation of cluttered environments.

REFERENCES

[1] M. Chen, S. L. Herbert, H. Hu, Y. Pu, J. F. Fisac, S. Bansal, S. Han, and C. J. Tomlin, "Fastrack:a modular framework for real-time motion planning and guaranteed safe tracking," *IEEE Transactions on Automatic Control*, vol. 66, pp. 5861–5876, 12 2021.

[2] N. Csomay-Shanklin, A. J. Taylor, U. Rosolia, and A. D. Ames, "Multi-rate planning and control of uncertain nonlinear systems: Model predictive control and control lyapunov functions," *Proceedings of the IEEE Control and Decisions Conference*, vol. 2022-December, pp. 3732–3739, 2022.

[3] T. G. Molnar, R. K. Cosner, A. W. Singletary, W. Ubellacker, and A. D. Ames, "Model-free safety-critical control for robotic systems," *IEEE Robotics Automation Letters*, vol. 7, no. 2, p. 944–951, Apr. 2022.

[4] M. H. Cohen, T. G. Molnar, and A. D. Ames, "Safety-critical control for autonomous systems: Control barrier functions via reduced-order models," *Annual Reviews in Control*, vol. 57, p. 100947, 2024.

[5] K. P. Wabersich, A. J. Taylor, J. J. Choi, K. Sreenath, C. J. Tomlin, A. D. Ames, and M. N. Zeilinger, "Data-driven safety filters: Hamilton-jacobi reachability, control barrier functions, and predictive methods for uncertain systems," *IEEE Control Systems Magazine*, vol. 43, no. 5, pp. 137–177, 2023.

[6] M. M. Tobenkin, I. R. Manchester, and R. Tedrake, "Invariant funnels around trajectories using sum-of-squares programming," *IFAC Proceedings Volumes*, vol. 44, pp. 9218–9223, 10 2010.

[7] C. Dawson, Z. Qin, S. Gao, and C. Fan, "Safe nonlinear control using robust neural lyapunov-barrier functions," in *Conference on Robot Learning*. PMLR, 2022, pp. 1724–1735.

[8] S. Bansal and C. J. Tomlin, "Deepreach: A deep learning approach to high-dimensional reachability," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2021-May, pp. 1817–1824, 11 2020.

[9] A. Girard and G. J. Pappas, "Hierarchical control system design using approximate simulation," *Automatica*, vol. 45, pp. 566–571, 2 2009.

[10] V. Kurtz, P. M. Wensing, and H. Lin, "Robust approximate simulation for hierarchical control of linear systems under disturbances," *Proceedings of the American Control Conference*, vol. 2020-July, pp. 5352–5357, 7 2020.

[11] P. Zhao, A. Lakshmanan, K. Ackerman, A. Gahlawat, M. Pavone, and N. Hovakimyan, "Tube-certified trajectory tracking for nonlinear systems with robust control contraction metrics," *IEEE Robotics and Automation Letters*, vol. 7, pp. 5528–5535, 9 2021.

[12] W. Langson, I. Chryssochoos, S. V. Raković, and D. Q. Mayne, "Robust model predictive control using tubes," *Automatica*, vol. 40, pp. 125–133, 1 2004.

[13] J. Sieber, S. Bennani, and M. N. Zeilinger, "A system level approach to tube-based model predictive control," *IEEE Control Systems Letters*, vol. 6, pp. 776–781, 3 2021.

[14] H.-T. Do and I. Prodan, "Tube MPC via flatness for multicopter trajectory tracking," in *2024 European Control Conference*, Stockholm, Sweden, Jun. 2024.

[15] N. Michel, S. Bertrand, S. Olaru, G. Valmorbida, and D. Dumur., "Design and flight experiments of a tube-based model predictive controller for the ar.drone 2.0 quadrotor," *IFAC-PapersOnLine*, vol. 52, no. 22, pp. 112–117, 2019, 1st IFAC Workshop on Robot Control WROCO 2019.

[16] D. Fridovich-Keil, S. L. Herbert, J. F. Fisac, S. Deglurkar, and C. J. Tomlin, "Planning, fast and slow: A framework for adaptive real-time safe trajectory planning," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 387–394, 10 2017.

[17] B. T. Lopez, J.-J. E. Slotine, and J. P. How, "Dynamic tube mpc for nonlinear systems," in *2019 American Control Conference (ACC)*. IEEE, 2019, pp. 1655–1662.

[18] D. D. Fan, A. akbar Agha-mohammadi, and E. A. Theodorou, "Deep learning tubes for tube mpc," *Robotics: Science and Systems*, 2020.

[19] A. Hakobyan, G. C. Kim, and I. Yang, "Risk-aware motion planning and control using cvar-constrained optimization," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3924–3931, 2019.

[20] R. K. Cosner, I. Sadalski, J. K. Woo, P. Culbertson, and A. D. Ames, "Generative modeling of residuals for real-time risk-sensitive safety with discrete-time control barrier functions," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. i–viii.

[21] R. Koenker and G. Bassett Jr, "Regression quantiles," *Econometrica: journal of the Econometric Society*, pp. 33–50, 1978.

[22] E. Bradford, L. Imsland, and E. A. D. Rio-Chanona, "Nonlinear model predictive control with explicit back-offs for gaussian process state space models," *Proceedings of the IEEE Conference on Decision and Control*, vol. 2019-December, pp. 4747–4754, 12 2019.

[23] M. Bujarbaruah, X. Zhang, M. Tanaskovic, and F. Borrelli, "Adaptive mpc under time varying uncertainty: Robust and stochastic," *Transactions on Automatic Control*, 9 2019.

[24] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun, and D. Scaramuzza, "Champion-level drone racing using deep reinforcement learning," *Nature 2023 620:7976*, vol. 620, pp. 982–987, 8 2023.

[25] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to walk in minutes using massively parallel deep reinforcement learning," *Proceedings of Machine Learning Research*, vol. 164, pp. 91–100, 9 2021.

[26] Q. Liao, B. Zhang, X. Huang, X. Huang, Z. Li, and K. Sreenath, "Berkeley humanoid: A research platform for learning-based control," *arXiv preprint arXiv:2407.21781*, 2024.

[27] C. Pezzato, C. Salmi, M. Spahn, E. Trevisan, J. Alonso-Mora, and C. H. Corbato, "Sampling-based model predictive control leveraging parallelizable physics simulations," *arXiv preprint arXiv:2307.09105*, 2023.

[28] Y.-C. Chang, N. Roohi, and S. Gao, "Neural lyapunov control," *Advances in neural information processing systems*, vol. 32, 2019.

[29] F. B. Mathiesen, S. C. Calvert, and L. Laurenti, "Safety certification for stochastic systems via neural barrier functions," *IEEE Control Systems Letters*, vol. 7, pp. 973–978, 2023.

[30] W. Compton, I. D. J. Rodriguez, N. Csomay-Shanklin, Y. Yue, and A. D. Ames, "Constructive nonlinear control of underactuated systems via zero dynamics policies," *Control and Decisions Conference*, 2024.

[31] G. Williams, B. Goldfain, P. Drews, K. Saigol, J. M. Rehg, and E. A. Theodorou, "Robust sampling based model predictive control with sparse objective information," *Systems & Control Letters*, 2001.

[32] N. Csomay-Shanklin, W. D. Compton, I. D. J. Rodriguez, E. R. Ambrose, Y. Yue, and A. D. Ames, "Robust agility via learned zero dynamics policies," *arXiv preprint arXiv:2409.06125*, 2024.

[33] N. Csomay-Shanklin, V. D. Dorobantu, and A. D. Ames, "Nonlinear model predictive control of a 3d hopping robot: Leveraging lie group integrators for dynamically stable behaviors," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2023-May, pp. 12 106–12 112, 2023.

[34] M. H. Raibert, H. B. Brown Jr, and M. Chepponis, "Experiments in balance with a 3d one-legged hopping machine," *The International Journal of Robotics Research*, vol. 3, no. 2, pp. 75–92, 1984.

[35] "Learing tubes mpc code," 2024. [Online]. Available: https://github.com/wdc3iii/LearningTubesMPC

[36] P. J. Huber, "Robust estimation of a location parameter," in *Breakthroughs in statistics: Methodology and distribution*. Springer, 1992, pp. 492–518.

[37] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, and G. State, "Isaac gym: High performance gpu based physics simu-lation for robot learning," *Neural Information Processing Systems*, 2021.

[38] Z. Li, X. B. Peng, P. Abbeel, S. Levine, G. Berseth, and K. Sreenath, "Reinforcement learning for versatile, dynamic, and robust bipedal locomotion control," *arXiv preprint arXiv:2401.16889*, 2024.

[39] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," *Neural Information Processing*, 2017.

[40] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.

[41] P. E. Gill, W. Murray, and M. A. Saunders, "Snopt: An sqp algorithm for large-scale constrained optimization," *SIAM review*, vol. 47, no. 1, pp. 99–131, 2005.

[42] T. Salzmann, E. Kaufmann, J. Arrizabalaga, M. Pavone, D. Scaramuzza, and M. Ryll, "Real-time neural-mpc: Deep learning model predictive control for quadrotors and agile robotic platforms," *IEEE Robotics and Automation Letters*, 2023.

[43] T. Salzmann, J. Arrizabalaga, J. Andersson, M. Pavone, and M. Ryll, "Learning for casadi: Data-driven models in numerical optimization," in *Learning for Dynamics and Control Conference (L4DC)*, 2024.