PING: A Physics-Informed Neuro-Symbolic Generator for Continuous-Time Planning

Mahyar Jahani-nasab, Hamid Rezatofighi, Mor Vered, Buser Say

Monash University

{mahyar.jahaninasab, hamid.rezatofighi, mor.vered, buser.say}@monash.edu

Abstract

We present PING (Physics-Informed Neuro-Symbolic Generator), a novel, continuous planning framework that leverages untrained neural networks as generative function approximators to synthesize high-fidelity trajectory candidates without data or training. Departing from conventional hybrid planners that oscillate between discrete search and numerical optimization, PING operates natively in continuous function-valued action spaces, embedding symbolic constraints directly into the generation process. We introduce a generative mechanism where neural networks produce function-space candidates that are structurally guaranteed to satisfy boundary conditions using symbolic rules, thereby circumventing discretization artifacts and the local minima traps inherent to gradient-based trajectory optimization. To ensure feasibility, a rigorous verification engine exploits automatic differentiation to validate candidates against domain-specific differential equations and manifold constraints. This architecture enables a dual-mode strategy: a primary generation phase for rapid solution synthesis, backed by an iterative refinement mechanism when validation fails. By decoupling generation from optimization, PING provides a training-free framework that can be instantiated for different domains via domain-specific dynamics and constraints. Empirical evaluation across navigation, reservoir control, and HVAC domains demonstrates highly efficient runtime performance and broader coverage, with planning latencies reduced to seconds through massive parallel verification of thousands of symbolic candidates.

Code — https://github.com/mahyar-jahaninasab/PING

Introduction

Planning, a fundamental challenge in robotics and control theory, involves generating dynamically feasible and safe trajectories under a wide range of operational constraints and environmental conditions (Testouri, Elghazaly, and Frank 2023; Vu, Migimatsu, and Bohg 2024). The evolution of planning algorithms has been marked by a persistent trade-off between computational tractability and solution quality. Early sampling-based methods, such as Rapidly-exploring Random Trees (RRT), suffer from slow convergence and the curse of dimensionality, making them difficult to scale to high-dimensional spaces (Chen and Wang 2022; Nasir et al. 2013). Their iterative, node-by-node exploration is computationally intensive and relies on spatial and temporal discretization, introducing artifacts that prevent the full exploitation of the system's underlying differential constraints (Zhang and Li 2024; Elango et al. 2024). As a result, these discrete approaches are fundamentally mismatched to continuous domains: they cannot directly encode or leverage the smooth structure of the system dynamics, failing to reason over the full continuum of states and actions that the system can realize.

In response to these limitations, continuous planning approaches emerged, leveraging optimal control theory to generate smooth, executable motions. Methods integrating Control Barrier Functions (CBFs) and Control Lyapunov Functions (CLFs) provide formal guarantees on safety and stability by embedding these properties directly into the control law (Mori, Mori, and Kuroe 1997; Ames et al. 2019). Similarly, Model Predictive Control (MPC) frameworks offer a formulation for optimal trajectory generation (Mittal et al. 2020). However, when applied to offline planning to generate complete trajectories at once, solving the exact, nonlinear optimal control problem over a full temporal horizon becomes highly computationally demanding. Unlike parallelizable sampling methods, relying on classical optimization techniques to resolve the entire sequence of states translates into a large-scale nonlinear programming problem. This scales poorly with horizon length and system dimensionality, creating a significant computational bottleneck for fast offline planning, even when utilizing state-of-the-art solvers (Cimini et al. 2021; Mohanty et al. 2022).

Recent attempts to narrow this speed–accuracy gap have turned to machine learning (Jiménez et al. 2012). New approaches, such as physics-informed neural networks, increase solution quality by embedding differential equations and stability conditions directly into the training loss (Kokolakis et al. 2025; Lai et al. 2025). These new approaches reframe planning as prediction even though the inference task is orders of magnitude faster than online optimization (Zhou et al. 2024). However, despite their rapid inference capabilities, these methods remain constrained by their strict reliance on extensive offline training and prior data generation. The network approximates a solution map, but minimizing a loss provides no guarantee that the re-

sulting trajectories satisfy hard constraints reliably (Xu and Pan 2024). To bridge the gap between fast prediction and continuous planning, we introduce PING (Physics-Informed Neuro-Symbolic Generator), a training-free framework that uses untrained neural networks as generative bases to produce high-fidelity, dynamically feasible trajectories. Unlike purely learning-based approaches that rely on soft constraint approximations, PING embeds explicit symbolic verification directly into generation. The network produces diverse function-space candidates that inherently satisfy boundary conditions, avoiding discretization artifacts and local minima common in sampling and gradient-based optimization. PING employs a dual-mode strategy: a rapid single-pass generate-and-verify phase, followed by iterative refinement that warm-starts from feasible candidates. By decoupling candidate generation from sequential numerical optimization and enabling parallel verification, PING reduces practical sensitivity to the number of constraints.

Problem Definition

We begin by formally defining the continuous planning problem. Our problem formulation adapts the definition of a deterministic metric hybrid planning problem proposed by (Say and Sanner 2019). While their work focuses on hybrid discrete-continuous systems, we extend this formulation specifically for continuous function-valued action spaces to address the novelty of our approach.

A continuous planning problem is defined as a tuple $\Pi = \langle S, A, F, C, I, G, M \rangle$, where:

- S is the vector of continuous state variables.
- A is the vector of continuous function-valued action variables.
- F denotes the transition dynamics governed by Ordinary Differential Equations (ODEs).
- C represents the hard constraints on state and action variables.
- I and G denote initial and goal state constraints, respectively.
- M is the cost function to be optimized.

State and Action Spaces The system state is defined by a vector of continuously differentiable state variables $S(t) = (s_1(t), \ldots, s_n(t))^\top$ defined over $t \in [0, T_f]$. The actions are functions $A(t) = (a_1(t), \ldots, a_m(t))^\top$ that directly influence the derivatives of the state variables.

Transition Dynamics The system evolution follows a set of ODEs acting as the continuous analogue to discrete transitions. We formulate the transition dynamics such that for each state variable $s_i(t) \in S$, the dynamics of its highest-order derivative $p_i \in P$ are determined by:

$$\frac{d^{p_i} s_i(t)}{dt^{p_i}} = f_i\big(S(t), D^P S(t), A(t)\big)\,, \tag{1}$$

where $D^P S(t)$ collects all lower-order derivatives. This specific formulation is chosen to capture complex physical dependencies inherent in continuous domains, such as thermodynamic couplings, which simpler dynamical models often overlook.

Hard Constraints and Goals Valid plans must satisfy the following conditions:

- **Initial State (I):** Specifies starting values for state variables: $s_i(0) = V_i^I$.
- **State and Action Constraints (C):** Hard algebraic or logical restrictions on the state and action variables, $C(S(t), A(t)) \leq 0$, which must be satisfied $\forall t \in [0, T_f]$.
- **Goal State (G):** Defines goal constraints $s_i(T_f) = V_i^G$.

Objective and Solution A solution to Π is a tuple $\langle \bar{A}(t), \bar{T}_f \rangle$ representing a feasible state evolution from the initial state I to the goal state G, updated by the transition dynamics F that satisfy constraints C over time $t \in [0, \bar{T}_f]$. The optimal solution is a solution that minimizes the cost function:

$$M(\bar{A}(t), \bar{T}_f) = \int_0^{\bar{T}_f} R(\bar{S}(t), \bar{A}(t))\, dt + \lambda \bar{T}_f \tag{2}$$

where R is a state and action cost function over time and $\lambda \bar{T}_f$ penalizes the makespan for some weight λ.

Methodology

We now introduce *Physics-Informed Neuro-Symbolic Generator* (PING) in order to solve Π effectively. Unlike previous paradigms that rely on solving the sequential optimization problem, we reformulate planning as a *parallel inference task* over a solution space. Our approach operates as a high-throughput pipeline consisting of three stages:

1. **Batched Generative Synthesis**. A batch of untrained neural networks generates candidate solutions fixed to initial I and goal G states;
2. **Parallel Verification**. A dual-stage filter to eliminate candidate solutions violating hard constraints C;
3. **Physics-Informed Refinement**. A fallback mechanism that fine-tunes solutions when the initial generate-and-verify pass is insufficient.

This architecture decouples the sampling, the verification and the refinement processes, enabling massive parallelism while maintaining strict adherence to modelled constraints. An overview is provided in Figure 1 (i.e., at the top of Figure 1).

Batched Synthesis with Analytical Pinning

The aim of the first stage is to effectively sample candidate solutions in parallel that satisfy initial and goal states. We employ a batch of untrained neural networks, parameterized by θ, which maps a time coordinate to a batch $b \in B$ of candidate state trajectories such that:

$$S_\theta^{(b)} : [0, T_f] \to \mathbb{R}^n. \tag{3}$$

Rather than hoping the network learns the initial and goal state constraints via a loss function, we enforce them via a deterministic *Analytical Pinning Layer*. This layer acts as a differentiable bridge, transforming the raw network output into a valid trajectory. We define a linear bridge $LB(t)$ that interpolates from the initial state I to the goal state G, and

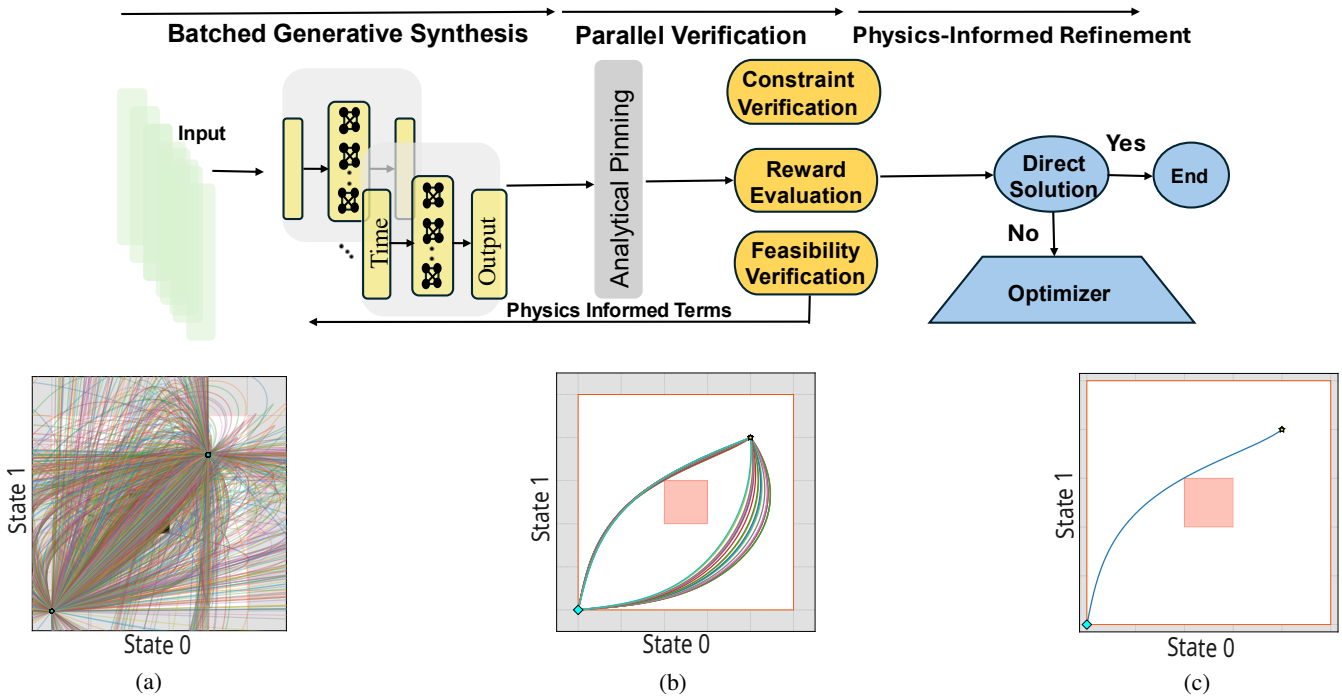

Figure 1: (Top) Our full end-to-end PING pipeline, where a latent batch is processed through the generator. (Bottom) Visualization of the three stages of PING after the analytical pinning layer for the Navigation 2D domain: (a) Output of the Batched Generative Synthesis stage, producing a diverse set of candidate solutions; (b) Results after Parallel Verification stage; (c) Final solution after Physics-Informed Refinement stage.

a gating function $H(t)$ that vanishes at the boundaries of I and G:

$$LB(t) = V^I \cdot (T_f - t) + V^G \cdot t \tag{4}$$

$$H(t) = t \cdot (T_f - t) \tag{5}$$

The final trajectory $\hat{S}^{(b)}(t)$ is synthesized by modulating the raw output with the gate and superimposing it onto the bridge:

$$\hat{S}^{(b)}(t) = LB(t) + H(t) \odot S_\theta^{(b)}(t) \tag{6}$$

where $\odot$ denotes element-wise multiplication. By construction, $H(t)$ forces the raw signal to satisfy the initial and the goal state constraints at $t = 0$ and $t = T_f$, regardless of the network weights θ. In our training-free setting, this provides a simple, deterministic way to encode I and G into the representation without introducing additional optimization terms. If one were to train the network in future extensions, boundary feasibility would not need to be enforced through a loss; instead, optimization could focus on the unconstrained topology of the state trajectory between the initial and goal states.

Parallel Symbolic and Physical Verification

Once a batch of candidate trajectories $\hat{\mathbf{S}}(t)$ is synthesized, we assess their validity. That is, we first apply the cheap symbolic logic to filter violations of C, and then apply the expensive differential operations to check feasibility of F.

Symbolic Constraint Verification We verify constraints C (e.g., obstacle avoidance, safe zones) by applying structured Boolean logic over the batch $\hat{\mathbf{S}}(t)$. Rather than relying on a coarse predefined time grid, constraints are implemented as differentiable predicates applied to the continuous state trajectory representation and evaluated via dense numerical quadrature. For a candidate state trajectory $S_\theta^{(b)}(t) \in \hat{\mathbf{S}}(t)$ to be retained, we require that all constraints hold for all evaluation points along its execution over the interval $[0, T_f^*]$. If any point along the state trajectory violates constraint C beyond a small numerical tolerance ϵ, the entire state trajectory is discarded. This deterministic filtering (with respect to the chosen quadrature scheme and tolerance) yields a subset of candidates $\hat{\mathbf{S}}_C(t) \subseteq \hat{\mathbf{S}}(t)$.

Trajectories in $\hat{\mathbf{S}}_C(t)$ therefore satisfy the constraints C on the state variables within the resolution of our continuous time representation, and subsequent cost evaluation M is restricted to this subset. Following this, we compute the reward for each candidate in $\hat{\mathbf{S}}_C(t)$ by integrating $R(S(t), A(t))$, ensuring that cost evaluation occurs exclusively over candidates that pass the constraint violation check.

Continuous Feasibility Verification via Automatic Differentiation Fundamentally, the neural network operates as a continuous function approximator, mapping the time interval $t \in [0, T_f]$ directly to the state space $S(t)$. Unlike clas-

sical state representations that suffer from discretization errors, the neural network employs smooth, differentiable activation functions (e.g., $\tanh$, σ). This guarantees that the generated state trajectories $\hat{\mathbf{S}}_C(t)$ are continuous, ensuring that the time derivatives $\dot{\hat{\mathbf{S}}}_C(t)$ exist analytically and are well-defined at every point in the continuous time domain. Consequently, we do not need to solve the Initial Value Problem via numerical integration to generate a state trajectory. Instead, we invert the problem, that is, we treat the generated state trajectory $\hat{\mathbf{S}}_C(t)$ as the ground truth and measure its local deviation from the ODE. We compile $\hat{\mathbf{S}}_C(t)$ as a fully differentiable computation graph as visualized in Figure 1, and calculate the exact time derivative $\frac{d\hat{\mathbf{S}}_C(t)}{dt}$ via automatic differentiation with respect to the input time. We then verify adherence to the system dynamics F by comparing this analytical derivative against each f_i. We quantify the infeasibility of each state trajectory in the batch via the ODE residuals as F_{err}

$$F_{\text{err}}\big(\hat{\mathbf{S}}_C(t), A(t)\big) = \frac{1}{T_f}\int_0^{T_f} \Big\| \frac{d\hat{\mathbf{S}}_C(t)}{dt} - F\big(\hat{\mathbf{S}}_C(t), D^P\hat{\mathbf{S}}_C(t), A(t)\big)\Big\|_2^2 dt. \tag{7}$$

which measures the magnitude of the violation between the network's output slope and the ODE-based slope. Here, $\frac{d\hat{\mathbf{S}}_C(t)}{dt}$ is computed via automatic differentiation with respect to time, and $\hat{\mathbf{A}}_{C,F}(t)$ is obtained, when required, by solving an inverse optimization problem. For a batch of state variables, we aggregate the state variable wise residuals into a batched quantity. Only trajectories exhibiting a vanishing residual (up to a small numerical tolerance) are retained for the final selection stage.

Solution Selection and Physics-Informed Refinement

Our architecture executes the synthesis and verification steps in a single forward pass. The final decision logic operates on the resulting set of verified candidates.

Direct Solution Selection We first identify the set of feasible state trajectories $\hat{\mathbf{S}}_{C,F}(t) \subseteq \hat{\mathbf{S}}_C(t)$, and the respective feasible actions $\hat{\mathbf{A}}_{C,F}(t)$ where the ODE residuals on F fall below a numerical tolerance ϵ_1. If this set is non-empty, the optimization problem is effectively solved in a single generate-and-verify pass. We simply select the best trajectory χ^* that minimizes the objective function M:

$$\chi^* = \underset{\hat{T}_f^{(b)}\in\hat{\mathbf{T}}_f, \hat{A}_{C,F}^{(b)}(t)\in\hat{\mathbf{A}}_{C,F}(t)}{\arg\min} M(\hat{A}_{C,F}^{(b)}(t), \hat{T}_f^{(b)}) \tag{8}$$

This mechanism allows the solver to bypass iterative optimization entirely when the generative model successfully samples a valid solution.

Physics-Informed Guided Refinement In complex scenarios where the random seed does not yield a feasible state trajectory, we initiate the refinement stage. Rather than optimizing from scratch, we construct a composite loss function that uses the candidate solutions $\hat{\mathbf{S}}_C(t)$ as warm starts. We formulate a guided optimization objective that jointly minimizes the ODE residual on F and the cost M, and regularize it by $\hat{\mathbf{S}}_C(t)$ such that:

$$\min F_{\text{err}}(\hat{\mathbf{S}}_C(t), A(t)) + \alpha M(A(t), T_f) + \beta \left\|\hat{\mathbf{S}}_C(t) - S^*(t)\right\|_2^2 \tag{9}$$

where the output of generation phase serves as a fixed reference, representing the initial candidate trajectory with the lowest F_{err}. During optimization, the first term actively minimizes the F_{err} of the updated trajectory $\hat{\mathbf{S}}_C(t)$, while the term $||\hat{\mathbf{S}}_C(t) - S^*(t)||_2^2$ acts as a regularizer to ensure the refined state trajectories do not deviate from the almost feasible initial guess $S^*(t)$. Note that α and β are set to 1.0 across all experiments without any domain-specific hyperparameter tuning.

Experimental Results

We now present the results of our computational experiments, evaluating the effectiveness of PING in approximately solving Π.

Evaluation Domains

To assess the capabilities of our planner, we selected three continuous control domains: *Navigation*, *Reservoir Control*, and *HVAC* (Wu, Say, and Sanner 2017). These domains span a diverse spectrum of complexities, from high-dimensional geometric non-convexity to safety-critical state invariance. Table 1 provides the formal specification for each domain below.

Domain	Brief Description
Navigation 2D	Continuous control of an agent in a 2D space with obstacles. Movement is based on: $\frac{ds_i^2(t)}{dt} = a_i(t)$.
Navigation 3D	Similar to Navigation 2D in 3D space.
Reservoir Control	Continuous control of n connected reservoirs preventing overflow. Water levels satisfy: $\frac{ds_i(t)}{dt} = \sum_{j\in J(i)} a_j(t) - a_i(t)$, where $J(i)$ are reservoirs flowing into i. Levels maintained within safety limits.
HVAC	Continuous HVAC management of a building with n rooms. Room temperatures satisfy: $\frac{ds_i(t)}{dt} = c_1 \sum_{j\in J(i)}(s_j(t) - s_i(t))^y + c_2 a_i(t)$, where $J(i)$ are adjacent rooms and c_i are constants, and $y \in \{1, 3\}$. Temperatures maintained within desired limits.

Table 1: Summary of the domain descriptions for Navigation, Reservoir, and HVAC. n is the number of state variables.

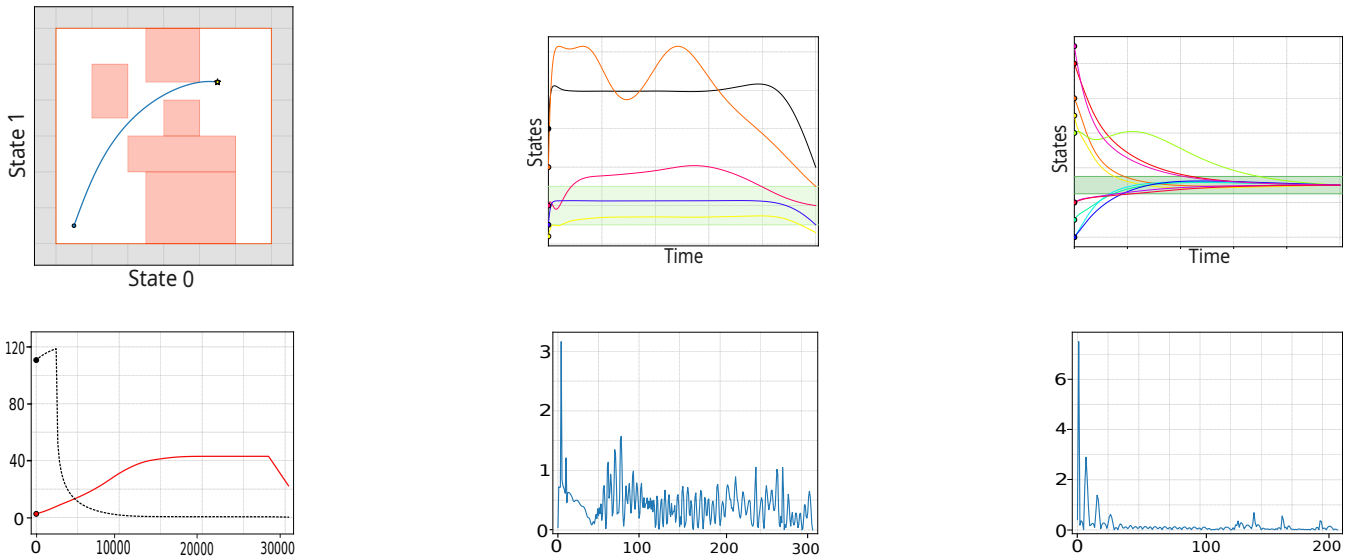

Figure 2: Results for Navigation 2D (left), Reservoir (center), and HVAC (right). Top: system behaviour, including the Navigation trajectory from initial point to goal and state evolution in Reservoir and HVAC. Bottom: training dynamics, showing duration loss (solid) and ODE loss (dashed) for Navigation, and ODE loss for Reservoir and HVAC.

Single-Pass Generate-and-Verify Performance Across Domains

A defining characteristic of the PING architecture is its ability to synthesize promising state trajectories without iterative optimization or domain-specific training. As previously discussed, this is achieved through the *Analytical Pinning Layer*, which analytically enforces initial and goal state constraints (I, G) for any network initialization under the assumed model. By shifting the burden of feasibility on I and G from learnable weights to a deterministic bridge function, PING transforms the planning problem into a high-throughput parallel search over a function space.

Table 2 quantifies this single-pass generate-and-verify capability across three domains over 45 instances in Navigation 2D, Navigation 3D, Reservoir Control and HVAC. The *Coverage* metric highlights the robustness of the generative mechanism: PING successfully identifies at least one feasible solution in 90% of Navigation 2D and HVAC instances, and 100% of Navigation 3D instances. The *Reservoir* domain proved most challenging (70% coverage), reflecting the difficulty of randomly sampling trajectories that satisfy the coupled differential constraints of the hydrological network. The computational efficiency of the *Parallel Verification* engine is evidenced by the discrepancy between total synthesis time (t_{total}) and per-plan generation time ($t_{\text{per-plan}}$). For Navigation 2D, the system generates valid plans at an amortized cost of 8 milliseconds. Even in the computationally heavier Navigation 3D domain, where the search space expands significantly, the system maintains a practical execution window (21.47s total). Crucially, the *Number of plans* column reveals the density of the feasible manifold. In Navigation 2D, the generator is capable of frequently sampling from rich feasible space of state trajectories, returning up to 4000 feasible state trajectories in a single pass.

Domain	Coverage	Plans	$t_{\text{per-plan}}$ (s)	t_{total} (s)
Nav 2D	9/10	(4000, 0)	0.008 ± 0.015	3.39 ± 1.80
Nav 3D	10/10	(2085, 1)	10.40 ± 10.17	21.47 ± 3.03
Reservoir	7/10	(48, 0)	1.18 ± 1.46	4.54 ± 1.77
HVAC	9/10	(54, 0)	1.07 ± 1.36	5.09 ± 2.02

Table 2: Coverage counts the number of instances with at least one feasible trajectory. "Plans" reports (maximum, minimum) feasible set size per batch. $t_{\text{per-plan}}$ and t_{total} are mean $\pm$ standard deviation over instances. Reservoir and HVAC feasibility additionally require mean squared ODE residual $< 10^{-3}$.

In contrast, the Reservoir and HVAC domains, which impose strict ODE residual thresholds ($< 10^{-3}$), yield significantly fewer survivors (max. 48 and 54, respectively). This disparity aligns with the methodological constraint that feasible state trajectories must satisfy both symbolic constraint verification and continuous feasibility verification; the latter is a much stricter filter for random function approximators. While single-pass generate-and-verify coverage varies by domain, the final physics-informed refinement recovers feasibility across all instances, achieving 10/10 coverage for all domains.

Comparative Analysis: PING vs. SCIPPlan

To benchmark against traditional optimization, we compare PING (4-layer, 128-neuron MLP with tanh activations) with SCIPPlan, a sequential convex programming planner (Say and Sanner 2018, 2019; Say 2023; Hojny et al. 2025). Table 3 presents the results for Navigation 2D/3D. The transition from Navigation 2D to Navigation 3D reveals a clear scalability gap. SCIPPlan achieves full coverage in

	Navigation 2D		Navigation 3D	
Metric	SCIPPLAN	PING	SCIPPLAN	PING
Coverage	10/10	10/10	5/10	**10/10**
Total runtime	30.35	**16.87**	903.24	**21.47**
	±53.00	**±14.41**	±945.20	**±3.02**
Cumul. Cost	**8.96**	24.78	**8.46**	46.41
	±3.80	±7.77	**±2.73**	±15.29

Table 3: Summary of coverage, total runtime (mean, std), and cumulative cost (mean, std) for Navigation 2D/3D. Note that for Navigation 3D, reward metrics are reported only for the five instances where both PING and SCIPPlan generated feasible solutions.

2D (10/10 coverage) but degrades substantially in 3D, solving only 50% of instances with a sharp runtime increase (903.24s). This slowdown is characteristic of optimization-based planners that struggle with the non-convexity introduced by higher-dimensional obstacles. Conversely, PING demonstrates superior scalability. In Navigation 3D, PING achieves perfect coverage (10/10) with a mean runtime of only 21.47s—orders of magnitude faster than the baseline. This performance resilience is a direct consequence of the *Batched Synthesis* strategy described in the methodology. Because PING samples functions rather than optimizing waypoints sequentially, the added dimensionality does not exponentially increase the complexity of the filtering step; it merely requires verifying an additional state coordinate in the parallel tensor operations. The *Reward* metric highlights the trade-off. SCIPPlan consistently achieves lower costs due to iterative convergence toward local optima, whereas PING selects the best candidate from a sampled distribution. PING prioritizes rapid feasible solution discovery over single-pass cost optimality. For fair comparison, physics-informed refinement is applied to the best candidate in each batch across all instances in Table 3. This improves feasibility: in Navigation 2D, coverage increases from 9/10 (single-pass) to 10/10 (post-refinement). In Navigation 3D, reward is reported only for the five instances where both methods produced feasible solutions, ensuring comparable coverage.

Scalability and Hardware Constraints

All experiments were run on an NVIDIA RTX 1000 Ada (6 GB VRAM) Generation laptop GPU. The methodology employs domain-adaptive sampling. Namely, for Navigation where checking C is the primary bottleneck, volume is prioritized (10,000 candidate trajectories) via batched inference. For Reservoir and HVAC, where F_{err} constrains the solution space, the search is restricted to a single batch of 1,000 models. The results indicate that PING's primary bottleneck is GPU memory (VRAM) rather than computational complexity. As the batch size increases, the parallel verification of symbolic constraints scales linearly in memory usage. However, unlike SCIPPlan, where runtime scales superlinearly with the number of constraints $|C|$, PING verifies all constraints simultaneously across the batch tensor. This architectural advantage suggests that PING is particularly well-suited for highly constrained environments where traditional sequential solvers face combinatorial explosion.

Refinement Design Trade-offs

The cumulative cost gap to SCIPPlan reflects two design choices in PING's refinement formulation:

1. **Local Optimization with Conservative Geometric Regularization:** The term $|\hat{\mathbf{S}}_C(t) - S^*(t)|_2^2$ constrains refinement to remain within the initially feasible manifold. For the navigation domains, this ensures collision avoidance is preserved but prevents discovery of substantially different geometric configurations that might achieve better time horizons. It should also be noted that in some instances, the non-convex loss landscape prevents gradient descent from finding better solutions, unlike SCIPPlan's global optimal control solver.
2. **Fixed Multi-Objective Loss Weighting:** Coefficients α and β are hyperparameters set globally across all instances, balancing ODE residual minimization, reward optimization, and geometric feasibility. This uniform weighting prioritizes constraint satisfaction over cost minimization, which is appropriate for ensuring constraint satisfaction but sacrifices optimality on the objective.

These limitations are architectural constraints of the neural network-based refinement approach, not fundamental to parallel trajectory generation. Future work could address this through learned parametrizations of time horizon or using other solvers.

Scalability in High-Dimensional HVAC Systems

To evaluate scalability, we test PING on six additional HVAC instances ranging from 10 to 60 rooms. These instances are more challenging than the previous ones, as the initial single-pass generate-and-verify stage fails when the problem size exceeds the available sampling budget. The difficulty is fundamentally combinatorial: a reward is obtained only if all rooms simultaneously remain within their comfort temperature ranges. Although the number of constraints grows linearly with the number of rooms, jointly satisfying all of them becomes increasingly difficult as the system scales. Despite the significantly harder task, PING preserves robust performance, with normalized rewards of 0.090 at 10 rooms and 0.076 at 60 rooms, corresponding to just a 15.6% reduction across a 6 times increase in dimensionality. This stability results from two mechanisms: first, the generative model produces $\hat{\mathbf{A}}_{C,F}(t)$, the set of feasible state trajectories, providing a warm start that already satisfies structural constraints; second, the refinement process leverages direct derivative computation to optimize state trajectories in the continuous function space, enabling efficient local improvement without resampling or sequential exploration. Lastly, runtime scales approximately linearly with the problem size in PING, directly resulting in speed-ups. This linear scaling contrasts sharply with exponential growth expected from sequential constraint satisfaction methods. The

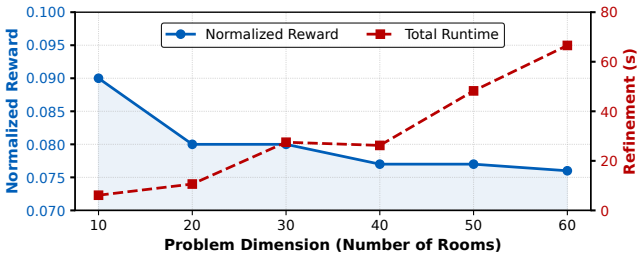

Figure 3: The scalability of PING is demonstrated in the HVAC Domain; as dimension scales from 10 to 60 rooms, normalized reward (blue) remains stable while runtime (red) scales linearly.

linear behavior reflects PING's architectural design: batched neural network generation synthesizes promising state trajectories across high-dimensional action spaces in parallel, while GPU-accelerated verification checks all constraints simultaneously. This parallelism avoids the sequential bottlenecks that plague traditional optimal control solvers when facing high-dimensional multi-objective optimization problems.

Related Work and Discussion

Continuous planning has traditionally relied on two complementary approaches: classical optimization-based methods and, more recently, learning-based neural approaches. PING bridges these paradigms by reformulating planning from sequential constraint satisfaction to parallel manifold inference over continuous function spaces, enabling orders-of-magnitude speedups while empirically maintaining strict constraint satisfaction up to numerical tolerances.

Classical Optimization and Control-Theoretic Methods

Control theory methods such as Model Predictive Control (MPC) and Control Barrier Functions (CBFs) are highly effective for online, closed-loop control. MPC with Lyapunov constraints unifies planning and tracking by iteratively solving optimal control problems during execution (Csomay-Shanklin et al. 2022), while CBFs combined with Control Lyapunov Functions (CLFs) reactively enforce safety and stability constraints at each timestep (Reis and Aguiar 2024; Ames et al. 2016). In contrast to these online control methods that evaluate actions iteratively in the loop, our work focuses specifically on continuos planning, generating complete, dynamically feasible trajectories proactively without real-world interaction. Sampling-based planners such as RRT and PRM achieve probabilistic completeness but exhibit poor scaling due to extensive node-by-node exploration, introducing discretization artifacts that prevent full exploitation of differential constraints (Kleinbort et al. 2018; Luo et al. 2023).

Unlike these sequential methods, PING eliminates iterative optimization through parallel generation: rather than solving a single Boundary Value Problem per problem instance, it generates thousands of candidate trajectories simultaneously, filtering them via symbolic constraint verification and continuous feasibility verification. This amortization enables planning in seconds rather than minutes, while maintaining the same hard constraints as the underlying continuous model, up to numerical tolerances.

Discretization Strategies in Hybrid Planning

Recent works on PDDL-based planners have explored multiple discretization strategies to bridge continuous and discrete planning. Scala et al. (Scala and Vallati 2021) use lifted reasoning to avoid combinatorial explosion in grounded models, while Percassi et al. (Percassi, Scala, and Vallati 2023b) discretize the continuous-time planning problem into the numeric planning problem, sacrificing fine-grained temporal semantics for access to mature planning engines. Extensions such as multiple discretizations support variable time steps across agents (Cardellini et al. 2024), and allow for plan repair methods when repairs become necessary at discrete intervention points (Percassi, Scala, and Vallati 2023a). These approaches accept discretization error as a trade-off for computational tractability. PING avoids explicit time discretization: all trajectory generation and verification processes operate on continuous-time differentiable function representations. The analytical pinning layer and automatic differentiation-based verification compute ODE residuals over the continuous representation without introducing additional time-step quantization error, greatly reducing accumulated discretization artifacts inherent to PDDL-based methods.

Physics-Informed and Learning-Based Planning

Physics-informed neural networks (PINNs) embed differential equations and boundary conditions as soft loss constraints to accelerate trajectory computation (Krishnapriyan et al. 2021). However, minimizing loss functions does not guarantee hard constraint satisfaction across all problem instances. Recent deterministic variants (hPINN, KKT-hPINN) address this through strengthened constraint encoding (Xu and Pan 2024; Cheng and Na 2024; Zhou et al. 2024; Kokolakis et al. 2025), demonstrating that separating constraint satisfaction from optimization can enable more reliable solutions. PING extends this insight by making boundary-condition satisfaction structural and handling other constraints through explicit verification. The analytical pinning layer deterministically enforces boundary conditions without penalty losses—not through learning but through analytical design. This is fundamentally different from hPINN-style learned hard constraints, as PING's boundary-satisfaction property holds for any random network weights θ. Gradient-based continuous planning methods such as TensorFlow Planning (Wu, Say, and Sanner 2017) and related deep model-based planners optimize open-loop action sequences (or policies) via automatic differentiation in high-dimensional action spaces. These approaches demonstrate impressive scalability (hundreds of thousands of continuous action parameters) but treat dynamics and constraints through differentiable objectives, without providing general hard constraint guarantees across all states and times. Deep Reactive Policies (Bueno et al. 2019)

similarly parametrize policies with deep networks and train them by gradient-based policy search in stochastic nonlinear domains, leveraging differentiable models and reparameterization. DRP-style methods can enforce simple action-space bounds (e.g., via squashing or clipping), but they do not provide formal satisfaction of richer state or trajectory constraints; constraint handling remains embedded in the loss and training process rather than being structurally guaranteed. PING achieves comparable scalability through batched parallel generation while requiring zero training: constraint satisfaction is enforced analytically at the representation level and verified symbolically, rather than being optimized statistically over data.

Generative and Diffusion-Based Approaches

Diffusion models leverage parallel sampling to generate diverse trajectory candidates, avoiding sequential optimization bottlenecks. Methods such as Motion Planning Diffusion (Carvalho et al. 2023) and PRESTO (Seo et al. 2025) can generate thousands of candidates in parallel. However, generated trajectories do not inherently satisfy constraints; subsequent verification or refinement steps often reveal violations. These recent works reintroduce optimization into the denoising process to enforce constraints, sacrificing the parallel efficiency advantage. PING preserves parallelism while ensuring feasibility: the analytical pinning layer enforces boundary satisfaction by construction before verification, and symbolic constraint checking operates over the continuous domain without coarse discretization. Unlike diffusion methods that generate trajectories post-hoc and then refine, PING embeds constraints directly into the generation architecture, reducing a posteriori violations from the outset.

Temporal Logic and Dense-Time Planning

Theoretical work on dense-time semantics (Gigante et al. 2022) establishes that continuous-time planning over $\mathbb{R}$ ranges from PSPACE-complete (with temporal separation constraints) to undecidable (with zero-duration actions). Interval temporal logic approaches (Bozzelli, Montanari, and Peron 2019) avoid discretization by explicitly modeling concurrency relations, but scalability remains challenging. PING sidesteps complexity-theoretic barriers by reformulating the problem: instead of searching a dense-time space, it generates feasible manifold regions analytically and verifies adherence symbolically. This transforms the problem from one of symbolic search over continuous time to one of gradient-based constraint satisfaction in function space.

Stochastic and Risk-Aware Planning

Risk-bounded motion planning approaches employ probabilistic safety guarantees (Huang et al. 2019), chance-constrained MPC (Schwarm and Nikolaou 1999), and scenario-based robustness (Comes et al. 2010). These methods sacrifice deterministic guarantees for handling stochasticity. Prioritized multi-agent planning (Kasaura, Nishimura, and Yonetani 2022) extends continuous-time semantics to dozens or hundreds of agents using collision-free primitives. PING is deterministic: it enforces constraint satisfaction at the representation and verification stages for nominal dynamics without probabilistic approximations. Future extensions could integrate stochastic verification through risk propagation within the verification stage, leveraging the modular architecture.

Positioning PING: Core Contribution

1. **Deterministic Structural Constraints:** Unlike PINNs and gradient-based planning or policy search methods that rely on loss-driven learning (e.g., TensorFlow Planning (Wu, Say, and Sanner 2017) and Deep Reactive Policies (Bueno et al. 2019)), PING's analytical pinning layer mathematically enforces boundary satisfaction regardless of network initialization, encoding boundary conditions structurally rather than as an additional optimization objective.
2. **Continuous Verification Without Coarse Discretization:** Unlike PDDL and sampling-based methods, PING leverages automatic differentiation to compute ODE residuals over continuous spatiotemporal domains. Verification time does not grow significantly with time-step resolution, and residuals are computed analytically over the continuous representation rather than being tied to a fixed grid.
3. **Parallel Generation and Verification:** Unlike sequential methods (MPC, CBF/CLF) and even recent diffusion planners that require post-hoc refinement, PING generates and verifies thousands of candidates in a single forward pass. This amortization enables planning latencies of seconds on GPU, with feasible set sizes enabling downstream multi-objective selection without resolving.

Conclusion

PING is a generative neuro-symbolic continuous-time planner that reformulates planning from sequential constraint satisfaction to parallel manifold inference over continuous function spaces. It decouples candidate generation from numerical optimization through three components: batched generative synthesis with analytical boundary pinning, parallel symbolic and physical verification, and physics-informed refinement. The framework achieves rapid trajectory synthesis in seconds without domain-specific training or offline data collection. Empirical evaluation across Navigation, Reservoir control and HVAC domains demonstrates robust feasibility across continuous state and action spaces. PING shows superior scalability compared to optimization-based baselines, with perfect coverage in the challenging Navigation 3D domain and only linear runtime growth in HVAC with up to 60 rooms. Limitation of PING include potential suboptimality and dependence on warm-starting for refinement trade-offs between rapid solution discovery and global optimality. Nevertheless, our experiments show that decoupling trajectory generation from verification enables a training-free planner with favourable scaling behaviour as constraint dimensionality increases, providing a practical alternative to conventional optimization and soft-constraint learning approaches.

References

Ames, A. D.; Coogan, S.; Egerstedt, M.; Notomista, G.; Sreenath, K.; and Tabuada, P. 2019. Control barrier functions: Theory and applications. In *2019 18th European Control Conference (ECC)*, 3420–3431. IEEE.

Ames, A. D.; Xu, X.; Grizzle, J. W.; and Tabuada, P. 2016. Control barrier function based quadratic programs for safety critical systems. *IEEE Transactions on Automatic Control*, 62(8): 3861–3876.

Bozzelli, L.; Montanari, A.; and Peron, A. 2019. Taming the complexity of timeline-based planning over dense temporal domains. In *39th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2019)*, 34–1. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

Bueno, T. P.; de Barros, L. N.; Mauá, D. D.; and Sanner, S. 2019. Deep reactive policies for planning in stochastic nonlinear domains. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 7530–7537.

Cardellini, M.; Maratea, M.; Percassi, F.; Scala, E.; and Vallati, M. 2024. Taming discretised PDDL+ through multiple discretisations. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 34, 59–67.

Carvalho, J.; Le, A. T.; Baierl, M.; Koert, D.; and Peters, J. 2023. Motion planning diffusion: Learning and planning of robot motions with diffusion models. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1916–1923. IEEE.

Chen, Q.; and Wang, M. 2022. An improved RRT* algorithm for mobile robots path planning. In *2022 China Automation Congress (CAC)*, 4334–4339.

Cheng, X.; and Na, S. 2024. Physics-informed neural networks with trust-region sequential quadratic programming. *arXiv preprint arXiv:2409.10777*.

Cimini, G.; Bernardini, D.; Levijoki, S.; and Bemporad, A. 2021. Embedded Model Predictive Control With Certified Real-Time Optimization for Synchronous Motors. *IEEE Transactions on Control Systems Technology*, 29(2): 893–900.

Comes, T.; Hiete, M.; Wijngaards, N.; and Schultmann, F. 2010. Enhancing robustness in multi-criteria decision-making: A scenario-based approach. In *2010 International Conference on Intelligent Networking and Collaborative Systems*, 484–489. IEEE.

Csomay-Shanklin, N.; Taylor, A. J.; Rosolia, U.; and Ames, A. D. 2022. Multi-rate planning and control of uncertain nonlinear systems: Model predictive control and control lyapunov functions. In *2022 IEEE 61st Conference on Decision and Control (CDC)*, 3732–3739. IEEE.

Elango, P.; Luo, D.; Uzun, S.; Kim, T.; and Açikmese, B. 2024. Successive Convexification for Trajectory Optimization with Continuous-Time Constraint Satisfaction.

Gigante, N.; Micheli, A.; Montanari, A.; and Scala, E. 2022. Decidability and complexity of action-based temporal planning over dense time. *Artificial Intelligence*, 307: 103686.

Hojny, C.; Besançon, M.; Bestuzheva, K.; Borst, S.; Chmiela, A.; Dionísio, J.; Eifler, L.; Ghannam, M.; Gleixner, A.; Göß, A.; Hoen, A.; van der Hulst, R.; Kamp, D.; Koch, T.; Kofler, K.; Lentz, J.; Maher, S. J.; Mexi, G.; Mühmer, E.; Pfetsch, M. E.; Pokutta, S.; Serrano, F.; Shinano, Y.; Turner, M.; Vigerske, S.; Walter, M.; Weninger, D.; and Xu, L. 2025. The SCIP Optimization Suite 10.0. Technical report, Optimization Online.

Huang, X.; Hong, S.; Hofmann, A.; and Williams, B. C. 2019. Online risk-bounded motion planning for autonomous vehicles in dynamic environments. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 29, 214–222.

Jiménez, S.; De La Rosa, T.; Fernández, S.; Fernández, F.; and Borrajo, D. 2012. A review of machine learning for automated planning. *The Knowledge Engineering Review*, 27(4): 433–467.

Kasaura, K.; Nishimura, M.; and Yonetani, R. 2022. Prioritized safe interval path planning for multi-agent pathfinding with continuous time on 2D roadmaps. *IEEE Robotics and Automation Letters*, 7(4): 10494–10501.

Kleinbort, M.; Solovey, K.; Littlefield, Z.; Bekris, K. E.; and Halperin, D. 2018. Probabilistic completeness of RRT for geometric and kinodynamic planning with forward propagation. *IEEE Robotics and Automation Letters*, 4(2): i–vii.

Kokolakis, N.-M. T.; Zhang, Z.; Liu, S.; Vamvoudakis, K. G.; Darbon, J.; and Karniadakis, G. E. 2025. Safe Physics-Informed Machine Learning for Optimal Predefined-Time Stabilization: A Lyapunov-Based Approach. *IEEE Transactions on Neural Networks and Learning Systems*.

Krishnapriyan, A.; Gholami, A.; Zhe, S.; Kirby, R.; and Mahoney, M. W. 2021. Characterizing possible failure modes in physics-informed neural networks. *Advances in neural information processing systems*, 34: 26548–26560.

Lai, J.; Wu, Z.; Ren, Z.; Chen, C.; Tan, Q.; and Xie, S. 2025. A Lyapunov-Based Framework for Trajectory Planning of Wheeled Vehicle Using Imitation Learning. *IEEE Transactions on Automation Science and Engineering*.

Luo, S.; Zhang, M.; Zhuang, Y.; Ma, C.; and Li, Q. 2023. A survey of path planning of industrial robots based on rapidly exploring random trees. *Frontiers in Neurorobotics*, 17: 1268447.

Mittal, M.; Gallieri, M.; Quaglino, A.; Salehian, S. S. M.; and Koutník, J. 2020. Neural lyapunov model predictive control. *arXiv preprint arXiv:2005.12611*.

Mohanty, N. R.; Jugade, C.; Patne, V.; Ingole, D.; and Sonawane, D. 2022. FPGA Implementation of Low Complexity Nonlinear Model Predictive Control Using Deep Learning Approach. In *2022 Eighth Indian Control Conference (ICC)*, 325–330.

Mori, Y.; Mori, T.; and Kuroe, Y. 1997. A solution to the common Lyapunov function problem for continuous-time systems. In *Proceedings of the 36th IEEE Conference on Decision and Control*, volume 4, 3530–3531. IEEE.

Nasir, J.; Islam, F.; Malik, U.; Ayaz, Y.; Hasan, O.; Khan, M.; and Muhammad, M. S. 2013. RRT*-SMART: A Rapid

Convergence Implementation of RRT*. *International Journal of Advanced Robotic Systems*, 10(7): 299.

Percassi, F.; Scala, E.; and Vallati, M. 2023a. On the Notion of Fixability of PDDL+ Plans. In *Proceedings of the International Symposium on Combinatorial Search*, volume 16, 177–178.

Percassi, F.; Scala, E.; and Vallati, M. 2023b. A practical approach to discretised PDDL+ problems by translation to numeric planning. *Journal of Artificial Intelligence Research*, 76: 115–162.

Reis, M. F.; and Aguiar, A. P. 2024. On the Stability of Undesirable Equilibria in the Quadratic Program Framework for Safety-Critical Control. *arXiv preprint arXiv:2402.08027*.

Say, B. 2023. Robust metric hybrid planning in stochastic nonlinear domains using mathematical optimization. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 33, 375–383.

Say, B.; and Sanner, S. 2018. Metric nonlinear hybrid planning with constraint generation. *PlanSOpt 2018*, 19–25.

Say, B.; and Sanner, S. 2019. Metric hybrid factored planning in nonlinear domains with constraint generation. In *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, 502–518. Springer.

Scala, E.; and Vallati, M. 2021. Effective grounding for hybrid planning problems represented in PDDL+. *The Knowledge Engineering Review*, 36: e9.

Schwarm, A. T.; and Nikolaou, M. 1999. Chance-constrained model predictive control. *AIChE Journal*, 45(8): 1743–1752.

Seo, M.; Cho, Y.; Sung, Y.; Stone, P.; Zhu, Y.; and Kim, B. 2025. Presto: Fast motion planning using diffusion models based on key-configuration environment representation. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, 10861–10867. IEEE.

Testouri, M.; Elghazaly, G.; and Frank, R. 2023. Towards a Safe Real-Time Motion Planning Framework for Autonomous Driving Systems: A Model Predictive Path Integral Approach. In *2023 3rd International Conference on Robotics, Automation and Artificial Intelligence (RAAI)*, 231–238. IEEE.

Vu, B.; Migimatsu, T.; and Bohg, J. 2024. Coast: Constraints and streams for task and motion planning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 14875–14881. IEEE.

Wu, G.; Say, B.; and Sanner, S. 2017. Scalable planning with tensorflow for hybrid nonlinear domains. *Advances in Neural Information Processing Systems*, 30.

Xu, R.; and Pan, X. 2024. Weakly-Supervised Physics-Informed Neural Networks with Hard Constraints for Sound Propagation Problems in Complex Environments. In *OCEANS 2024 - Singapore*, 1–6.

Zhang, B.; and Li, C. 2024. The Optimization and Application Research of the RRT-APF-Based Path Planning Algorithm. *Electronics*, 13(24): 4963.

Zhou, C.; Li, T.; Lan, C.; Du, R.; Xin, G.; Nan, P.; Yang, H.; Wang, G.; Liu, X.; and Li, W. 2024. Physics-Informed Neural Networks with Complementary Soft and Hard Constraints for Solving Complex Boundary Navier-Stokes Equations. *arXiv preprint arXiv:2411.08122*.