2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Neural Kinodynamic Planning: Learning for KinoDynamic Tree Expansion

Tin Lai[†,§,*], Weiming Zhi[~], Tucker Hermans[‡,£] and Fabio Ramos[†,£]

***Abstract*— We integrate neural networks into kinodynamic motion planning and present the Learning for KinoDynamic Tree Expansion (L4KDE) method. Tree-based planning approaches, such as rapidly exploring random tree (RRT), are the dominant approach to finding globally optimal plans in continuous state-space motion planning. Central to these approaches is tree expansion, the procedure in which new nodes are added to an ever-expanding tree. We study the kinodynamic variants of tree-based planning, where we have known system dynamics and kinematic constraints. In the interest of quickly selecting nodes to connect newly sampled coordinates, existing methods typically cannot optimise the finding of nodes that have a low cost to transition to sampled coordinates. Instead, they use metrics like Euclidean distance between coordinates as a heuristic for selecting candidate nodes to connect to the search tree. We propose L4KDE to address this issue. L4KDE uses a neural network to predict transition costs between queried states, which can be efficiently computed in batch, providing much higher quality estimates of transition cost compared to commonly used heuristics while maintaining almost-surely asymptotic optimality guarantee. We empirically demonstrate the significant performance improvement provided by L4KDE on a variety of challenging system dynamics, with the ability to generalise across different instances of the same model class and in conjunction with a suite of modern tree-based motion planners.**

I. Introduction

This paper seeks to integrate neural network-based learning into kinodynamic motion planning. We focus on asymptotically optimal, kinodynamic motion planning, where one aims to find a globally optimal plan between the start and goal coordinates while remaining both (1) collision-free and (2) compliant with the kinodynamic constraints of the robot or control system. When the search-space is continuous, kinodynamic variants of tree-based motion planning algorithms are often the main workhorse for tackling these problems. Existing tree-based methods often consider both collision and kinodynamic constraints jointly during the motion planning procedure. However, if one disentangles the two constraints, one notices that collision constraints are *environment-specific*. In contrast, kinodynamic constraints are inherited from the robot hardware and do not vary over different planning tasks.

Our motivation stems from the critical need for efficient and globally optimal motion planning in the context of kinodynamic systems. Kinodynamic motion planning is a fundamental problem in robotics. Robots must navigate through complex environments while adhering to collision avoidance and kinematic constraints. Achieving this balance is essential for real-world applications like autonomous vehicles, industrial robots, and humanoid robots. In particular, unlike the assumption of holonomic robots, practical, real-world robots are often subjected to some form of kinodynamic constraints. The choice of nodes directly impacts the efficiency and quality of the resulting motion plan, where existing methods typically rely on simplistic heuristics, like Euclidean distance, to guide node selection. By providing a more accurate estimation of state-transition cost, tree expansion can often benefit from successful and realistic distance estimation during state forward propagation, thereby expanding the capabilities of kinodynamic planners.

With this insight, we integrate neural networks into the planning process and contribute the **L**earning for **K**ino**D**ynamic Tree **E**xpansion (L4KDE) method within tree-based motion planners. L4KDE utilises a neural network to predict the cost of transitioning between states under kinodynamic constraints, allowing for efficient retrieval of the cost when planning online. By leveraging the high representation capacity of neural networks, L4KDE is exceptionally novel in that it can also be directly conditioned on the parameters of the dynamics model (For example, velocity limits). By employing the cost as a node selection heuristic in tree expansion of the underlying planning algorithm, we obtain significant improvements in planning performance over the commonly used Euclidean heuristic. We show that L4KDE consistently improves performance when used within a wide range of tree-based motion planning algorithms across a variety of tasks. Importantly we show that the improvement provided by L4KDE holds for state-of-the-art, tree-based, asymptotically optimal kinodynamic motion planners [1], [2].

II. Related Work

Kinodynamic planning refers to the class of robot systems that must obey kinematic and differential constraints [3]. For example, velocity, acceleration or force bounds in systems such as nonholonomic vehicles [4] and locomotion systems [5]. *Sampling-based motion planners* (SBPs) are a class of algorithms that avoid explicit representation of state space while connecting feasible states to form solution trajectories. Compared to solutions given by trajectory optimisation techniques which are prone to local minima [6], or reactive methods which can be myopic [7], [8], SBPs provides *probabilistic completeness* [9] and *almost-surely asymptotic optimality* guarantee [10].

Tree-based methods such as RRT [11] are the predominant methods in SBPs. RRTs utilise random sampling to create Voronoi bias [12], which helps to explore the space

*Correspondence to tin.lai@sydney.edu.au [†]School of Computer Science, The University of Sydney, Australia. [§]Fait Corporation, Australia. [~]Robotics Institute, Carnegie Mellon University. [‡]School of Computing, University of Utah. [£]NVIDIA, USA.

rapidly [13]. Compared to roadmap-based methods such as PRM [14], tree-based methods do not require a steering function, which requires solving the two-point boundary value problem. Techniques that focus on sampling-based planning can either focus on the algorithmic parts of graph construction [15] or the sampling aspect of the planner [16]. Experience-based approaches can also improve planning efficiency, for example, the Lightning [17] and Thunder [18] framework uses database structure to store past trajectories. However, most of these approaches only focus on kinematic planning and are not directly applicable to kinodynamic planning. Additionally, L4KDE is able to generalise over entire classes of dynamics models, where each may have different constraint limits or parameter values, which are conditioned.

For kinodynamic motion planning, we can form a suitable steering function with LQR [19] in systems dynamics that quadratic functions can linearise. Some approaches use techniques to discover narrow passages [20], [21], uses space projection [22] or heuristic measures of obstacles boundary [23] to improve sampling. However, most approaches do not work with kinodynamic constraints. There have also been numerous learning approaches, such as leveraging experience or environment [24] to utilise learned neural networks for sampling. However, this is often not directly applicable to the general class of kinodynamic problems, when more complex differential constraints are involved.

Learning-based approaches have been one of the main focuses in the sampling procedure of SBPs, for example, by learning sampling distribution [16] from previous experience or utilising obstacle information to form an informed distribution [25]. However, the node selection procedure is understudied, as most existing works use Euclidean distance to guide tree expansion [26]. Moreover, there are a few works dedicated to asymptotically optimal, kinodynamic planning [1], [2], [27], but most continue to use the ill-formed Euclidean distance metric to expand nodes in their search trees. In this work, we attempt to tackle this issue by formalising the learning-based tree expansion procedure under the kinodynamic planning framework.

III. Kinodynamic Tree-based Motion Planning

A. Problem Setup

We shall first outline the kinodynamic motion planning problem. Let the state space be denoted as $\mathcal{X}$, the set of controls within the control constraints as $\mathcal{U}$, the set of states where kinematic constraints are satisfied as $\mathcal{K}$, and the set of collision-free states as $\mathcal{O}$.

Definition 1: A feasible kinodynamic planning problem asks, given an initial state $\mathbf{x}_0$, goal state $\mathbf{x}_{goal}$, system dynamics given by f, to produce a trajectory $\mathbf{x}(t) : [0, t_{max}] \to \mathcal{X}$ and a sequence of controls $\mathbf{u}(t) : [0, t_{max}] \to \mathcal{U}$, such that:

$$\mathbf{x}(0) = \mathbf{x}_0, \quad \text{(initial state)} \quad (1)$$

$$\mathbf{x}(t_{max}) = \mathbf{x}_{goal}, \quad \text{(goal state)} \quad (2)$$

$$\mathbf{x}(t) \in \mathcal{K} \subseteq \mathcal{X}, \forall t \in [0, t_{max}], \quad \text{(kinematics)} \quad (3)$$

$$\mathbf{x}(t) \in \mathcal{O} \subseteq \mathcal{X}, \forall t \in [0, t_{max}], \quad \text{(obstacles)} \quad (4)$$

$$\mathbf{x}'(t) = f(\mathbf{x}(t), \mathbf{u}(t)), \forall t \in [0, t_{max}], \quad \text{(dynamics)} \quad (5)$$

The *optimal kinodynamic planning problem* additionally asks to find a trajectory that minimises an objective cost, subject to the constraints in eqs. (1) to (5). This objective is given by:

$$C(\mathbf{x}_0, \mathbf{x}) = \int_0^{t_{max}} L(\mathbf{x}(t), \mathbf{u}(t))\mathrm{d}t + \Phi(\mathbf{x}(t_{max})), \quad (6)$$

where L is the incremental cost, and Φ is the terminal cost.

B. Tree-Expansion

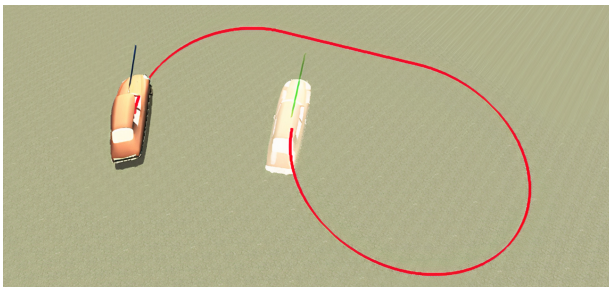
Tree-based motion planners like RRT* [28] iteratively connect new coordinates into a space-filling tree by randomly sampling and checking coordinates before expanding the tree to connect with valid samples. This typically requires a *steering function*, which finds the optimal path between two coordinates under the presence of no obstacle—in the absence of kinodynamic constraints, this is simply finding a straight line. Although a steering function may be available for a restrictive class of kinodynamically constrained systems, we cannot generally find the optimal path without resorting to solving an optimal kinodynamic planning problem between a tree node and the sampled point.

To address this challenge, kinodynamic variants of tree-based methods have been introduced. These methods do not seek to connect sampled coordinates deemed valid but simplify the optimisation and attempt to forward propagate from the nearest existing tree node in a kinodynamically feasible manner to a viable coordinate "closest enough" to the sample, where the viable coordinate to the tree in lieu of the sample.

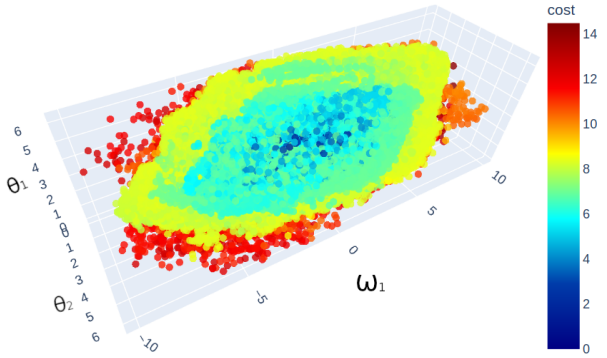
The algorithmic outline of the tree expansion is shown in algorithm 1. In the tree-building process, provided a randomly sampled coordinate, we: (i) find the nearest node, $\mathbf{x}_{active}$, in the existing tree $\mathcal{T}$, via $\texttt{Nearest}(\cdot)$; (ii) propagate trajectories by holding a valid control from $\mathbf{x}_{active}$, for a randomly sampled amount of time, to find a candidate "close" coordinate $\mathbf{x}_{candidate}$. This is typically performed by either randomly selecting a control, or randomly sampling and holding multiple controls and selecting the sampled control where the final state is the closest to $\mathbf{x}_{sample}$; (iii) validate whether $\mathbf{x}_{candidate}$ can be feasibly connected to $\mathbf{x}_{active}$, by checking if the connection satisfies collision constraints and if the cost to reach the new node is below the current best cost c^*; (iv) connect $\mathbf{x}_{candidate}$ as a new node to $\mathcal{T}$. The tree expansion process is run iteratively as we sample more coordinates as $\mathbf{x}_{sample}$.

IV. L4KDE

In this section, we introduce our method of kinodynamic tree expansion, **L**earning for **K**ino**D**ynamic Tree **E**xpansion (L4KDE). In the following sections, we (i) outline the flaws of the existing active node selection heuristic in tree-based



(a) Existing tree-based kinodynamic methods select the active node with Euclidean distance. The two coordinates are close in Euclidean space but moving from the left to the destination on the right, assuming Dublin's car dynamics, may be fairly costly to reach (via red trajectory).



(b) Transition costs between the origin state ($\theta_1 = \pi, \theta_2 = \pi$) and other coordinates in state-space, for the *double pendulum* (with 4D state-space). Axes shows the the two joint angles θ_1, θ_2 and the angular velocity of the first joint ω_1 (ω_2 omitted for visualisation).

Fig. 1: Illustrations of (a) the problem with an ill-informed metric and (b) visualisation of learning the differential system dynamics with the proposed L4KDE. Existing tree-based kinodynamic methods select the active node with Euclidean distance. The two coordinates are close in Euclidean space. However, moving from the current coordinates on the left to the destination on the right, assuming Dublin's car dynamics, may be fairly costly to reach (via red trajectory).

Algorithm 1 Kinodynamic Tree-Expansion

Require: $\mathcal{T}$, $\mathbf{x}_{sample}$, $\texttt{HoldControl}(\cdot,\cdot)$, $\texttt{Nearest}(\cdot)$, $\mathcal{U}$, $C(\cdot,\cdot)$, c^*, $\texttt{CollisionCheck}(\cdot)$, $\texttt{ExpandTree}(\cdot,\cdot,\cdot)$
$\mathbf{x}_{active} \leftarrow \texttt{Nearest}(\mathbf{x}_{sample})$
$\mathbf{x}_{candidate} \leftarrow \texttt{HoldControl}(\mathbf{x}_{active}, \mathbf{u}, t)$
if $(\texttt{CollisionCheck}(\mathbf{x}_{candidate}))$ & $(C(\mathbf{x}_0, \mathbf{x}_{candidate}) < c^*)$ **then**
  $\mathcal{T} \leftarrow \texttt{ConnectNewNode}(\mathcal{T}, \mathbf{x}_{active}, \mathbf{x}_{candidate})$
end if

planning in the kinodynamic setup, where Euclidean distance or absolute angular difference is used as a proxy for transition cost [1]; (ii) describe a learning-based approach to predict transition costs for the selection of active nodes quickly; (iii) elaborate on training a neural network to predict state transition costs.

A. Euclidean-based Heuristics: Fast but Flawed

The heuristic $\texttt{Nearest}(\cdot)$ aims to find the node in the existing tree, which is the nearest to the sampled point under kinematic constraints. From algorithm 1, we can see that this is decoupled from the collision-checking. When calling $\texttt{Nearest}(\cdot)$ during tree-expansion, existing kinodynamic tree-based planning methods select the closest node coordinates from the tree, $\mathcal{T}$, using Euclidean distance,

$$\mathbf{x}_{active} = \texttt{Nearest}(\mathbf{x}_{sample}) \tag{7}$$

$$= \arg\min_{\mathbf{x}\in\mathcal{T}} ||\mathbf{x} - \mathbf{x}_{sample}||_2. \tag{8}$$

In the absence of kinodynamic constraints, choosing the active node via minimising the Euclidean distance is guaranteed to provide the lowest cost path to the sampled coordinate. However, under kinodynamic constraints, this is often not the case and a high cost may be required to transition coordinates close to Euclidean space. We illustrate in fig. 1a an example of coordinates that are close in Euclidean distance but are relatively costly to reach under the Dubins car dynamics. In later sections, we shall empirically demonstrate that in many common kinodynamic planning tasks, the nearest node, according to Euclidean distance, is frequently costly to transition to.

B. Transition Cost Prediction for Tree Expansion

Would it then be possible to find the nearest point, being the node on the existing tree $\mathcal{T}$, which can reach $\mathbf{x}_{sample}$ while incurring the lowest cost? Computing the (lowest) cost to kinodynamically feasibly transition between two coordinates in the absence of obstacles requires solving the optimal cost kinodynamic planning problem between nodes in the tree and the sampled coordinate. Finding the nearest node based on transition cost requires us to solve the constrained optimisation problem:

$$\mathbf{x}_{active} = \arg\min_{\mathbf{x}\in\mathcal{T}} \texttt{TransitionCost}(\mathbf{x}, \mathbf{x}_{sample}), \tag{9}$$

$$\texttt{TransitionCost}(\mathbf{x}, \mathbf{x}_{sample}) = \min_{\mathbf{u}\in\mathcal{U}} C(\mathbf{x}, \mathbf{x}_{sample}) \tag{10}$$

$$\text{s.t. } \mathbf{x}(0) = \mathbf{x},\ \mathbf{x}(t_{max}) = \mathbf{x}_{sample}, \tag{11}$$

$$\mathbf{x}(t) \in \mathcal{K} \subseteq \mathcal{X}, \tag{12}$$

$$\mathbf{x}'(t) = f(\mathbf{x}(t), \mathbf{u}(t)), \forall t \in [0, t_{max}]. \tag{13}$$

In general, finding the cost of a feasible trajectory between nodes in the existing tree and the newly sampled node cannot be efficiently solved online. We propose to approximate instead the obstacle-free transition cost function in eq. (10) via offline learning with a function approximator and minimise the approximate cost online to select the active node. Importantly, as the cost function factors in the kinodynamic constraints but not the environment-specific collision-avoidance constraints, the trained model is specific to the system dynamics and does not need to be tuned when planning in different environments. We use a neural network, denoted as $g_{\boldsymbol{\theta}}$, with parameters $\boldsymbol{\theta}$ as the function approximator. Finding the nearest node by the approximate transition cost is then given as the unconstrained:

$$\mathbf{x}_{active} = \arg\min_{\mathbf{x}\in\mathcal{T}} g_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{x}_{sample}). \tag{14}$$

$$g_{\boldsymbol{\theta}} \approx \texttt{TransitionCost}((\mathbf{x}, \mathbf{x}_{sample}). \tag{15}$$

This can be solved very efficiently, as it only requires one

batched query of $g_{\boldsymbol{\theta}}$, then selecting the $\arg\min$ from the outputs. The batched querying of neural networks can be performed efficiently in parallel on a GPU. Empirically, we found the run-time of finding $\mathbf{x}_{active}$ via eq. (15) to be roughly twice as long as finding $\mathbf{x}_{active}$ via eq. (8), as is done in existing planning methods. It is important to note that our approach continues to maintain almost-surely asymptotic optimality guarantee for our various optimal planners as our procedure does not alter their subroutine optimality convergence. We had omitted the discussion in this work for brevity.

C. Training the Function Approximator

As the kinematic constraints are not tied to a specific environment, we can train the function approximator offline $g_\theta : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$ to approximate the transition cost. We collect a dataset with n data examples $\mathcal{D} = \{(\mathbf{x}^i_{start}, \mathbf{x}^i_{goal}), c^i\}_{i=1}^n$, where c is the cost to the goal, $\mathbf{x}_{goal}$, from the start $\mathbf{x}_{start}$. This dataset can come either from offline simulation or by observing planning runs online in a life-long learning manner. As stated above, we use a neural network as the function approximator $g_{\boldsymbol{\theta}}$. Neural networks are universal function approximators [29], and have been widely used on learning problems involving large and high-dimensional datasets. We train the neural network via mean squared error loss:

$$\mathcal{L}(\boldsymbol{\theta}) = ||c - g_{\boldsymbol{\theta}}(\mathbf{x}_{start} \oplus \mathbf{x}_{goal})||_2^2, \tag{16}$$

where $\mathbf{x}_{start} \oplus \mathbf{x}_{goal}$ denotes the vector concatenation of the start and goal coordinates. The model $g_{\boldsymbol{\theta}}$ can then be trained via gradient descent optimisers.

Typically, the dataset $\mathcal{D}$ is collected from a single system, and the learned transition cost predictor will be specific to the dynamics model and kinematics constraints of that system. Importantly, we can learn a transition cost predictor to a class of dynamic models with varying kinematic constraints by conditioning on the limit values of the constraints. For example, we can learn a cost predictor model for Dubins car models with various turn rate limits, which we denote as Ω. We then augment the input space of the neural network by concatenating the turn rate, i.e. $g_{\boldsymbol{\theta}}(\mathbf{x}_{start} \oplus \mathbf{x}_{goal} \oplus \Omega)$. The results are shown in table I.

V. Experimental Results

We extensively validate L4KDE for tree expansion using multiple dynamics models; the model dynamics examined include: (1) Dubins car dynamics with condition-able turn rate; (2) Double pendulum dynamics; and (3) Gooseneck trailer dynamics.

The **Dubins car** dynamics are given by $\mathbf{x} = [x, y, \theta]$ and $\mathbf{u} = [u_s, u_\phi]$, where $\dot{x} = u_s \cos\theta$, $\dot{y} = u_s \sin\theta$, $\dot{\theta} = u_s / L_{\text{axles}} \tan u_\phi$, and L_{axles} denotes the distance between the front and rear axles. The **Double Pendulum** dynamics are given by $\mathbf{x} = [\theta_1, \omega_1, \theta_2, \omega_2]$ and $\mathbf{u} = [u_{\alpha_1}, u_{\alpha_2}]$, where $\dot{\theta}_i = \omega_i$, $\dot{\omega}_i = u_{\alpha_i}/(mL_i^2) - g \sin\theta_i / L_i$, and L_i denotes the length of the i^{th} arm. The **Gooseneck Trailer** dynamics are given by $\mathbf{x} = [x, y, \theta_1, \theta_2]$ and $\mathbf{u} = [u_s, u_\phi]$, where $\dot{x} = u_s \cos\theta$, $\dot{y} = u_s \sin\theta$, $\dot{\theta}_1 = u_s / L_1 \tan u_\phi$, $\dot{\theta}_2 = u_s / L_2 \sin(\theta_1 - \theta_2)$, and L_1, L_2 denote the axles distance.

The L4KDE tree-expansion method can be used within a suite of widely-used tree-base kinodynamic motion planners; these include: (1) RRT [30]; (2)Anytime RRT [31]; (3) SST [2]; (4) SST* [2]; and (5) AO-RRT [1]. We additionally compared our planner with (6) AO-EST [1] which is incompatible with L4KDE because its tree-expansion mechanism measures the space's expansiveness.

All experiments are repeated 20 times, with a maximum run-time of 30 seconds for the Dubins vehicle, 3 minutes for the double pendulum and 5 minutes for the gooseneck trailer dynamics. All our models use the same neural network architecture: state space input dimension, followed by 5 fully connected layers, each with ReLU activation functions in between, and a Sigmoid as the output activation function. During training, we scale the target cost to cover the full range of the Sigmoid layer's output. The metrics used in our experiments are (1) *time-to-solution*: the time, in seconds, taken to find a feasible solution; (2) *success percentage*: the percentage of runs where feasible solutions are found.

A. Dubins Car: Conditioning on Constraint Limit Values

We wish to investigate whether L4KDE can generalise over an entire class of dynamics models and predict the transition cost of Dubins car models with different kinematic constraints. We consider Dubins car models with different turn rate constraints, where $|u_\phi| \leq \arctan(L_{\text{axles}} \cdot \Omega)$. The bounds on turn rate $\frac{\mathrm{d}\theta}{\mathrm{d}s}$ are denoted as Ω. We collect transition cost data from Dubins car models where the turning rate limits, Ω, are 15 equi-distance values between 0.5 and π. We then run L4KDE within the tree-based motion planners, where $\Omega = \{0.5, 1.38, 2.26, \pi\}$. An illustration of the test environment, along with an example solution (in red) and the tree (in grey), is given in fig. 4a. The results of using L4KDE, as compared to using Euclidean distance to select nearest nodes, within tree-based approaches is tabulated in table I, and the shortest time-to-solution plan for each Ω is given in bold. We observe that for different turn rate limits and over different motion planning methods, using L4KDE for tree expansion gives consistently shorter time-to-solution values and higher success percentages. The success percentages of each tree-based motion planner for each Ω, over different run-times, are shown in fig. 2. We observe that the L4KDE variants of motion planning algorithms have a higher success rate at almost any given time than the standard variants.

B. Gooseneck Trailer and Double Pendulum: L4KDE for Highly Challenging Systems

We validate that L4KDE improves the quality of the tree expansion in the challenging gooseneck trailer and double pendulum tasks. An outline of the environmental obstacle setup is shown in fig. 4b and fig. 4c. The time-to-solution and success percentage results are tabulated in table II. We see that for each of the different motion planning algorithms, using L4KDE significantly improves performance, often reducing the time-to-solution by more than half. The percentage of success over time for each method is illustrated in fig. 3. We observe that L4KDE variants of the suite of motion planning

TABLE I: Time-to-solution and success percentage for the Dubins Car turn-rate experiments.

	Motion Planner	Dubins Car Maximum turning rate ($\mu \pm \sigma$)							
		$\Omega = 0.5$		$\Omega = 1.38$		$\Omega = 2.26$		$\Omega = \pi$	
		Standard	with L4KDE	Standard	with L4KDE	Standard	with L4KDE	Standard	with L4KDE
Time-to-sol.	RRT	10.84 ± 7.55	7.98 ± 6.81	4.07 ± 4.02	2.38 ± 1.75	2.66 ± 3.34	0.96 ± 0.93	1.39 ± 0.95	2.05 ± 1.79
	AO-RRT	17.99 ± 3.89	7.45 ± 7.86	8.75 ± 9.27	2.74 ± 1.82	4.29 ± 5.17	**0.64 ± 0.40**	3.58 ± 3.01	**0.71 ± 0.60**
	Anytime-RRT	15.31 ± 4.65	**6.95 ± 4.58**	5.84 ± 4.52	2.35 ± 1.85	3.63 ± 2.68	1.33 ± 1.07	3.93 ± 3.05	2.26 ± 2.04
	SST	-	9.05 ± 5.81	6.40 ± 6.32	**1.31 ± 0.97**	5.52 ± 6.54	1.59 ± 1.34	3.86 ± 4.03	1.59 ± 1.69
	SST*	-	20.50 ± 5.64	8.29 ± 6.43	3.33 ± 2.57	4.91 ± 4.52	1.37 ± 1.58	3.76 ± 4.50	2.10 ± 1.76
	AO-EST	-	N/A	20.70 ± 5.41	N/A	17.23 ± 5.68	N/A	11.84 ± 5.95	N/A
Success pct.	RRT	65.0%	70.0%	100.0%	100.0%	100.0%	100.0%	85.0%	100.0%
	AO-RRT	10.0%	85.0%	95.0%	100.0%	95.0%	100.0%	95.0%	100.0%
	Anytime-RRT	40.0%	50.0%	80.0%	100%	90.0%	100%	95.0%	100%
	SST	0.0%	85.0%	95.0%	100%	100.0%	100.0%	95.0%	100.0%
	SST*	0.0%	45.0%	85.0%	100.0%	100.0%	100.0%	100.0%	100.0%
	AO-EST	0.0%	N/A	75.0%	N/A	90.0%	N/A	95.0%	N/A

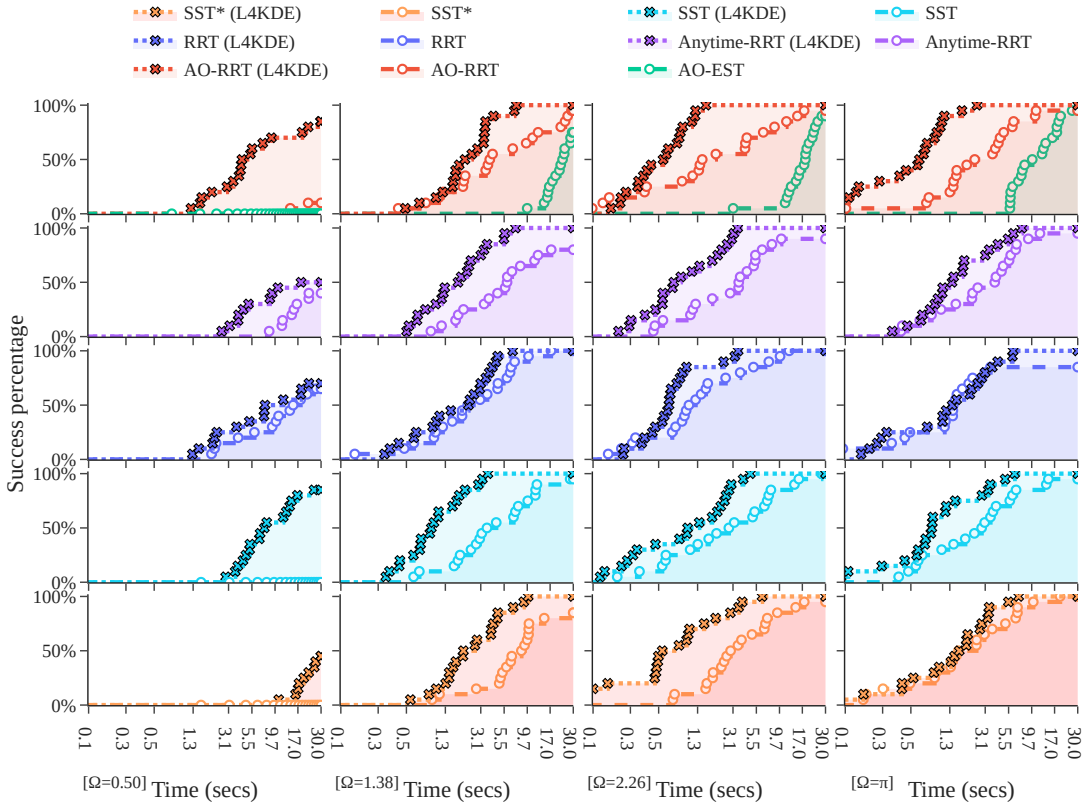


Fig. 2: Success percentage for various max turn-rate Ω in the Dubins car environment.

TABLE II: Time-to-solution ($\mu \pm \sigma$) and success percentage for the double pendulum and beetle gooseneck trailer

	Motion Planner	Double Pendulum	
		Standard	with L4KDC
Time-to-sol.	RRT	53.58 ± 44.76	20.58 ± 24.21
	AO-RRT	54.23 ± 27.69	23.99 ± 20.06
	Anytime-RRT	50.94 ± 29.90	**16.76 ± 10.21**
	SST	55.41 ± 33.71	28.52 ± 34.72
	SST*	63.04 ± 37.16	36.97 ± 19.31
	AO-EST	119.39 ± 28.57	N/A
Success pct.	RRT	100.0%	100.0%
	AO-RRT	80.0%	100.0%
	Anytime-RRT	95.0%	100.0%
	SST	100.0%	100.0%
	SST*	100.0%	100.0%
	AO-EST	35.0%	N/A

	Motion Planner	Beetle Gooseneck Trailer	
		Standard	with L4KDC
Time-to-sol.	RRT	171.76 ± 116.88	95.04 ± 94.92
	AO-RRT	293.50 ± 28.35	201.93 ± 108.89
	Anytime-RRT	126.43 ± 108.44	**65.31 ± 68.08**
	SST	131.50 ± 110.57	95.17 ± 112.39
	SST*	177.74 ± 115.96	167.84 ± 121.38
	AO-EST	281.91 ± 56.69	N/A
Success pct.	RRT	65.0%	90.0%
	AO-RRT	5.0%	55.0%
	Anytime-RRT	80.0%	95.0%
	SST	85.0%	80.0%
	SST*	55.0%	60.0%
	AO-EST	10.0%	N/A

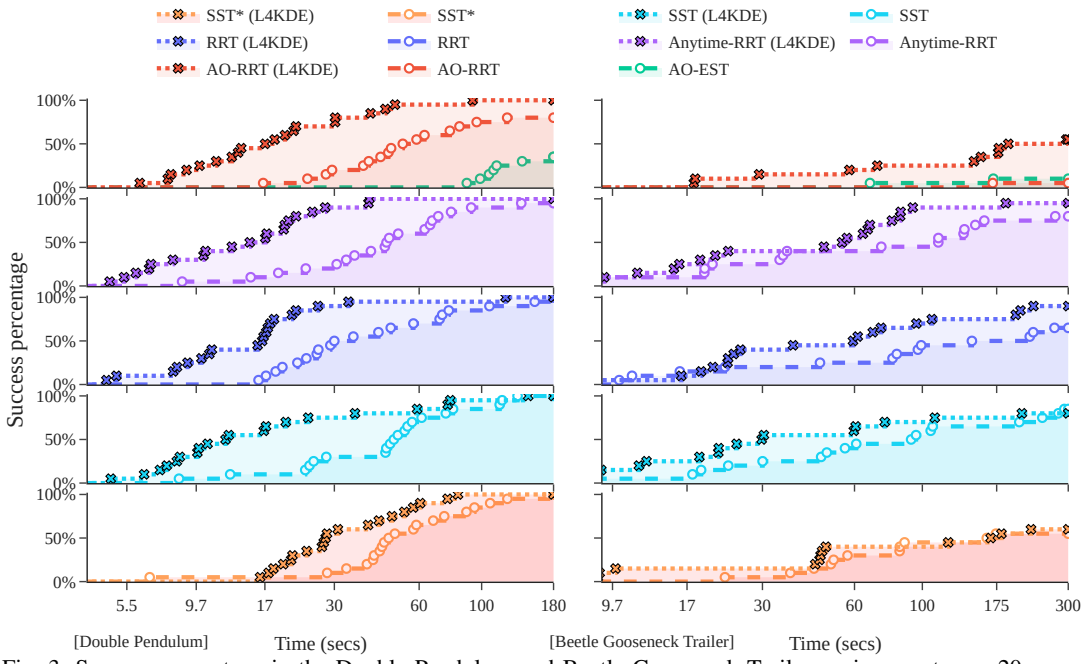
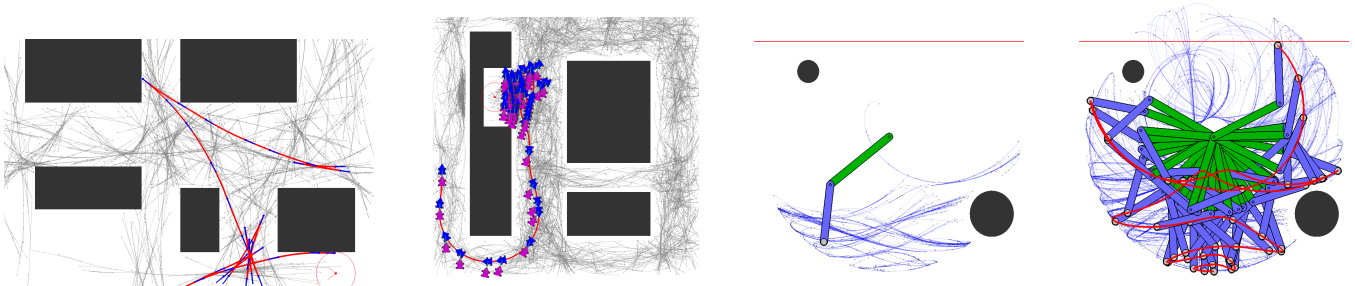


Fig. 3: Success percentage in the Double Pendulum and Beetle Gooseneck Trailer environment over 20 runs.



(a) Dubins car. Red circle denotes goal

(b) Gooseneck trailer. Red circle denotes goal

(c) DP during planning. Red line denotes goal

(d) DP with a successful trajectory

Fig. 4: Planning environments. Solution trajectories are shown in red; tree in grey ((a) and (b)) and blue ((c) and (d)); obstacles in black. Subfigure (c) and (d) illustrate the Double Pendulum (DP) problem with obstacles.

algorithms have a consistently higher success percentage over almost all points in time for both challenging problem setups.

C. Offline Training time vs Cumulative Performance Improvement

Training of the predictive model g_θ can be accomplished offline and as such is not time-critical to online planning performance. As such, we can characterise training as a "sunk cost" before the online motion planning. We wish to contextualise the benefit of L4KDE with respect to the time that it took to fully train our predictive model. In fig. 5, we record our model's training and testing loss against wall time for the double pendulum problem. We then run multiple planning episodes using AO-RRT with and without L4KDE on the double pendulum problem setup, and record the cumulative time-to-solution differences between the two. All procedures are performed on the same hardware. We found that training our model can fully converge within one minute. This is the one-time cost of training. Our sunk cost (in wall-time) can then be recovered within 1-2 episodes, and thereafter, the cumulative benefit of using L4KDE grows with every additional planning episode. This illustrates the simple yet powerful cumulative benefit of utilising our learning-based method in kinodynamic problems. The sunk cost of training the predictive model can be recovered after a fairly small number of motion planning episodes.

VI. Conclusion

We integrate neural networks into kinodynamic motion planning. We introduce **L**earning for **K**ino**D**ynamic Tree **E**xpansion (L4KDE) method for kinodynamic tree-based planning. Existing kinodynamic tree-based methods use Euclidean distance as a proxy for transition cost when selecting the nearest nodes on the tree. Using Euclidean distance for node selection is fast but often leads to suboptimal nodes being selected. Instead, we propose to train a neural network to predict the transition cost online efficiently. Our model is able to condition the parameters and limits of the dynamics model, enabling us to predict the costs over an entire class of models rather than a single model. We demonstrate that utilising

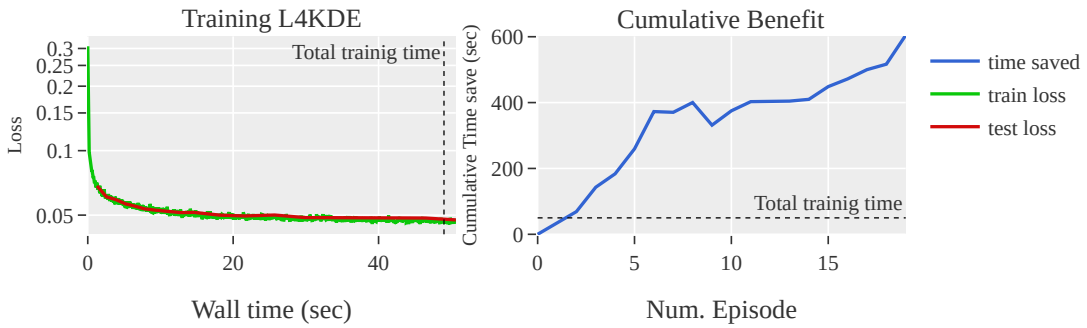


Fig. 5: The cumulative benefit of using L4KDE is that it accelerates planning after passively observing the transitions of system dynamics. Left plot shows the training loss against time in the *double pendulum problem*. Right plot illustrates the corresponding time saved (cumulative differences of time-to-solution, $y_k = \sum_i^k (t_{\text{standard},i} - t_{\text{L4KDE},i})$) when using L4KDE for the tree expansion.

L4KDE for tree expansion provides significant performance improvements in terms of success rate for a given time budget and time to solution within a suite of state-of-the-art asymptotically optimal, tree-based motion planning methods. We additionally show that the upfront cost of training our model quickly pays for itself after only a few online motion planning episodes in terms of overall compute time saved.

REFERENCES

[1] K. Hauser and Y. Zhou, "Asymptotically optimal planning by feasible kinodynamic planning in a state–cost space," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1431–1443, 2016.

[2] Y. Li, Z. Littlefield, and K. E. Bekris, "Asymptotically optimal sampling-based kinodynamic planning," *The International Journal of Robotics Research*, vol. 35, no. 5, pp. 528–564, 2016.

[3] B. Donald, P. Xavier, J. Canny, and J. Reif, "Kinodynamic motion planning," *Journal of the ACM (JACM)*, vol. 40, no. 5, pp. 1048–1066, 1993.

[4] M. Likhachev and D. Ferguson, "Planning long dynamically feasible maneuvers for autonomous vehicles," *The International Journal of Robotics Research*, vol. 28, no. 8, pp. 933–945, 2009.

[5] S. Kuindersma, F. Permenter, and R. Tedrake, "An efficiently solvable quadratic program for stabilizing dynamic locomotion," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2589–2594, IEEE, 2014.

[6] J. T. Betts, "Survey of numerical methods for trajectory optimization," *Journal of guidance, control, and dynamics*, vol. 21, no. 2, pp. 193–207, 1998.

[7] W. Zhi, T. Lai, L. Ott, and F. Ramos, "Diffeomorphic transforms for generalised imitation learning," in *Proceedings of The 4th Annual Learning for Dynamics and Control Conference*, 2022.

[8] W. Zhi, I. Akinola, K. Van Wyk, N. D. Ratliff, and F. Ramos, "Global and reactive motion generation with geometric fabric command sequences," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[9] D. Verscheure, B. Demeulenaere, J. Swevers, J. De Schutter, and M. Diehl, "Time-optimal path tracking for robots: A convex optimization approach," *IEEE Transactions on Automatic Control*, vol. 54, no. 10, pp. 2318–2327, 2009.

[10] M. Elbanhawi and M. Simic, "Sampling-based robot motion planning: A review," *IEEE Access*, vol. 2, pp. 56–77, 2014.

[11] S. M. LaValle *et al.*, "Rapidly-exploring random trees: A new tool for path planning," tech. rep., Computer Science Department, Iowa State University, 1998.

[12] S. R. Lindemann and S. M. LaValle, "Incrementally reducing dispersion by increasing voronoi bias in rrts," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings.*, vol. 4, pp. 3251–3257, IEEE, 2004.

[13] T. Lai, F. Ramos, and G. Francis, "Balancing Global Exploration and Local-Connectivity Exploitation with Rapidly-Exploring Random disjointed-Trees," in *Proceedings of The International Conference on Robotics and Automation*, IEEE, 2019.

[14] L. E. Kavraki, P. Svestka, J. C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, pp. 566–580, Aug. 1996.

[15] T. Lai and F. Ramos, "LTR*: Rapid Replanning in Executing Consecutive Tasks with Lazy Experience Graph," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 8784–8790, IEEE, 2022.

[16] B. Ichter, J. Harrison, and M. Pavone, "Learning sampling distributions for robot motion planning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7087–7094, IEEE.

[17] D. Berenson, P. Abbeel, and K. Goldberg, "A robot path planning framework that learns from experience," in *2012 IEEE International Conference on Robotics and Automation*, pp. 3671–3678, IEEE.

[18] D. Coleman, I. A. Şucan, M. Moll, K. Okada, and N. Correll, "Experience-based planning with sparse roadmap spanners," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 900–905, IEEE.

[19] A. Perez, R. Platt, G. Konidaris, L. Kaelbling, and T. Lozano-Perez, "LQR-RRT*: Optimal sampling-based motion planning with automatically derived extension heuristics," in *2012 IEEE International Conference on Robotics and Automation*, pp. 2537–2542, IEEE, 2012.

[20] T. Lai, P. Morere, F. Ramos, and G. Francis, "Bayesian Local Sampling-Based Planning," vol. 5, no. 2, pp. 1954–1961, 2020.

[21] T. Lai and F. Ramos, "Adaptively Exploits Local Structure With Generalised Multi-Trees Motion Planning," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 1111–1117, 2021.

[22] A. Orthey and M. Toussaint, "Rapidly-Exploring Quotient-Space Trees: Motion Planning using Sequential Simplifications,"

[23] J. Lee, O. Kwon, L. Zhang, and S.-e. Yoon, "SR-RRT: Selective retraction-based RRT planner," in *Proc. of ICRA*.

[24] T. Lai and F. Ramos, "Plannerflows: Learning motion samplers with normalising flows," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2542–2548, IEEE, 2021.

[25] T. Lai, W. Zhi, T. Hermans, and F. Ramos, "Parallelised diffeomorphic sampling-based motion planning," in *5th Annual Conference on Robot Learning*, 2021.

[26] T. Y. Lai, *Robot Learning and Planning with a Probabilistic Perspective*. PhD thesis, The University of Sydney, 2023.

[27] T. Kunz and M. Stilman, "Probabilistically complete kinodynamic planning for robot manipulators with acceleration limits," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3713–3719, IEEE, 2014.

[28] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[29] S. Sonoda and N. Murata, "Neural network with unbounded activation functions is universal approximator," *Applied and Computational Harmonic Analysis*, vol. 43, no. 2, pp. 233–268, 2017.

[30] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *The international journal of robotics research*, vol. 20, no. 5, pp. 378–400, 2001.

[31] D. Ferguson and A. Stentz, "Anytime rrts," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5369–5375, IEEE, 2006.