DynoFluxBench: Benchmarking Kinodynamic Space-Time Planners in Dynamic Environments

Franz Queißner[1], Andreas Orthey[1], Wolfgang Hönig[1]

***Abstract*— Robots that leave structured, static environments must plan motions that are kinodynamically feasible and safe among moving obstacles. However, there are no dedicated benchmark frameworks that combine both aspects. To overcome this, we present DynoFluxBench, a framework to compare kinodynamic planners in known, dynamic environments with unbounded arrival time. To demonstrate its utility and establish strong baselines, we develop three dedicated planners, named ST-Db-RRT, ST-GBRRT, and KIST, that fuse kinodynamic and space-time methods, covering different kinodynamic search paradigms: ST-Db-RRT expands with randomly selected discontinuity-bounded motion primitives using trajectory optimization, whereas KIST and ST-GBRRT maintain a kinodynamically feasible tree with different heuristic guidance. We analyze the probabilistic completeness guarantees of those new planners in dynamic environments. Finally, we evaluate ST-Db-RRT, ST-GBRRT, and KIST using DynoFluxBench, showing that ST-Db-RRT reaches a first solution up to 32 times faster, while KIST and ST-GBRRT remain valuable where trajectory optimization is fragile. Videos and further analysis can be found at `https://dynofluxbench.github.io/dynofluxbench/`.**

I. Introduction

Planning feasible kinodynamic motions and avoiding dynamic obstacles have both been studied intensively, but largely in isolation. The first is addressed by *kinodynamic* motion planning [1], [2], [3], which takes the kinematic and differential constraints of the robot into account, such as the turning rate of a car or the limited acceleration of a torque-constrained system. The second is addressed by planning in *known dynamic environments* [4], [5], [6]: when the trajectories of moving obstacles are known in advance, as, for example, in multi-robot planning, we can treat time as an additional dimension, and the planner can anticipate the motion of the obstacles [6].

Recent kinodynamic planners such as Iterative Discontinuity-bounded Rapidly-exploring Random Tree (iDb-RRT) [7] and Generalized Bidirectional RRT (GBRRT) [8] find kinodynamically feasible trajectories for a wide range of systems, but only in static environments. By contrast, Space-Time-RRT* (ST-RRT*) [6] plans optimally through space-time with moving obstacles and unknown arrival time, but only for holonomic or steering-capable systems. While kinodynamic planners for dynamic environments exist [9], [10], they require a steering function and do not address problems with unbounded time. To the best of our knowledge, there exists no kinodynamic motion planner for general dynamical systems that simultaneously avoids steering functions and plans through space-time with dynamic obstacles and unknown, unbounded arrival time.

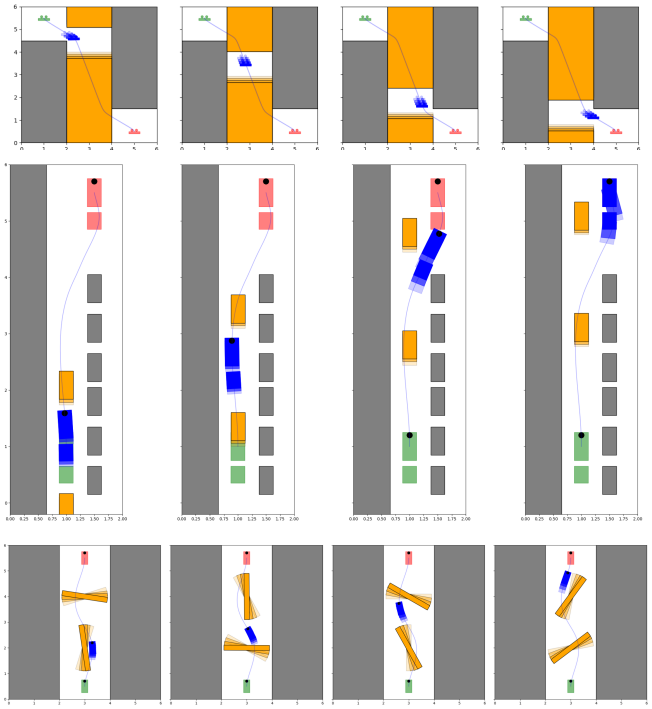


Fig. 1. Example kinodynamic trajectories in dynamic environments from DynoFluxBench. **Top:** Planar rotor timing its passage through a moving elevator gap. **Middle:** Car performing a lane switch among moving traffic. **Bottom:** Second-order unicycle traversing two revolving doors. Each row shows four snapshots along the planned space-time path (green: start, red: goal, orange: dynamic obstacles).

To combine both aspects, we develop three new planners, ST-Db-RRT, ST-GBRRT, and KIST, that plan for kinodynamic systems in space-time for dynamic environments *without requiring a steering function or a known arrival time*. All three planners share the same space-time search strategy but apply it to different kinodynamic search paradigms. Beyond the algorithms themselves, we analyze the theoretical consequences of the space-time extension, showing that probabilistic completeness of GBRRT is preserved, but the resolution completeness of Db-RRT is only preserved with bounded time. Those three planners are used as baselines for DynoFluxBench, a novel benchmark which evaluates kinodynamic space-time planners on a set of dynamic environments (Fig. 1). To summarize, we make the following contributions:

- **DynoFluxBench:** Create a novel benchmark set to eval-

[1]TU Berlin, Germany

uate kinodynamic space-time planners on 25 scenarios involving four dynamical systems.
- **Three Planners:** Extension of existing kinodynamic planners iDb-RRT [7] and GBRRT [8] to space-time obtaining ST-Db-RRT and ST-GBRRT, respectively. Additionally, we develop KIST, a combination of different aspects of ST-Db-RRT and ST-GBRRT.
- **Theoretical Analysis:** Analysis of the algorithmic properties of Naive Db-RRT, ST-Db-RRT, ST-GBRRT, and KIST pertaining to dynamic environments.

II. Related Work

Motion planning in dynamic environments and kinodynamic motion planning are well studied fields. In the following, we review both fields as well as the existing works at their intersection.

A. Planning in Dynamic Environments

Planning in dynamic environments is often approached reactively, without assuming prior knowledge of the obstacle trajectories. Execution-extended RRT [11] reuses waypoints of previous solutions to speed up replanning, Real-Time RRT* [12] interleaves online tree rewiring with path execution, and RRTX [10] maintains an asymptotically optimal search tree that is repaired online.

When the obstacle trajectories are known in advance, the motion of obstacles can be anticipated. Path-velocity decomposition [13] does so in two stages, planning a geometric path first and a velocity profile along it that avoids the moving obstacles. Committing to a fixed path early, however, can render every velocity profile infeasible. A more general approach is to treat time as an explicit planning dimension by planning in space-time [4]. Time-Based RRT [5] leverages space-time to plan a rendezvous of two dynamic systems with known arrival time. SIPP [14] compresses the time dimension into safe intervals and computes optimal trajectories, but requires a discretization of the state space, restricting it to low-dimensional problems. Sampling-based planners like T-PRM [15] avoid this state space discretization by augmenting the roadmap with a time dimension, while SI-RRT [16] transfers the safe-interval representation into an RRT-Connect search for high-DoF manipulators, an idea also applied to multi-robot planning [17]. Both, however, are restricted to holonomic systems. ST-RRT* [6] lifts RRT* into space-time without assuming prior knowledge of the arrival time, planning asymptotically optimal trajectories in dynamic environments. However, its extension step interpolates linearly between states, restricting it to holonomic systems with velocity limits, and kinodynamic constraints cannot be handled.

B. Kinodynamic Motion Planning

Kinodynamic motion planning accounts for differential constraints and actuation limits. There are mainly three categories of approaches: Search-based methods discretize the control space with kinodynamically feasible motion primitives [18] to construct a graph of reachable states, which can then be searched with search algorithms like A*. This approach provides strong theoretical guarantees relative to the chosen discretization, but the size of the graph grows exponentially with the state dimension, making exploration of high-dimensional state spaces infeasible.

Sampling-based planners explore the state space more rapidly by growing a tree [1] towards randomly sampled states using propagated controls, while preserving completeness under correct propagation [19]. RRT methods can achieve asymptotic optimality by planning in state-cost space (AO-RRT) [20], using witness sets (SST*) [3] or rewiring the tree (RRT*) [21], [10], [6]. Rewiring, however, requires connecting two states exactly which in the kinodynamic case can be described by a two-point boundary value problem (BVP) that can be solved in closed form only for restricted system classes such as linear dynamics [22], and is computationally expensive or unsolvable for general kinodynamic systems. Heuristics can be used to improve sampling with informed sets [23] or AIT* [24], which computes a cost-to-go heuristic through an improving reverse search tree. GBRRT [8] makes this feasible for the general kinodynamic setting by avoiding the requirement to solve any BVP using a heuristic-guided bidirectional search with forward propagation.

Optimization-based methods such as CHOMP [25] or sequential convex optimization [26] optimize trajectories while incorporating complex constraints well. Their performance is highly dependent on an initial guess and suffers from local minima [27]. T-CHOMP [28] extends the CHOMP [25] trajectory optimization framework to space-time, enabling it to handle time-dependent constraints, but requires a prior fixed trajectory duration. More recently, hybrid methods like db-A* [29], [30] connect precomputed motion primitives while tolerating a bounded discontinuity at the junctions, which is subsequently repaired by trajectory optimization to an optimal solution. iDb-RRT [7] embeds this discontinuity-bounded expansion into the RRT framework, removing the need to solve a BVP. These planners efficiently solve kinodynamic problems in static environments. Time, however, is not an explicit planning dimension, and moving obstacles cannot be accounted for.

C. Kinodynamic Planning in Dynamic Environments

A smaller body of work addresses both challenges simultaneously. One of the earliest randomized kinodynamic planners [31] builds a tree of milestones in state-time space, avoiding moving obstacles with known trajectories. While general with respect to the system dynamics, the planner expands only at random within a bounded time horizon, without goal-directed guidance and without any notion of solution quality. Chen et al. [32] extend the Space Exploration Guided Heuristic Search to the time domain, guiding a kinodynamic search with a time-dependent heuristic. The approach targets car-like robots and relies on a circle-based decomposition of the workspace. More recently, safe intervals [14] have been combined with kinodynamic models [33], [34], which retain completeness and optimality. However, each of those works

is either restricted to a specific class of systems or depends on a discretization of the state space.

In summary, the reviewed planners for dynamic environments handle moving obstacles but assume holonomic systems, kinodynamic planners handle differential constraints but assume static environments, and the few works at their intersection are tied to specific system classes or state-space discretizations. We address this integrated problem with sampling-based planners that combine kinodynamic planning in dynamic environments with an unknown arrival time.

III. Problem Statement

Let $\mathcal{X} = \mathcal{Q} \times \mathcal{T}$ be a space-time state space [6], which pairs the configuration state space $\mathcal{Q}$ with the time axis $\mathcal{T}$. A space-time state $x = (q,\, t) \in \mathcal{X}$ therefore consists of a configuration state $q \in \mathcal{Q}$ and a time $t \in \mathcal{T}$. The system dynamics act on the configuration component only: applying a control $u \in \mathcal{U}$ for one timestep advances it by $q_{k+1} = step(q_k,\, u_k)$ following an Euler integration with a known system velocity limit $\dot{q}_{\max}$, while the time component advances deterministically by the fixed step Δt. The resulting space-time transition is

$$x_{k+1} = (q_{k+1},\, t_{k+1}) = \big(\, step(q_k,\, u_k),\; t_k + \Delta t \,\big). \quad (1)$$

Let $\mathcal{X}_{\text{free}}(k) \subseteq \mathcal{X}$ be the collision-free states at time k, x_{start} the start state, and $\mathcal{X}_{\text{goal}}$ the goal region. The lower-bound time to cover the distance between two configurations is denoted $\underline{t}(q_1,\, q_2)$. The desired output is a solution path $\mathcal{S} = (X, U)$ connecting x_{start} to the goal region $\mathcal{X}_{\text{goal}}$ with a state sequence $X = [x_{\text{start}}, x_1, \ldots, x_{\text{goal}}]$, $x_{\text{goal}} \in \mathcal{X}_{\text{goal}}$, of length K and a control sequence $U = [u_0, u_1, \ldots, u_{K-1}]$ such that $x_k \in \mathcal{X}_{\text{free}}(k)$. The cost of a feasible path is $c = K \cdot \Delta t$.

IV. Kinodynamic Space-Time Primitives

To plan in kinodynamic space-time, we use goal region expansion and distance computation from space-time planning [6] and discontinuity-bounded expansion with motion primitives for kinodynamic systems [7].

A. Goal Region Expansion

To handle unbounded space-time, we leverage the conditional sampling and the goal region expansion methods from ST-RRT* [6].

Conditional sampling samples a random configuration and then computes the lower-bound time to the start state, conditioning the sampled time on the reachability from the start. The upper time bound is defined by the last valid time the system could still reach the latest goal arrival time of all currently existing goals.

The goal region expansion uses the two parameters *initial batch size* and *range factor* to progressively update the goal region. In every iteration, we check if the current sampling batch is exhausted and if so, expand the goal region by updating the time bounds by the value of the *range factor*. Thus, the initial batch size controls how many samples are drawn in the time bounds before the goal region is expanded, and the range factor controls how much the goal region is expanded. When the planners find a first solution, the progressive goal region expansion is stopped. Instead, the upper time bound is set to the arrival time of the best current solution, and the lower time bound is set to the lower-bound time from start to goal. In this way, only samples that could improve the current best solution are drawn.

B. Distance Computation

The distance between states is defined by the space-time pseudometric [6]. This pseudometric consists of the intrinsic configuration distance $d_{\mathcal{Q}}$ and the temporal difference $d_{\mathcal{T}}$, which is weighted by $\lambda \in (0, 1)$. For two given states, *time-consistency* is enforced, which means a state may only be reached from an earlier state, and only if the velocity required to cover the configuration gap within the available time respects the limits $\dot{q}_{\max}$; states violating either condition are infinitely far apart.

C. Kinodynamic Expansion Modes

We leverage motion primitives to discretise the action space for fast kinodynamic expansion [29]. A primitive is defined as $m = (X, U)$ consisting of a kinodynamically feasible sequence of states $X = [x_0, x_1, \ldots, x_f]$ and actions $U = [u_0, u_1, \ldots, u_{f-1}]$. Discretising the action space into a finite set of primitives means that, in general, no primitive's start state x_0 exactly matches the state being expanded, which introduces a discontinuity between that state and the primitive. Allowing a discontinuity up to a certain bound δ makes a trajectory *δ-discontinuity-bounded*. We use $x_f = x \oplus m$ to indicate that the motion m is applied from a state x with a *δ-discontinuity-bound*, landing in a final state x_f.

To expand and connect states with primitives, we use one of three selection policies:

$$\sigma \in \{\textsc{Rand},\ \textsc{Best},\ \textsc{Prop}\}.$$

With $\sigma = \textsc{Rand}$, the first valid, randomly drawn primitive is applied as a δ-discontinuity-bounded jump $x \oplus m$. With $\sigma = \textsc{Best}$, the primitives are ordered by the distance of their landing state to the expansion target before the first valid one is applied, again as a discontinuity-bounded jump. With $\sigma = \textsc{Prop}$, the primitives are ordered as in BEST, but the actions of a candidate are forward propagated from the parent state with the system dynamics, resulting in a kinodynamically feasible motion without discontinuity.

By using the three selection policies, we can define both expansion and connection methods.

a) Expand: The EXPAND method grows a tree T towards a sampled state x_{rand}. It queries the $k = \log(|T|+1)$ nearest neighbors, and calls the selection policy for the actual expansion. The first successful new state is added to the tree and returned. If the neighborhood is exhausted, the expansion fails.

b) Connect: A connection attempts to greedily establish a connection between two states, by repeatedly expanding towards the target state using the selection policy, as long as the distance to the target decreases, checking every intermediate node.

V. Kinodynamic Space-Time Planners

By relying on the kinodynamic space-time primitives, we develop three novel planners by generalizing Db-RRT [7] and GBRRT [8]. To motivate this, we first discuss a naive Db-RRT planner.

A. Naive Db-RRT for Dynamic Environments

A first naive approach to elevate Db-RRT to deal with dynamic environments is to keep the original state space (without an explicit time dimension) and add dynamic collision checks to avoid moving obstacles. Since we know the duration of every motion primitive we can keep track of each node's time. This can be achieved by using the time as cost which makes the cost-to-come represent the time of each node. The durations of the discontinuities between the motion primitives are accounted for using the lower time bound function $\underline{t}(q_1, q_2)$ [30]. Knowing each node time enables time-dependent collision checks and allows the planner to find discontinuity-bounded solutions. As we show in Sec. VI, Naive Db-RRT is not probabilistically complete (PC).

B. Planner 1: ST-Db-RRT

To enable planning in space-time with Db-RRT, we develop the **S**pace-**T**ime **D**iscontinuity-**b**ounded RRT (ST-Db-RRT). ST-Db-RRT is a bidirectional tree planner, which uses the same sampling mechanism as ST-RRT* [6] based upon sampling in space-time and adaptive goal expansion. Contrary to ST-RRT*, it uses the fast randomized expansion mechanism $\sigma = \textsc{Rand}$ of Db-RRT [7] to expand both trees in space-time. To connect the two trees, the $\sigma = \textsc{Best}$ connection mode is used as discussed in Sec. IV. Once a connection is found, the discontinuity-bounded solution is extracted.

Repairing the discontinuity-bounded solution in dynamic environments requires a time-dependent trajectory optimization. We adopt the Differential Dynamic Programming (DDP) repair of Db-RRT to achieve this. DDP solves trajectory optimization problems of the form

$$\min_{Q,U} \sum_{k=0}^{K-1} c(q_k, u_k) + c_K(q_K) \tag{2a}$$

$$\text{s.t.} \quad q_{k+1} = step(q_k, u_k)\ \forall k \in \{0, \dots, K-1\} \tag{2b}$$

$$q_0 = q_{\text{start}}, \tag{2c}$$

and enforces constraints such as collision avoidance through a cost term evaluated at every iteration. We keep time separate from the state: it is not a decision variable but is passed to the cost function (Eq. 2a) solely to evaluate the collision constraint. The algorithm terminates if a planner termination condition is fulfilled.

C. Planner 2: ST-GBRRT

ST-GBRRT is a direct transfer of GBRRT [8] to dynamic environments. GBRRT introduced the idea of guiding a feasible forward tree towards the goal with a cost-to-goal heuristic taken from a reverse tree, so that the two trees never have to be connected and no boundary value problem needs to be solved.

We adapted GBRRT by growing the reverse tree with fast δ-discontinuity-bounded motion primitives instead of GBRRT's reverse propagation. The forward tree keeps GBRRT's feasible action rollouts, interleaving exploration of the state space with exploitation of the reverse-tree heuristic. Because the forward tree is kinodynamically feasible and grows directly into the goal region, ST-GBRRT returns a solution without any completion policy or trajectory optimization.

TABLE I

Overview of the presented planners and properties. The expansion and connection modes (Rand: random primitive; Best: best primitive; Prop: forward propagation of the best primitive); ST: plans in space-time; TO: uses trajectory optimization; RC: resolution complete with respect to the provided motion primitives; RC*: resolution complete with respect to the provided motion primitives, but not complete in unbounded time; PC: probabilistically complete;

			Completeness		
Planner	ST	TO	static	dynamic	Key features
Naive Db-RRT	×	✓	RC	×	time-annotated nodes, time-dependent collision checks
ST-Db-RRT	✓	✓	RC	RC*	Rand db-expansion, Best δ-connect in space-time
KIST	✓	×	PC	PC	Prop/Best expansion, completion policy after tree connection
ST-GBRRT	✓	×	PC	PC	Prop/Best expansion, continuous reverse-tree heuristic

D. Planner 3: KIST

To combine aspects of both planners, we develop **KIST**, a **KI**nodynamic **S**pace-**T**ime planning algorithm for dynamic environments. Similar to ST-Db-RRT, sampling follows ST-RRT* [6] and expansion follows the Db-RRT [7] method for both trees. However, similar to GBRRT [8], no connection between the trees is attempted. Instead, the trees are considered to be connected when they are in δ proximity, similar to Db-RRT. Instead of expanding with discontinuities, KIST avoids optimization by growing a feasible forward tree with primitive action rollouts. Additionally, it has a probability to perform a random action rollout that propagates a random action from $\mathcal{U}$ for a random duration $t \in [0, t_{\max}]$. The reverse tree grows with δ-discontinuity-bounded motion primitives, but after an approximate connection with the forward tree, the algorithm uses the path of the reverse tree solely as a guiding heuristic to expand the forward tree greedily to the goal. We use a greedy A*-like search. If this completion policy succeeds, the solution path is extracted.

VI. Theoretical Analysis

We next analyze the different properties of the algorithms. A complete overview is found in Tab. I. We show the key features of each planner, along with a description of whether

planning is done in space-time (ST), if trajectory optimization (TO) is used, and the respective completeness guarantees in static and dynamic environments. In this section, we provide the details on the probabilistic completeness (PC) guarantees in dynamic environments.

Naive Db-RRT: To start this off, we first state a negative result of PC for Naive Db-RRT.

Remark. *Naive Db-RRT is not PC in dynamic environments.*

Proof Sketch. Duplicate pruning ([7] Alg. 2, Line 14) of Db-RRT rejects a new node when a node would be placed in a δ-neighborhood, i.e., it suppresses nodes too close in state space. Consider a configuration q that must be occupied at time t_1 to pass through a door within a short temporal window. If one branch reaches close to q before t_1, duplicate pruning can suppress a later branch that would reach q at t_1 and thus will never find a solution. Without planning in space-time, the requirement "be at q, but at a later time" cannot be represented. In space-time, (q, t_0) and (q, t_1) are different states and duplicate pruning can distinguish between them, as the time is included in the distance metric. □

ST-Db-RRT: ST-Db-RRT does not suffer from the duplicate pruning problem that Db-RRT has in dynamic environments, as the pseudometric takes time into account. Planning in space-time, however, has other theoretical implications on PC. The proof of PC for iDb-RRT and the family of Db-Algorithms [29], [30], [7] uses the chain of balls argument [19]. A core requirement for the proof is a positive probability of reaching the next ball in the chain, which they show for random forward propagation in Lemma 3. The Db-Algorithms instead discretize the action space with motion primitives. Their proof iteratively adds motion primitives and decreases the discontinuity bound, so that eventually all motions required to reach the successor ball are considered. This assumes that adding motion primitives covers the state space asymptotically (see Def. 4-5 and Remark 3 in [30]). Their generation achieves this by randomly sampling start and goal states and using trajectory optimization, resulting in time-optimal motions. This is the point where the proof breaks in space-time, since covering space-time also requires time-suboptimal motions. The completeness argument therefore needs the primitive-generating method to place positive measure on time-suboptimal motions of varied duration. Random forward propagation with randomized duration satisfies this, achieving resolution completeness with respect to the generated motion primitives.

However, even with a primitive-generating method that covers space-time asymptotically, ST-Db-RRT is not complete, because the unbounded time dimension keeps it from reporting that no solution exists. Db-RRT can terminate and move to the next iteration of iDb-RRT with more motion primitives. ST-Db-RRT, on the other hand, never terminates and keeps searching the unbounded time dimension. A known or estimated upper bound on time could fix this. **ST-GBRRT**: ST-GBRRT is a direct transfer of GBRRT [8] to space-time, so we can adapt the original proof [8]. While the best-input selection of motion primitives is not complete, completeness is restored by the random action rollout. The space-time sampling strategy ensures that the goal region is expanded and that there is a positive probability of sampling a feasible goal arrival time. This ensures that ST-GBRRT retains PC in dynamic environments.

KIST: KIST resembles the structure of ST-RRT* at a high level, but uses a mixture of motion primitives and action rollouts for expansion. Its PC follows ST-RRT*'s proof sketch and the same chain of balls argument [19] as ST-Db-RRT (Sec. VI). As there, the best-input selection policy alone would lose completeness, since the limited set of primitives gives no guarantee of expanding into the next ball in space-time. We restore PC by expanding the forward tree with a random action rollout. Finally, in the case of unbounded goal time, we follow the argument of ST-RRT*: through progressive goal region expansion and conditional sampling, there is a positive probability of eventually sampling a feasible goal arrival time and any node that is part of a feasible solution.

VII. Benchmark Results

This section evaluates Naive Db-RRT and the three space-time planners on the DynoFluxBench set. The algorithms are implemented in *Dynoplan* [7], which provides the dynamical systems including motion primitives and infrastructure. For this work, we extended the existing infrastructure to deal with dynamic obstacles.

The evaluation is presented in three parts. First, we present a hyperparameter study to select a common parameter set. Second, we compare Db-RRT and ST-Db-RRT to show the influence of the time dimension. Third, we compare ST-Db-RRT, ST-GBRRT, and KIST in terms of success rate, time to first solution, and solution cost.

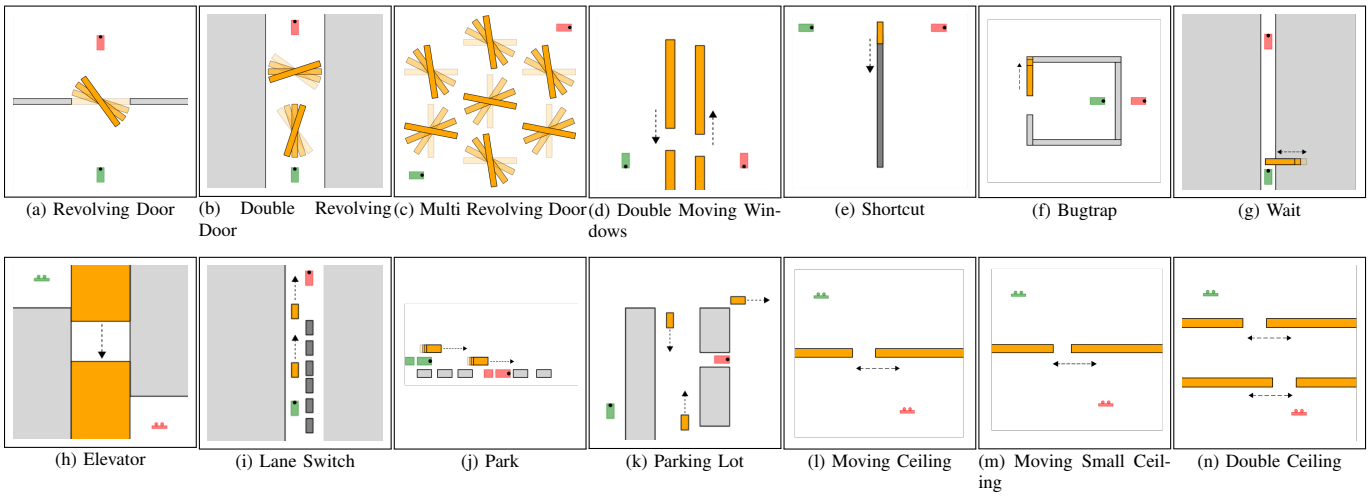
a) Hardware and Benchmark Parameters: Each individual run of a planner uses a single core of an AMD CPU running at 3.60 GHz with 128 GB RAM using Ubuntu 22.04. Each run has a time limit of $180\,\mathrm{s}$ and returns a solution cost c, the time to the first solution t_1, and the success rate sr for each algorithm. The results are given as averages over a set of $N = 20$ runs.

b) Environments: We use a suite of 15 environments comprising 25 instances across four dynamical systems, including 10 newly designed layouts and 4 dynamic adaptations of static environments from *Dynobench* [30] (Fig. 2 shows the 14 dynamic layouts; Empty is omitted).

c) Dynamical Systems: In the benchmarks, we use four dynamical systems. The first two are a 1st and 2nd order unicycle system. The third is a car-with-trailer model, which reflects a realistic driving system. Finally, we use a planar rotor as an underactuated flying system [7].

A. Planner Parameter Study

To choose the best set of parameters for ST-Db-RRT, ST-GBRRT, and KIST, we use DynoFluxBench to conduct a hyperparameter optimization on four parameters. First,



(a) Revolving Door (b) Double Revolving Door (c) Multi Revolving Door (d) Double Moving Windows (e) Shortcut (f) Bugtrap (g) Wait

(h) Elevator (i) Lane Switch (j) Park (k) Parking Lot (l) Moving Ceiling (m) Moving Small Ceiling (n) Double Ceiling

Fig. 2. Environments used in DynoFluxBench. The robot is shown in the start configuration (green) and the goal configuration (red). Static obstacles (grey) and dynamic obstacles (orange) are shown.

TABLE II

COMPARING ALL FOUR PLANNERS ON ALL ENVIRONMENTS. BOLD VALUES MARK THE BEST RESULT PER METRIC AMONG THE THREE SPACE-TIME PLANNERS (LOWEST COST AND TIME-TO-FIRST SOLUTION, HIGHEST SUCCESS RATE); NAIVE DB-RRT IS ALSO BOLDED WHEN IT MATCHES OR BEATS ST-DB-RRT ON THAT METRIC. DASHES INDICATE THAT NO SOLUTION WAS FOUND. GREY CELLS MARK NAIVE DB-RRT, WHICH IS NOT PROBABILISTICALLY COMPLETE.

#	Scenario	Dynamics	Naive Db-RRT			ST-Db-RRT			KIST			ST-GBRRT		
			c [s]	t_1 [s]	sr	c [s]	t_1 [s]	sr	c [s]	t_1 [s]	sr	c [s]	t_1 [s]	sr
01	Empty	1st Order Unicycle	15.5	**0.1**	**1.0**	14.7	**0.5**	**1.0**	15.1	1.1	**1.0**	**14.4**	1.9	**1.0**
02	Revolving Door	1st Order Unicycle	11.5	**0.2**	**1.0**	11.5	**1.1**	**1.0**	10.7	1.7	**1.0**	**10.5**	2.6	**1.0**
03	Double Revolving Door	1st Order Unicycle	13.8	**0.3**	**1.0**	13.4	**1.7**	**1.0**	12.8	2.8	**1.0**	**12.6**	5.7	**1.0**
04	Multi Revolving Door	1st Order Unicycle	**17.2**	**0.5**	**1.0**	17.7	**2.7**	**1.0**	18.4	5.5	**1.0**	**16.2**	10.6	**1.0**
05	Double Moving Windows	1st Order Unicycle	11.5	**0.2**	**1.0**	11.6	2.4	**1.0**	11.6	**2.3**	**1.0**	**10.6**	4.4	**1.0**
06	Shortcut	1st Order Unicycle	15.2	**0.1**	**1.0**	14.7	**0.8**	**1.0**	14.6	2.1	**1.0**	**14.1**	3.4	**1.0**
07	Bugtrap	1st Order Unicycle	**28.8**	**0.6**	**1.0**	**32.2**	**32.3**	**1.0**	41.1	42.3	**1.0**	–	–	0.0
08	Wait	1st Order Unicycle	–	–	0.0	39.4	**2.0**	**1.0**	**39.1**	32.2	0.7	–	–	0.0
09	Elevator	1st Order Unicycle	17.9	35.4	0.9	17.4	18.0	0.5	18.4	16.9	0.6	**16.6**	**15.4**	**1.0**
10	Empty	2nd Order Unicycle	21.7	**0.2**	**1.0**	19.6	**0.5**	**1.0**	19.9	1.3	**1.0**	**18.7**	44.8	0.9
11	Revolving Door	2nd Order Unicycle	13.4	**0.2**	**1.0**	12.7	**1.2**	**1.0**	13.2	3.8	**1.0**	**12.2**	10.7	**1.0**
12	Double Revolving Door	2nd Order Unicycle	22.4	**0.3**	0.8	**20.7**	**1.9**	**1.0**	23.3	5.4	**1.0**	–	–	0.0
13	Multi Revolving Door	2nd Order Unicycle	23.6	**0.5**	**1.0**	22.5	**2.5**	**1.0**	34.3	7.7	**1.0**	**21.5**	79.9	0.8
14	Double Moving Windows	2nd Order Unicycle	15.9	**0.3**	**1.0**	17.7	**3.5**	**1.0**	27.6	8.5	**1.0**	**15.2**	74.5	0.2
15	Wait	2nd Order Unicycle	–	–	0.0	**41.9**	**38.5**	**0.7**	–	–	0.0	–	–	0.0
16	Lane Switch	2nd Order Unicycle	30.6	**0.3**	**1.0**	**29.8**	**3.1**	**1.0**	29.9	7.0	**1.0**	–	–	0.0
17	Empty	Car with Trailer	14.4	**0.1**	**1.0**	17.9	1.3	**1.0**	14.1	**0.5**	**1.0**	**13.2**	4.3	**1.0**
18	Lane Switch	Car with Trailer	29.4	**0.3**	**1.0**	31.6	**2.1**	**1.0**	**28.7**	26.9	0.9	–	–	0.0
19	Park	Car with Trailer	7.8	0.3	**1.0**	8.2	**0.4**	**1.0**	7.7	0.9	**1.0**	**7.0**	1.4	**1.0**
20	Parking Lot	Car with Trailer	**25.1**	**0.4**	**1.0**	30.0	**2.8**	**1.0**	**25.8**	16.8	**1.0**	–	–	0.0
21	Empty	Planar Rotor	2.8	**0.6**	**1.0**	5.8	11.0	**1.0**	3.5	**4.1**	**1.0**	**2.6**	94.4	0.5
22	Moving Ceiling	Planar Rotor	5.3	**4.7**	**1.0**	7.6	**18.0**	**1.0**	**5.9**	19.7	**1.0**	–	–	0.0
23	Moving Small Ceiling	Planar Rotor	7.1	4.1	**1.0**	9.5	33.9	**1.0**	**6.8**	**28.0**	**1.0**	–	–	0.0
24	Double Ceiling	Planar Rotor	8.3	6.1	0.9	12.2	73.7	0.9	**7.6**	**40.0**	**1.0**	–	–	0.0
25	Elevator	Planar Rotor	15.6	**5.9**	**0.8**	19.7	**80.8**	0.6	**16.9**	91.2	**0.8**	–	–	0.0

we compare time-optimal versus random motion primitives. Our results indicate that time-optimal primitives slightly outperform random primitives. Second, we investigate the motion primitive density and discontinuity bound. From the results, we have chosen a low to medium density of 1000 primitives (3000 for planar rotor) and a discontinuity bound of $\delta = 0.3$. Third, we compared the sampling time bounds, including the batch size to sample and the relative increase

of the upper bound. Our analysis gives a value of $b = 1200$ and $f = 1.7$ as optimal values for this dataset. Finally, we optimized for the time weight in the distance function, finding that a small value of $\lambda = 0.2$ gives the best results.

B. Planner Comparisons

Using the optimized parameter set, we compare all three space-time planners and Naive Db-RRT on the scenario set. The results are depicted in Tab. II.

1) Naive Db-RRT vs ST-Db-RRT: Db-RRT reaches a success rate of $sr = 1.0$ on 19 of the 25 instances and returns a first solution within $t_1 = 0.6\,\mathrm{s}$ on 17 of the 18 unicycle and car instances it solves, up to $54\times$ faster than ST-Db-RRT in the *Bugtrap* environment ($t_1 = 0.6\,\mathrm{s}$ against $t_1 = 32.3\,\mathrm{s}$).

However, in the *Wait* environments, which require the robot to wait for a long time in a confined space, Db-RRT fails while ST-Db-RRT solves them with $sr = 1.0$ for the 1st Order Unicycle and $sr = 0.7$ for the 2nd Order Unicycle. Outside the *Wait* environments the success rates of the two planners are close, and the time dimension is not by itself an advantage: on all five Planar Rotor instances Db-RRT matches or exceeds ST-Db-RRT.

Planning in space-time does not pay off as clearly in the solution cost. ST-Db-RRT is cheaper on only 9 of the 23 jointly solved instances, and all nine are unicycle instances, where the margin is a moderate 3% to 10%. On the Car with Trailer and Planar Rotor systems the ordering inverts: Db-RRT attains the lower cost on every one of these nine instances, by 26% to 107% on the rotor, e.g. in the *Double Ceiling* ($c = 8.3\,\mathrm{s}$ against $c = 12.2\,\mathrm{s}$).

2) Space-Time Planner Comparison: We compare the space-time planners in terms of success rate, time to first solution, and solution cost.

a) Success Rate: ST-Db-RRT achieves 100% on 21 of the 25 instances and is the only planner that solves every instance at least once. Most notably, it is the only planner to find any solution in the 2nd Order Unicycle *Wait* instance ($sr = 0.7$), which forces the system to wait roughly $30\,\mathrm{s}$ in a confined region. Its success rate is surpassed on only three instances, all with a narrow arrival window: the 1st Order Unicycle *Elevator* ($sr = 0.5$ against $sr = 1.0$ for ST-GBRRT), the Planar Rotor *Double Ceiling* ($sr = 0.9$ against $sr = 1.0$ for KIST) and the Planar Rotor *Elevator* ($sr = 0.6$ against $sr = 0.8$), confirming the recurring weakness of ST-Db-RRT in those environments. KIST is similarly reliable ($sr = 1.0$ on 20 of the 25 instances), and its success rate declines only on the temporally hardest problems: it drops to $sr = 0.7$ in the 1st Order Unicycle *Wait*, to $sr = 0.6$ in the 1st Order Unicycle *Elevator*, and fails the 2nd Order Unicycle *Wait*. On the Planar Rotor it is the most reliable planner of the three, reaching $sr = 1.0$ on four of the five instances. ST-GBRRT has the most limited coverage and fails on 11 of the 25 instances, mostly on the long-horizon problems *Bugtrap*, *Wait*, *Lane Switch* and *Parking Lot*, and on the Planar Rotor, where only the *Empty* instance is solved. On the 2nd Order Unicycle it additionally fails in three scenarios outright and drops to $sr = 0.2$ in the *Double Moving Windows*.

b) Time to First Solution: The clearest separation between the planners appears in the time to the first solution. ST-Db-RRT is the fastest of the three planners on 19 of the 25 instances and stays below $t_1 = 4\,\mathrm{s}$ on 17 of the 20 unicycle and car instances, the exceptions being *Bugtrap*, *Wait*, and the 1st Order Unicycle *Elevator*. Its t_1 is thereby largely insensitive to the temporal constraints that slow the optimization-free planners down by an order of magnitude: KIST needs $t_1 = 32.2\,\mathrm{s}$ against $t_1 = 2.0\,\mathrm{s}$ in the 1st Order Unicycle *Wait*, a $16\times$ speed-up for ST-Db-RRT. On the Planar Rotor, however, this ordering no longer holds: KIST reaches a first solution faster than ST-Db-RRT on three of the five instances, clearly so in the *Double Ceiling* ($t_1 = 40.0\,\mathrm{s}$ against $t_1 = 73.7\,\mathrm{s}$). ST-GBRRT is the slowest planner on almost every instance it solves: on the 2nd Order Unicycle it needs up $32\times$ longer than ST-Db-RRT, e.g. $t_1 = 79.9\,\mathrm{s}$ against $t_1 = 2.5\,\mathrm{s}$ in the *Multi Revolving Door*, which also explains its reduced success rates there.

c) Solution Cost: ST-GBRRT converges to the lowest cost on every one of the 14 instances it solves, despite reaching its first solution last. KIST attains the best cost on 7 instances, and is the cheapest planner on four of the five Planar Rotor instances. Its cost inflates, however, in the dynamically crowded 2nd Order Unicycle environments, where it exceeds ST-Db-RRT by roughly 50% in the *Multi Revolving Door* and the *Double Moving Windows*.ST-Db-RRT, however, is no longer uniformly close to the best value: its cost stays within 10% of the best observed value on all but one unicycle instance, but it is 36% above the best cost in the Car with Trailer *Empty* and between 29% and 123% above it on the Planar Rotor.

VIII. Conclusion

We presented DynoFluxBench, a benchmark environment to evaluate kinodynamic space-time motion planners in dynamic environments. To demonstrate its utility and establish strong baselines, we developed three dedicated planners that fuse kinodynamic and space-time methods: We used state-of-the-art kinodynamic planners iDb-RRT [7] and GBRRT [8] and combined them with aspects of the space-time planner ST-RRT* [6]. The results are new planners ST-Db-RRT and ST-GBRRT, respectively. Additionally, we combined different aspects of those planners leading to the KIST planner. For those three planners, we conducted a theoretical analysis of completeness properties, and then used DynoFluxBench to conduct (a) a hyperparameter analysis, (b) a comparison between iDb-RRT and ST-Db-RRT, and (c) a full evaluation of ST-Db-RRT, ST-GBRRT, and KIST on a benchmark set of 25 scenarios involving dynamic obstacles and four different dynamical systems.

Our evaluation showed that the three planners differ in all three metrics. ST-Db-RRT remains resolution-complete and is the only planner to solve every dynamic instance while it reaches the first solution up to $90\times$ faster than the optimization-free planners, demonstrating that deferring

feasibility to a single repair step wins coverage and runtime under a fixed time budget. It does not, however, win on cost: ST-GBRRT converges to the lowest cost on every instance its slower search can cover, while KIST is both the most reliable and the cheapest planner on the Planar Rotor. This picture inverts on the higher-dimensional systems and wherever the feasible arrival window is narrow, as in the *Elevator* and moving-ceiling instances, revealing that optimization-free planning remains valuable when the optimization struggles.

There are multiple immediate extensions of the planners. As outlined, a more informed arrival-time estimate could reach late feasible arrival windows more reliably, kinodynamic rewiring in space-time could recover the asymptotic optimality of ST-RRT*, and variable-time trajectory optimization could further improve solution quality.

REFERENCES

[1] S. LaValle and J. Kuffner, "Randomized kinodynamic planning," in *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 1, 1999, pp. 473–479.

[2] I. A. Sucan and L. E. Kavraki, "Kinodynamic motion planning by interior-exterior cell exploration," in *Workshop on the Algorithmic Foundations of Robotics*, 2009, pp. 449–464.

[3] Y. Li, Z. Littlefield, and K. E. Bekris, *Sparse Methods for Efficient Asymptotically Optimal Kinodynamic Planning*. Cham: Springer International Publishing, 2015, pp. 263–282.

[4] T. Fraichard, "Trajectory planning in a dynamic workspace: a 'state-time space' approach," *Advanced Robotics*, vol. 13, no. 1, pp. 75–94, 1998.

[5] A. Sintov and A. Shapiro, "Time-based rrt algorithm for rendezvous planning of two dynamic systems," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 6745–6750.

[6] F. Grothe, V. N. Hartmann, A. Orthey, and M. Toussaint, "ST-RRT*: Asymptotically-optimal bidirectional motion planning through space-time," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2022, pp. 3314–3320.

[7] J. Ortiz-Haro, W. Hönig, V. N. Hartmann, M. Toussaint, and L. Righetti, "iDb-RRT: Sampling-based kinodynamic motion planning with motion primitives and trajectory optimization," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024, pp. 10 702–10 709.

[8] S. Nayak and M. W. Otte, "Bidirectional sampling-based motion planning without two-point boundary value solution," *IEEE Transactions on Robotics*, vol. 38, no. 6, pp. 3636–3654, 2022.

[9] J. van den Berg and M. Overmars, "Kinodynamic motion planning on roadmaps in dynamic environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2007, pp. 4253–4258.

[10] M. Otte and E. Frazzoli, "RRTX: Asymptotically optimal single-query sampling-based motion planning with quick replanning," *The International Journal of Robotics Research*, vol. 35, no. 7, pp. 797–822, 2016.

[11] J. Bruce and M. Veloso, "Real-time randomized path planning for robot navigation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 3, 2002, pp. 2383–2388.

[12] K. Naderi, J. Rajamäki, and P. Hämäläinen, "RT-RRT*: a real-time path planning algorithm based on rrt*," in *Proceedings of the 8th ACM SIGGRAPH Conference on Motion in Games (MIG)*, 2015, pp. 113–118.

[13] K. Kant and S. W. Zucker, "Toward efficient trajectory planning: The path-velocity decomposition," *The International Journal of Robotics Research*, vol. 5, no. 3, pp. 72–89, 1986.

[14] M. Phillips and M. Likhachev, "SIPP: Safe interval path planning for dynamic environments," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 5628–5635.

[15] M. Hüppi, L. Bartolomei, R. Mascaro, and M. Chli, "T-PRM: Temporal probabilistic roadmap for path planning in dynamic environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022.

[16] N. Kerimov, A. Onegin, and K. Yakovlev, "Safe interval randomized path planning for manipulators," in *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, vol. 35, no. 1, 2025, pp. 213–217.

[17] J. Sim, J. Kim, and C. Nam, "Safe interval rrt* for scalable multi-robot path planning in continuous space," 2025. [Online]. Available: https://arxiv.org/abs/2404.01752

[18] M. Pivtoraiko and A. Kelly, "Kinodynamic motion planning with state lattice motion primitives," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011, pp. 2172–2179.

[19] M. Kleinbort, K. Solovey, Z. Littlefield, K. E. Bekris, and D. Halperin, "Probabilistic completeness of rrt for geometric and kinodynamic planning with forward propagation," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, 2019.

[20] 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.

[21] 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.

[22] D. J. Webb and J. van den Berg, "Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2013, pp. 5054–5061.

[23] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2014, pp. 2997–3004.

[24] M. P. Strub and J. D. Gammell, "Adaptively informed trees (AIT*): Fast asymptotically optimal path planning through adaptive heuristics," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 3191–3198.

[25] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "CHOMP: Gradient optimization techniques for efficient motion planning," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2009, pp. 489–494.

[26] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *The International Journal of Robotics Research*, vol. 33, pp. 1251–1270, 2014.

[27] A. Orthey, B. Frész, and M. Toussaint, "Motion planning explorer: Visualizing local minima using a local-minima tree," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 346–353, apr 2020.

[28] A. Byravan, B. Boots, S. S. Srinivasa, and D. Fox, "Space-time functional gradient optimization for motion planning," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 6499–6506.

[29] W. Hönig, J. Ortiz-Haro, and M. Toussaint, "db-A*: Discontinuity-bounded search for kinodynamic mobile robot motion planning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 13 540–13 547.

[30] J. Ortiz-Haro, W. Hönig, V. N. Hartmann, and M. Toussaint, "iDb-A*: Iterative search and optimization for optimal kinodynamic motion planning," *IEEE Transactions on Robotics*, vol. 41, pp. 2031–2049, 2025.

[31] D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock, "Randomized kinodynamic motion planning with moving obstacles," *The International Journal of Robotics Research*, vol. 21, no. 3, pp. 233–255, Mar. 2002.

[32] C. Chen, M. Rickert, and A. Knoll, "Kinodynamic motion planning with space-time exploration guided heuristic search for car-like robots in dynamic environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 2666–2671.

[33] Z. A. Ali and K. Yakovlev, "Safe interval path planning with kinodynamic constraints," in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, vol. 37, no. 10, 2023, pp. 12 330–12 337.

[34] S. Huang, Y. Wu, Y. Tao, and V. Kumar, "Safe interval motion planning for quadrotors in dynamic environments," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2025, pp. 2780–2786.