BOWConnect: Parallel Bayesian Optimization over Windows with Learned Local Cost Maps for Sample-Efficient Kinodynamic Motion Planning

***Abstract*—This paper presents BOWConnect, a bidirectional parallel kinodynamic motion planner that addresses three fundamental limitations of existing sampling-based methods: sample inefficiency in high-dimensional state spaces, unreliable cost heuristics under dynamic constraints, and poor performance in narrow passage environments. Unlike classical planners that rely on random control sampling and geometric distance heuristics, BOWConnect integrates Bayesian Optimization over Windows (BOW) as a learning-based steering function within a parallel tree-based exploration framework, enabling each worker to learn local cost maps and constraints to guide sampling toward dynamically feasible and collision-free controls. A bidirectional architecture simultaneously grows forward and backward trees from the start and goal regions in parallel threads, with a spatial hashing mechanism enabling fast connection queries and a boundary value problem solver generating kinodynamically consistent bridge trajectories.Extensive evaluations across ten benchmark environments demonstrate that BOWConnect achieves 100% success in complex scenarios, including narrow passages and non-convex spaces where state-of-the-art planners fail or degrade substantially, with real-world deployment on a ground vehicle and a quadrotor confirming real-time planning with no collisions. Videos of real-world and simulated experiments are available at https://bow-connect.github.io/, with the open-source code to be released upon publication.**

I. Introduction

Kinodynamic motion planning remains a fundamental challenge in robotics because the relevant search space is the system *state space*, which is typically higher-dimensional than configuration space due to the inclusion of configuration variables and their derivatives (e.g., velocities and higher-order derivatives) [1, 2]. This dimensionality makes sampling-based exploration susceptible to the curse of dimensionality and slow convergence [3, 4].

Beyond dimensionality, kinodynamic planning must satisfy differential constraints throughout the trajectory [1, 2], meaning geometrically collision-free paths may still be dynamically infeasible. For many systems, an exact steering function is difficult or impractical to derive [5], so planners typically generate motion by sampling controls and forward-propagating the dynamics rather than by simple interpolation [2, 6, 7]. Guiding this search is equally difficult: existing methods often rely on Euclidean distance heuristics for candidate selection because estimating transition costs under dynamics is nontrivial, though learning-based approaches have recently been proposed to address this limitation [8, 9]. These challenges compound further in highly constrained environments such as narrow passages, where generating dynamically feasible connections through constrained regions is particularly demanding [6, 10].

Parallelism has improved performance for geometric planners such as RRT-Connect through GPU-parallel expansion and collision checking [11], but kinodynamic planners have historically been harder to parallelize due to propagation-dominated, serial computation patterns. This has motivated recent work that explicitly restructures kinodynamic tree growth for massively parallel devices [12], yet scalable bidirectional kinodynamic planning with efficient connection discovery remains an open problem.

In this work, we present **BOWConnect**, a bidirectional and parallel kinodynamic motion planner designed to improve sample efficiency and accelerate connection discovery in continuous state spaces. BOWConnect grows two trees simultaneously from the start and goal, using an online learning-based trajectory generator to produce collision-free, kinodynamically feasible motions directly in continuous state and control spaces. A spatial hashing mechanism embedded in the `MotionTree` data structure enables efficient multi-stage feasibility verification, reducing redundant propagation and accelerating connection queries between the two trees. Together, these components improve computational efficiency while maintaining trajectory feasibility across diverse robotic platforms, as demonstrated through experiments on ground and aerial vehicles.

The main contributions of this work are:

- An online learning-based method for generating collision-free, kinodynamically feasible trajectories directly in continuous state and control spaces.
- A bidirectional parallel planning architecture that enhances global connectivity while preserving local trajectory quality.
- A spatial hashing mechanism within the `MotionTree` data structure that enables efficient multi-stage feasibility verification.
- Extensive experimental validation on ground vehicles with unicycle and bicycle models, and aerial vehicles with quadrotor dynamics.

II. Related Work

Sampling-based and optimization-based planners have emerged as dominant paradigms for kinodynamic motion planning in complex environments [13]. Sampling-based methods construct feasible trajectories through stochastic exploration of continuous state and control spaces while explicitly respecting dynamic constraints [14, 1, 5]. These algorithms provide probabilistic completeness and strong global exploration capabilities; however, their solutions are often suboptimal and may require post-processing. Optimization-based approaches such as CHOMP, STOMP, and TrajOpt formulate motion planning as nonlinear trajectory optimization [15, 16, 17]. Although they generate smooth, high-quality trajectories, their performance depends heavily on gradient information and good initialization, making them susceptible to local minima in cluttered environments.

To improve exploration efficiency, sampling-based planners have incorporated structural and heuristic refinements. Early methods such as RRT and RRT* rely on uniform random sampling and frequently evaluate edges that do not contribute to the final solution [1, 13]. Subsequent approaches introduced bidirectional search [18], informed sampling [19], and cost-aware edge prioritization [20, 21] to concentrate computation in promising regions. Batch-informed algorithms such as BIT* integrate graph-search principles with sampling-based planning to reduce redundant computation [21]. In kinodynamic settings specifically, EST performs forward simulation of randomly sampled controls while biasing expansion toward sparsely explored regions [2], KPIECE leverages low-dimensional projections to guide exploration [6], and SST introduces pruning mechanisms to maintain a sparse set of representative nodes while achieving asymptotic near-optimality [5].

Several approaches adopt a "plan first, optimize later" paradigm, refining RRT-generated paths using time-optimal steering or bang–bang control primitives, motivated by empirical evidence that post-optimization can yield high-quality solutions faster than asymptotically optimal variants such as RRT* or SST* [13, 5, 22].

Despite these advances, most kinodynamic sampling-based planners still rely heavily on repeated forward simulation and collision checking, which remain primary computational bottlenecks. Moreover, their discrete graph constructions typically do not explicitly reason about uncertainty or efficiently generate multiple locally optimal motion hypotheses under strict real-time constraints.

Receding-horizon control (RHC) offers an alternative perspective by repeatedly solving short-horizon optimization problems and executing only the first control action before replanning [23]. This paradigm has been successfully applied to unified motion planning and control in dynamic environments [24]. Reactive planners such as the Dynamic Window Approach and Hybrid Reciprocal Velocity Obstacles operate over short temporal windows directly in velocity space [25, 26]. However, these approaches generally sacrifice long-horizon planning capability for reactivity.

Bayesian optimization (BO), combined with Gaussian Processes (GPs), provides a principled framework for data-efficient optimization under uncertainty and has recently been explored for motion planning [27]. GP-based planners model trajectory costs probabilistically and use acquisition functions to guide exploration toward safe and promising regions of the search space [28]. Extensions such as GPMP and GPMP2 leverage sparse GP representations to enable efficient continuous-time trajectory optimization under kinodynamic constraints [29]. Nevertheless, many BO-based planners remain computationally expensive and are typically applied in unidirectional, locally greedy manners.

Bayesian Optimization over Windows (BOW) addresses this limitation by applying BO over short receding-horizon control windows in continuous state and control spaces. By leveraging GP surrogates to guide control sampling, BOW enables locally optimal and dynamically feasible trajectory generation while explicitly accounting for uncertainty [30]. However, unidirectional window-based BO planners may struggle with global exploration, particularly in environments with narrow passages or complex connectivity structures.

III. Problem Formulation

Let $\mathcal{C}$ be the robot configuration space, modeled as an n-dimensional smooth manifold. Let $\mathcal{C}_{\text{free}} \subset \mathcal{C}$ denote the open subset of configurations in which the robot does not intersect any obstacle. The kinodynamic planning problem extends beyond geometric path planning by incorporating the robot's dynamics. Let $\mathbf{x} = (q, \dot{q}) \in \mathcal{X}$ be the $2n$-dimensional state vector, where $\mathcal{X} = T(\mathcal{C})$ is the tangent bundle of $\mathcal{C}$. The feasible state space is defined as:

$$\mathcal{X}_{\text{free}} = \{(q, \dot{q}) \in \mathcal{X} \mid q \in \mathcal{C}_{\text{free}}\} \tag{1}$$

The system dynamics are governed by the ordinary differential equation $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$, where $\mathbf{u} \in \mathcal{U} \subset \mathbb{R}^m$ is a control input drawn from a compact action set. For a candidate control $\mathbf{u}$, the system dynamics are integrated using fourth-order Runge-Kutta over a finite time horizon T to obtain the predicted trajectory:

$$\tau(\mathbf{u}) = \Phi(\mathbf{x}, \mathbf{u}) = \{\mathbf{x}(t) : t \in [0, T]\} \tag{2}$$

where $\mathbf{p}(t)$ denotes the position component of $\mathbf{x}(t)$ at time t.

Problem (Kinodynamic Planning)**.** *Given initial and goal states $\mathbf{x}_I, \mathbf{x}_G \in \mathcal{X}_{free}$, find a control sequence $\mathbf{u} : [0, t_F] \to \mathcal{U}$ such that the resulting trajectory $\tau(\mathbf{u}) = \Phi(\mathbf{x}_I, \mathbf{u})$ satisfies:*

$$\tau(\mathbf{u}) \subset \mathcal{X}_{free}, \quad \mathbf{x}(t_F) = \mathbf{x}_G \tag{3}$$

When time-optimality is required, t_F is minimized over all feasible solutions.

A key operation in bidirectional tree-based planners is connecting states from opposing trees, which requires solving a Boundary Value Problem (BVP) [31]. Given two states $\mathbf{x}_f$ and $\mathbf{x}_b$ belonging to the forward tree $\mathcal{T}_I$ and backward tree $\mathcal{T}_G$ respectively, the BVP seeks a control $\mathbf{u}_c : [0, t_c] \to \mathcal{U}$ such that the connecting trajectory $\tau(\mathbf{u}_c) = \Phi(\mathbf{x}_f, \mathbf{u}_c)$ satisfies:

$$\mathbf{x}(t_c) = \mathbf{x}_b, \quad \tau(\mathbf{u}_c) \subset \mathcal{X}_{\text{free}} \tag{4}$$

The kinematic feasibility of this connection is checked by verifying that the required heading $\theta_{\text{req}} = \arctan 2(y_b - y_f, x_b - x_f)$ and travel distance satisfy:

$$|\theta_{\text{req}} - \theta_f| < \theta_{\max} \quad \text{and} \quad \frac{\|\mathbf{p}_b - \mathbf{p}_f\|}{v_{\max}} < T_{\max} \tag{5}$$

Unlike geometric planners, where straight-line interpolation often suffices, the BVP solution in kinodynamic planning must satisfy $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$ at every point along the connecting trajectory, requiring the robot to arrive at $\mathbf{x}_b$ with the correct velocity and dynamic state.

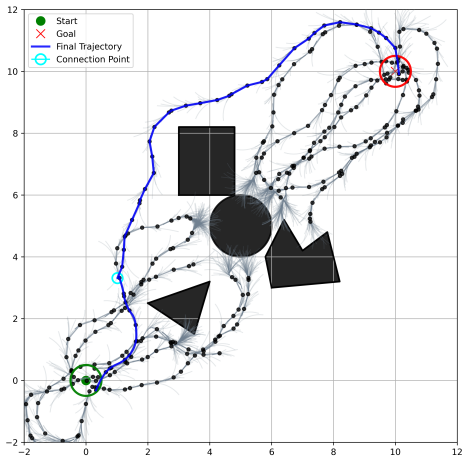


Fig. 1: Parallel motion tree growth of **BOW-Connect** in a cluttered planar environment. Trees are initialized from the start (green) and goal (red) regions and extended in separate threads using the BOW planner [30]. Grey trajectories depict sampled short-horizon trajectories and the bold blue curve indicates the final connected path. A successful bidirectional connection via boundary value problem solution is highlighted in cyan.

IV. BOWConnect Algorithm

BOWConnect leverages multiple independent BOW planner [30] instances within a tree-based exploration framework. Unlike traditional bidirectional planners that grow a single forward and backward tree, BOWConnect spawns $N/2$ forward workers and $N/2$ backward workers, each maintaining its own search tree and BOW instance. This high-level parallelism enables diverse exploration patterns while preserving the kinodynamic feasibility guarantees of the underlying BOW planner.

Given an initial state $\mathbf{x}_{\text{start}} \in \mathbb{R}^n$ and goal position $\mathbf{p}_{\text{goal}} \in \mathbb{R}^d$ where d could be 2 or 3, BOWConnect first samples multiple initial configurations around both regions. Specifically, it generates collision-free start states $\mathcal{S} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ and goal states $\mathcal{G} = \{\mathbf{x}'_1, \ldots, \mathbf{x}'_N\}$ by sampling uniformly within radius r_{goal} of the respective centers. Each sampled state includes a random heading $\theta \in [-\pi, \pi]$, enabling workers to explore from different initial orientations. This sampling strategy addresses heading ambiguity at the goal and provides diversified exploration directions, significantly improving connection probability between trees.

A. Tree Growth with Constrained Bayesian Optimization

Figure 1 illustrates the tree growth behavior of BOW-Connect in a planar environment. Each worker thread maintains a `MotionTree` data structure that stores explored states and their parent relationships. The tree growth process follows the RRT paradigm but uses BOW as the steering function instead of simple geometric interpolation. At each iteration, a worker samples a random state $\mathbf{x}_{\text{rand}}$ in the configuration space, finds the nearest state $\mathbf{x}_{\text{near}}$ in its current tree, and invokes BOW to generate a kinodynamically feasible trajectory connecting them.

The BOW planner formulates local planning as a constrained Bayesian Optimization problem over the control space $\mathcal{U} \subset \mathbb{R}^m$, where the dimension m depends on the robot type. For each candidate control $\mathbf{u}$, BOW uses a robot-specific motion model to simulate forward trajectory through numerical integration. The motion model captures the system dynamics via ordinary differential equations $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$, which are integrated using fourth-order Runge-Kutta to obtain the predicted trajectory $\tau(\mathbf{u})$ over a finite time horizon T. For each simulated trajectory, BOW evaluates two functions: a reward and a constraint satisfaction indicator. The reward is defined as the negative distance to the local target for collision-free trajectories:

$$r(\mathbf{u}) = \begin{cases} -\|\mathbf{p}(T) - \mathbf{p}_{\text{target}}\| & \text{if trajectory is collision-free} \\ -\infty & \text{otherwise} \end{cases} \tag{6}$$

where $\mathbf{p}(T)$ represents the position component of the predicted state after time horizon T. The constraint function returns 1 if the entire trajectory avoids obstacles and 0 otherwise:

$$c(\mathbf{u}) = 1 - \mathbb{I}_{\text{collision}}(\tau(\mathbf{u})) \tag{7}$$

BOW learns two separate Gaussian Process models: one for the reward $\mathcal{GP}_r$ and one for the constraint $\mathcal{GP}_c$. These models provide mean predictions $\mu_r(\mathbf{u})$, $\mu_c(\mathbf{u})$ and uncertainties $\sigma_r^2(\mathbf{u})$, $\sigma_c^2(\mathbf{u})$ respectively. **The acquisition function balances exploration and exploitation while respecting collision constraints through the probability of feasibility**:

$$P_{\text{feas}}(\mathbf{u}) = \Phi\left(\frac{\mu_c(\mathbf{u})}{\sigma_c(\mathbf{u})}\right) \tag{8}$$

where Φ is the standard normal cumulative distribution function. The next control to evaluate is selected by optimizing the constrained acquisition function $\alpha(\mathbf{u}) = [\mu_r(\mathbf{u}) - \kappa\sigma_r(\mathbf{u})] \times P_{\text{feas}}(\mathbf{u})$. **This learned cost map over the control space guides the search toward promising collision-free controls.**

When BOW successfully generates a collision-free trajectory $\tau = [\mathbf{x}_{\text{near}}, \mathbf{x}_1, \ldots, \mathbf{x}_m]$, the worker adds these states sequentially to its tree. The kinodynamic constraints are inherently satisfied because BOW respects the robot-specific velocity limits, acceleration bounds, and curvature constraints through its motion model integration. This ensures that every edge in the tree represents a dynamically feasible motion primitive for the given robot platform.

B. Connection Detection and BVP Solving

While workers grow their trees independently, the main thread periodically checks for potential connections between forward and backward trees. To enable efficient connection queries, BOWConnect employs spatial hashing through the `MotionTree` structure. Each state $\mathbf{x}$ is mapped to a discrete grid cell via the hash function:

$$h(\mathbf{x}) = \lfloor (x - x_0)/\Delta \rfloor \oplus \lfloor (y - y_0)/\Delta \rfloor \oplus \lfloor (z - z_0)/\Delta \rfloor \tag{9}$$

where Δ is the grid resolution and $\oplus$ denotes bit-packing operations. This allows $\mathcal{O}(1)$ average-case lookup to determine if a forward tree state has nearby backward tree states.

When a potential connection is detected, BOWConnect performs a multi-stage verification. First, it checks kinematic

feasibility by ensuring the required heading change and distance are within vehicle constraints. Specifically, for states $\mathbf{x}_f$ and $\mathbf{x}_b$, the connection is kinematically feasible if:

$$|\theta_{\text{req}} - \theta_f| < \theta_{\max} \quad \text{and} \quad \frac{\|\mathbf{p}_b - \mathbf{p}_f\|}{v_{\max}} < T_{\max} \tag{10}$$

where $\theta_{\text{req}} = \arctan 2(y_b - y_f, x_b - x_f)$ is the required heading. If feasible, BOWConnect solves a boundary value problem to generate the connecting trajectory.

The BVP solver employs proportional control to smoothly transition between states while respecting kinodynamic constraints. Starting from $\mathbf{x}_f$, it iteratively computes heading error $\Delta\theta = \theta_{\text{desired}} - \theta_{\text{current}}$ and applies yaw rate $\omega = \text{clamp}(k_p \Delta\theta / \Delta t, -\omega_{\max}, \omega_{\max})$ while moving toward $\mathbf{x}_b$ with speed proportional to remaining distance. The resulting trajectory undergoes collision checking before acceptance. If multiple connections are found, BOWConnect selects the one with minimum Euclidean distance, as shorter connections are more reliable and easier to verify.

C. Solution Extraction and Fallback Mechanisms

BOWConnect terminates when trees successfully connect or when any worker reaches the goal region. Upon connection, the algorithm extracts trajectories from both trees up to the connection points, generates the BVP bridge trajectory, and concatenates them. If the backward trajectory does not fully reach the goal, an additional BVP call extends it from the last state to the goal configuration.

The algorithm implements graceful degradation through fallback solutions. If no connection is found within the time budget but a forward worker reached the goal, that unidirectional solution is returned. Similarly, backward solutions serve as fallbacks. **The parallel architecture provides theoretical speedup proportional to the number of workers, though practical speedup is limited by connection checking overhead and sequential BVP solving**. With N_f forward and N_b backward workers, the probability of finding at least one successful path is:

$$P_{\text{success}} = 1 - (1 - p_{\text{single}})^{N_f \times N_b} \tag{11}$$

where p_{single} is the success probability of a single worker pair. This portfolio effect, combined with BOW's efficient local planning, enables BOWConnect to solve complex kinodynamic planning problems in real-time.

V. Experiments and Results

All experiments were performed on a desktop computer running Ubuntu 22.04 LTS with an AMD Ryzen™ 9 7950X processor, an NVIDIA GTX 4070 GPU, and 32 GB of RAM. Robot localization was achieved using a VICON motion capture system sampling at 120 Hz. The BOWConnect Planner was parameterized with the following constraints: maximum and minimum speeds of 1.0 m/s and 0.0 m/s respectively, maximum acceleration of 0.5 m/s^2, maximum yaw rate of 0.6981 rad/s, and maximum yaw acceleration of 2.0472 rad/s^2. The planner employed a time discretization of 0.1 s and a prediction horizon of 5.0 s.

A. UGV Benchmark Results

We evaluate *BOWConnect* against state-of-the-art kinodynamic planners including RRT [1], SST [5], EST [2], KPIECE [6], and BOW [30] across six UGV environments: *Bugtrap*, *Narrow Passage-1*, *Narrow Passage-2*, *Forest*, *Nonconvex*, and *Intel*. For each environment–planner pair, five independent trials were conducted, and the mean and standard deviation of all key metrics (except success rate) were computed. All planners were evaluated under both *Unicycle* and *Bicycle* motion models. The evaluation metrics include total planning time, trajectory length, success rate, average velocity, and average jerk (Table I).

Across all environments, *BOWConnect* achieves a 100% success rate under both motion models, including geometrically constrained scenarios such as *Forest*, *Nonconvex*, and *Intel*. In contrast, all baseline planners exhibit reduced robustness under the Bicycle model across every environment, with success rates dropping substantially relative to the Unicycle model within the 30-second time budget.

In *Narrow Passage-2*, the Bicycle variants of RRT and SST achieve 40% and 20% success rates, respectively, whereas *BOWConnect* maintains 100% success with an average planning time of 0.048 s. This demonstrates improved feasibility handling under strict nonholonomic constraints. Similar degradation patterns are observed in *Forest*, *Nonconvex*, and *Intel*, demonstrating that the robustness gap is systematic rather than environment-specific.

BOWConnect consistently achieves the lowest or near-lowest computation time across all environments. In *Bugtrap*, it computes solutions in 0.035 s (Unicycle) and 0.021 s (Bicycle), outperforming all baselines except BOW under the Bicycle model. In both *Narrow Passage-1* and *Narrow Passage-2*, where several classical planners approach the time limit or fail under the Bicycle model, *BOWConnect* solves the tasks in under 0.06 s on average and achieves the lowest computation time. Similarly, in *Forest*, *Nonconvex*, and *Intel*, *BOWConnect* yields the smallest planning time under both motion models.

In addition to robustness and efficiency, *BOWConnect* produces high-quality trajectories. In *Nonconvex* (Unicycle), it achieves the shortest average trajectory length (15.814 m) among all planners, while some baselines such as KPIECE fail to find feasible solutions. Although the Bicycle model results in longer trajectories due to curvature constraints, feasible solutions are consistently obtained within the time budget.

In terms of average velocity, *BOWConnect* remains competitive across all environments and achieves the highest value in *Nonconvex* for both of the motion model.

BOW-based planners further exhibit significantly lower jerk values than classical sampling-based methods, indicating smoother control profiles. For example, in *Bugtrap* (Bicycle), *BOWConnect* achieves an average jerk of 0.0279, substantially lower than RRT and EST. This reduction in jerk reflects the benefit of local Bayesian optimization within the window-based refinement framework.

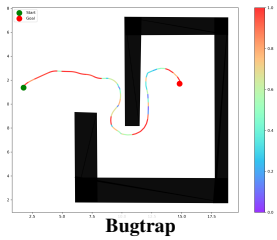
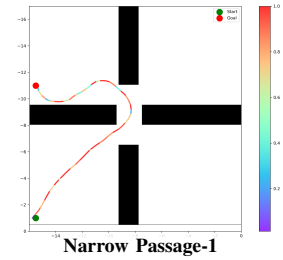
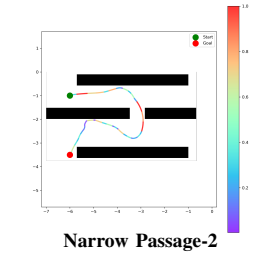
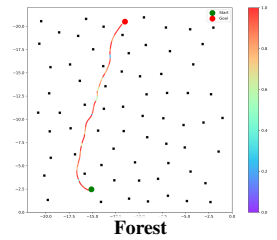
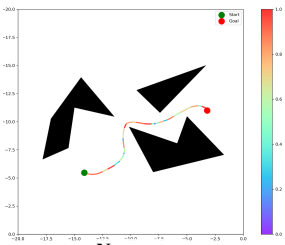
Environment	Planner	Model	Total Time (s)	Traj. Length	Succ. (%)	Avg. Velocity	Avg. Jerk
 Bugtrap	RRT[1]	Unicycle	0.815 ± 0.734	22.648 ± 3.679	**100.0**	0.781 ± 0.005	0.8871 ± 0.0698
		Bicycle	—	—	0.0	—	—
	SST[5]	Unicycle	5.066 ± 0.007	24.601 ± 3.493	**100.0**	0.794 ± 0.024	0.8406 ± 0.0494
		Bicycle	—	—	0.0	—	—
	EST[2]	Unicycle	0.849 ± 0.569	**20.937 ± 1.869**	**100.0**	0.808 ± 0.037	0.8982 ± 0.0806
		Bicycle	—	—	0.0	—	—
	KPIECE[6]	Unicycle	2.220 ± 2.591	22.752 ± 4.194	60.0	**0.851 ± 0.016**	0.7024 ± 0.0464
		Bicycle	—	—	0.0	—	—
	BOW[30]	Unicycle	0.200 ± 0.156	45.324 ± 35.758	**100.0**	0.735 ± 0.052	**0.1373 ± 0.0102**
		Bicycle	**0.019 ± 0.006**	46.536 ± 10.557	**100.0**	**1.060 ± 0.033**	**0.0065 ± 0.0010**
	BOWConnect	Unicycle	**0.035 ± 0.014**	22.091 ± 3.427	**100.0**	0.800 ± 0.045	0.1881 ± 0.0289
		Bicycle	0.021 ± 0.005	**23.838 ± 2.397**	**100.0**	1.000 ± 0.006	0.0279 ± 0.0047
 Narrow Passage-1	RRT[1]	Unicycle	0.903 ± 0.606	21.714 ± 1.363	**100.0**	0.732 ± 0.05	0.8528 ± 0.0320
		Bicycle	25.989 ± 4.714	74.487 ± 6.992	80.0	0.765 ± 0.08	0.7982 ± 0.0474
	SST[5]	Unicycle	30.214 ± 0.009	20.789 ± 1.135	**100.0**	0.794 ± 0.007	0.8618 ± 0.0414
		Bicycle	30.091 ± 0.009	74.640 ± 8.995	80.0	0.784 ± 0.032	0.7549 ± 0.0588
	EST[2]	Unicycle	6.112 ± 7.231	**20.090 ± 0.537**	**100.0**	0.787 ± 0.029	0.8813 ± 0.0520
		Bicycle	—	—	0.0	—	—
	KPIECE[6]	Unicycle	18.932 ± 11.173	20.853 ± 0.889	**100.0**	**0.813 ± 0.030**	0.7052 ± 0.0201
		Bicycle	—	—	0.0	—	—
	BOW[30]	Unicycle	19.538 ± 13.017	53.613 ± 20.695	**100.0**	0.610 ± 0.041	**0.1677 ± 0.0047**
		Bicycle	—	—	0.0	—	—
	BOWConnect	Unicycle	**0.048 ± 0.003**	24.801 ± 6.550	**100.0**	0.783 ± 0.077	0.3740 ± 0.1797
		Bicycle	**0.034 ± 0.008**	**22.190 ± 2.059**	**100.0**	**0.992 ± 0.040**	**0.0381 ± 0.0064**
 Narrow Passage-2	RRT[1]	Unicycle	0.357 ± 0.307	11.374 ± 3.418	**100.0**	**0.710 ± 0.083**	0.7586 ± 0.0364
		Bicycle	26.528 ± 6.883	**8.229 ± 10.622**	40.0	0.477 ± 0.244	0.5550 ± 0.2486
	SST[5]	Unicycle	30.062 ± 0.009	**6.596 ± 0.165**	**100.0**	0.701 ± 0.028	0.7912 ± 0.1364
		Bicycle	30.003 ± 0.001	4.252 ± 8.476	20.0	0.459 ± 0.234	0.1403 ± 0.3137
	EST[2]	Unicycle	0.384 ± 0.206	8.434 ± 1.392	**100.0**	0.676 ± 0.056	0.8347 ± 0.0642
		Bicycle	—	—	0.0	—	—
	KPIECE[6]	Unicycle	1.432 ± 0.999	8.522 ± 1.026	**100.0**	0.700 ± 0.068	0.6730 ± 0.0969
		Bicycle	—	—	0.0	—	—
	BOW[30]	Unicycle	6.971 ± 11.596	31.626 ± 21.971	**100.0**	0.431 ± 0.036	**0.1504 ± 0.0038**
		Bicycle	—	—	0.0	—	—
	BOWConnect	Unicycle	**0.056 ± 0.021**	8.401 ± 0.737	**100.0**	0.522 ± 0.032	0.2232 ± 0.0203
		Bicycle	**0.048 ± 0.011**	8.603 ± 0.034	**100.0**	**0.641 ± 0.031**	**0.0493 ± 0.0041**
 Forest	RRT[1]	Unicycle	1.020 ± 1.104	21.954 ± 1.646	**100.0**	0.782 ± 0.035	0.8671 ± 0.0249
		Bicycle	1.148 ± 0.582	62.091 ± 14.824	**100.0**	0.706 ± 0.016	0.8065 ± 0.0440
	SST[5]	Unicycle	5.052 ± 0.005	26.491 ± 7.449	**100.0**	0.801 ± 0.012	0.8692 ± 0.0545
		Bicycle	5.028 ± 0.002	63.091 ± 3.387	60.0	0.791 ± 0.018	0.6921 ± 0.0110
	EST[2]	Unicycle	2.072 ± 2.196	23.727 ± 5.727	80.0	0.807 ± 0.032	0.8942 ± 0.0426
		Bicycle	—	—	0.0	—	—
	KPIECE[6]	Unicycle	4.309 ± 1.621	30.613 ± 10.465	20.0	**0.817 ± 0.041**	0.7486 ± 0.0601
		Bicycle	—	—	0.0	—	—
	BOW[30]	Unicycle	0.045 ± 0.029	24.523 ± 1.402	**100.0**	0.790 ± 0.054	**0.1671 ± 0.0126**
		Bicycle	0.030 ± 0.005	29.437 ± 6.909	**100.0**	**0.971 ± 0.021**	**0.0101 ± 0.0013**
	BOWConnect	Unicycle	**0.024 ± 0.001**	**20.176 ± 1.496**	**100.0**	0.811 ± 0.036	0.2423 ± 0.0434
		Bicycle	**0.025 ± 0.001**	**21.464 ± 2.795**	**100.0**	0.871 ± 0.052	0.0193 ± 0.0075
 Nonconvex	RRT[1]	Unicycle	0.612 ± 0.742	26.355 ± 11.862	**100.0**	0.754 ± 0.033	0.8427 ± 0.0416
		Bicycle	3.976 ± 1.480	31.496 ± 12.258	40.0	0.667 ± 0.055	0.7312 ± 0.1038
	SST[5]	Unicycle	5.068 ± 0.006	21.715 ± 12.541	80.0	0.764 ± 0.013	0.7782 ± 0.0797
		Bicycle	5.021 ± 0.005	35.528 ± 16.913	20.0	0.732 ± 0.057	0.6652 ± 0.1004
	EST[2]	Unicycle	1.369 ± 2.241	19.414 ± 8.204	80.0	0.795 ± 0.026	0.8891 ± 0.0434
		Bicycle	—	—	0.0	—	—
	KPIECE[6]	Unicycle	—	—	0.0	—	—
		Bicycle	—	—	0.0	—	—
	BOW[30]	Unicycle	0.034 ± 0.015	16.770 ± 1.044	**100.0**	0.770 ± 0.079	**0.1400 ± 0.0070**
		Bicycle	0.022 ± 0.012	28.626 ± 18.668	80.0	0.849 ± 0.476	**0.0049 ± 0.0028**
	BOWConnect	Unicycle	**0.024 ± 0.007**	**15.814 ± 0.824**	**100.0**	**0.859 ± 0.060**	0.2187 ± 0.0397
		Bicycle	**0.019 ± 0.004**	**17.816 ± 3.969**	**100.0**	**0.973 ± 0.039**	0.0301 ± 0.0066
 Intel	RRT[1]	Unicycle	6.886 ± 5.033	77.852 ± 4.577	**100.0**	0.777 ± 0.016	0.8563 ± 0.0153
		Bicycle	15.945 ± 3.522	164.459 ± 4.196	**100.0**	0.769 ± 0.013	0.9034 ± 0.0238
	SST[5]	Unicycle	30.184 ± 0.016	75.559 ± 7.137	60.0	0.781 ± 0.011	0.8389 ± 0.0303
		Bicycle	30.088 ± 0.004	**152.552 ± 11.493**	60.0	0.784 ± 0.005	0.8790 ± 0.0218
	EST[2]	Unicycle	23.704 ± 12.025	**70.698 ± 4.662**	40.0	**0.808 ± 0.011**	0.8625 ± 0.0471
		Bicycle	—	—	0.0	—	—
	KPIECE[6]	Unicycle	—	—	0.0	—	—
		Bicycle	—	—	0.0	—	—
	BOW[30]	Unicycle	12.893 ± 14.040	180.675 ± 119.789	80.0	0.430 ± 0.242	**0.1369 ± 0.0767**
		Bicycle	—	—	0.0	—	—
	BOWConnect	Unicycle	**0.397 ± 0.039**	83.803 ± 14.019	**100.0**	0.566 ± 0.009	0.1904 ± 0.0137
		Bicycle	**2.133 ± 1.707**	917.917 ± 773.043	**100.0**	**1.089 ± 0.008**	**0.0118 ± 0.0023**

TABLE I: Comparison of state-of-the-art kinodynamic motion planning methods across six environments for a UGV platform. The best results in each environment are highlighted in **bold**. A dash (—) denotes that the corresponding planner failed to find a feasible solution. Each planner was evaluated under two motion models: Unicycle and Bicycle.

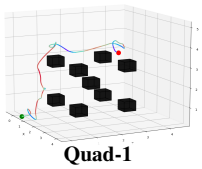
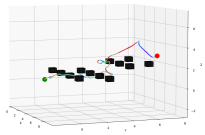
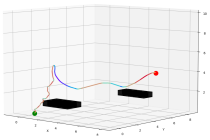
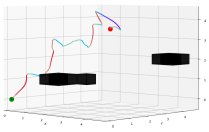
Environment	Planner	Total Time (s)	Traj. Length	Succ. (%)	Avg. Velocity	Avg. Jerk
 **Quad-1**	RRT[1]	7.815 ± 5.281	7.558 ± 1.099	**100.0**	0.443 ± 0.088	0.0364 ± 0.0041
	SST[5]	30.197 ± 0.009	7.822 ± 1.126	**100.0**	0.479 ± 0.159	0.0374 ± 0.0063
	EST[2]	42.196 ± 1.998	8.027 ± 3.189	**100.0**	0.578 ± 0.040	0.0293 ± 0.0047
	KPIECE[6]	29.366 ± 2.941	25.404 ± 27.838	**100.0**	**1.769 ± 1.879**	0.0460 ± 0.0074
	BOW[30]	**0.094 ± 0.027**	6.673 ± 0.845	**100.0**	0.575 ± 0.132	0.0317 ± 0.0038
	BOWConnect	0.117 ± 0.092	**6.562 ± 0.790**	**100.0**	0.568 ± 0.068	**0.0279 ± 0.0039**
 **Quad-2**	RRT[1]	4.558 ± 3.380	16.835 ± 2.252	**100.0**	0.792 ± 0.170	**0.0341 ± 0.0030**
	SST[5]	30.200 ± 0.006	**16.715 ± 2.075**	**100.0**	0.886 ± 0.451	0.0353 ± 0.0028
	EST[2]	41.701 ± 0.989	25.492 ± 7.931	**100.0**	**2.686 ± 2.178**	0.0533 ± 0.0221
	KPIECE[6]	32.116 ± 0.093	30.546 ± 16.021	**100.0**	2.155 ± 1.659	0.0519 ± 0.0179
	BOW[30]	0.103 ± 0.040	33.944 ± 34.635	**100.0**	0.684 ± 0.066	0.0470 ± 0.0041
	BOWConnect	**0.086 ± 0.069**	27.145 ± 26.096	**100.0**	0.680 ± 0.072	0.0432 ± 0.0053
 **Quad-3**	RRT[1]	3.885 ± 2.856	18.224 ± 3.044	**100.0**	0.611 ± 0.230	0.0346 ± 0.0012
	SST[5]	30.208 ± 0.006	**16.615 ± 0.492**	**100.0**	0.775 ± 0.127	**0.0334 ± 0.0031**
	EST[2]	40.987 ± 0.695	16.943 ± 4.446	**100.0**	1.301 ± 0.662	0.0364 ± 0.0210
	KPIECE[6]	32.116 ± 0.104	26.399 ± 16.319	**100.0**	**2.269 ± 2.273**	0.0445 ± 0.0110
	BOW[30]	**0.074 ± 0.009**	22.674 ± 10.714	**100.0**	0.657 ± 0.039	0.0504 ± 0.0019
	BOWConnect	0.081 ± 0.029	17.079 ± 2.566	**100.0**	0.705 ± 0.057	0.0467 ± 0.0062
 **Quad-4**	RRT[1]	3.802 ± 2.566	9.866 ± 1.714	**100.0**	0.531 ± 0.129	0.0372 ± 0.0047
	SST[5]	30.198 ± 0.007	9.833 ± 3.309	**100.0**	0.644 ± 0.212	0.0372 ± 0.0056
	EST[2]	40.895 ± 1.017	11.502 ± 9.871	**100.0**	**1.201 ± 0.668**	0.0320 ± 0.0145
	KPIECE[6]	31.475 ± 1.591	11.528 ± 3.423	**100.0**	0.819 ± 0.294	0.0426 ± 0.0032
	BOW[30]	**0.058 ± 0.014**	6.463 ± 1.595	**100.0**	0.637 ± 0.107	0.0278 ± 0.0049
	BOWConnect	0.091 ± 0.078	**6.231 ± 0.572**	**100.0**	0.597 ± 0.043	**0.0246 ± 0.0048**

TABLE II: Comparison of state-of-the-art kinodynamic motion planning methods across four quadrotor environments (Quad-1, Quad-2, Quad-3, and Quad-4). The best results in each environment are highlighted in **bold**. All planners were evaluated using a quadrotor dynamic model.

B. UAV Benchmark Results

We evaluated *BOWConnect* against the state-of-the-art planners listed in Section V-A across four quadrotor environments: *Quad-1*, *Quad-2*, *Quad-3*, and *Quad-4*. All planners were tested using the same quadrotor dynamic model under identical dynamic constraints. The evaluation metrics include total planning time, trajectory length, success rate, average velocity, and average jerk. The quantitative results are summarized in Table II.

All planners achieve a 100% success rate in every environment, indicating that each method is capable of generating feasible kinodynamic trajectories under the considered quadrotor constraints. However, significant differences arise in computational efficiency and trajectory quality.

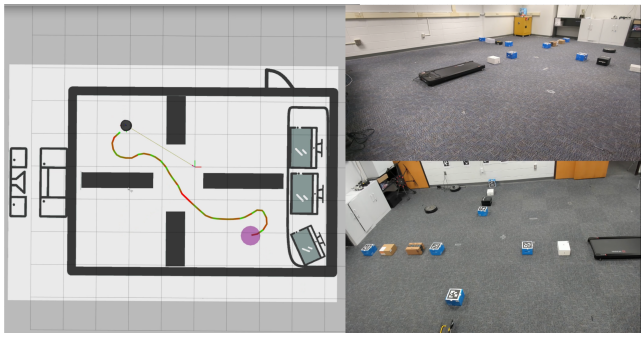
BOWConnect consistently computes solutions in under 0.12 s across all environments (0.117 s, 0.086 s, 0.081 s, and 0.091 s for Quad-1 through Quad-4, respectively). In contrast, classical sampling-based planners such as SST, EST, and KPIECE frequently approach the full time budget (approximately 30–42 s). For example, in *Quad-1*, BOWConnect requires only 0.117 s compared to 7.815 s for RRT and over 29 s for SST and KPIECE. In *Quad-2*, BOWConnect achieves the lowest average planning time among all planners. Although BOW attains slightly lower computation times in *Quad-3* and *Quad-4*, BOWConnect remains within the same order of magnitude while delivering improved trajectory consistency.

BOWConnect produces competitive and often shorter trajectories relative to the baselines. It achieves the shortest average trajectory in *Quad-1* (6.562 m) and *Quad-4* (6.231 m). In *Quad-2* and *Quad-3*, SST yields the shortest mean trajectory lengths (16.715 m and 16.615 m, respectively). While BOW occasionally attains comparable paths, it exhibits significantly higher variance, particularly in *Quad-2* and *Quad-3*, indicating reduced consistency. In contrast, BOWConnect maintains lower variance across environments except in *Quad-2*, reflecting more stable trajectory generation.

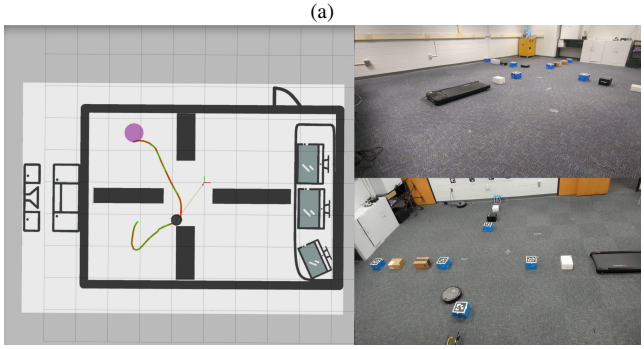
EST and KPIECE tend to produce the highest average velocities across environments. Specifically, EST attains the highest average velocity in *Quad-2* and *Quad-4*, while KPIECE exhibits the highest values in *Quad-1* and *Quad-3*. However, these higher velocities do not correspond to shorter computation times or improved smoothness. BOWConnect maintains moderate average velocities across all environments, indicating balanced dynamic behavior.

Average jerk values indicate that BOW-based planners generally generate smoother control profiles than classical sampling-based methods. In *Quad-1*, BOWConnect achieves the lowest average jerk (0.0279), outperforming RRT (0.0364) and KPIECE (0.0460). Similarly, in *Quad-4*, BOWConnect attains the lowest jerk (0.0246). In *Quad-2*, RRT yields the lowest jerk, and in *Quad-3*, SST produces the lowest jerk. BOWConnect remains competitive in both cases while maintaining substantially lower computation time.

Overall, the results demonstrate that BOWConnect provides a favorable trade-off between computational efficiency, trajectory length, and smoothness under quadrotor dynamic constraints. While some baseline planners occasionally achieve



(a)



(b)

Fig. 2: Experimental validation of the proposed kinodynamic motion planner in a cluttered indoor environment. The left panel shows the planned trajectory visualized in RViz, with the color gradient (green to red) indicating temporal progression. The right panel shows the corresponding real-world UGV experiment conducted in the laboratory under the similar obstacle configuration.

marginally better performance in isolated metrics (e.g., shortest path or lowest jerk in specific environments), BOWConnect consistently delivers fast, robust, and dynamically smooth trajectories across all tested scenarios.

C. Unmanned Ground Vehicle Real-World Experiments

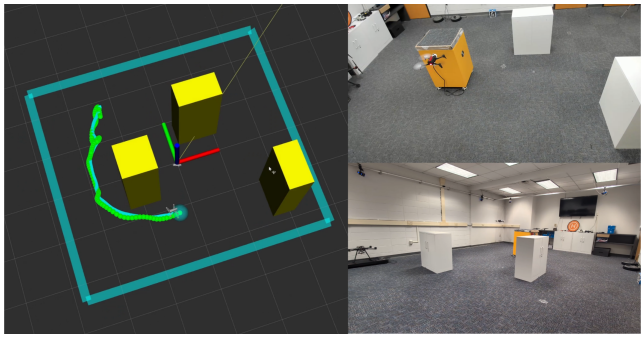
We implemented BOWConnect on a non-holonomic differential-drive Unmanned Ground Vehicle (UGV) operating in a planar indoor environment to replicate the *Narrow Passage-1* benchmark under real-world conditions. The hardware platform consists of an iRobot Create 3 educational robot with a radius of 0.17 m, operating within a bounded 6.5 m × 5.5 m workspace populated with obstacle configurations designed to emulate the constrained geometry of the benchmark scenario. Collision checking was performed using a 2D occupancy grid map constructed from the laboratory layout, while localization was provided by a Vicon motion capture system to ensure high-precision state estimation during execution. The UGV is modeled using a five-dimensional state vector $\mathbf{x} = (x, y, \theta, v, \omega)$, where (x, y) denote planar position, θ is the heading angle, v the linear velocity, and ω the angular velocity, with control input $\mathbf{u} = (v_c, \omega_c)$ representing commanded linear and angular velocities with respect to the body frame. The Unicycle motion model was used for the physical experiments. As illustrated in Fig. 2, the goal pose (marked as a purple disk) was specified interactively in RViz using a 2D goal command, after which BOWConnect generated a dynamically feasible trajectory that was executed by the robot. The primary objective was to validate kinodynamic feasibility and real-time performance, and multiple start–goal configurations were tested across the workspace, with BOWConnect consistently generating feasible trajectories in under 0.15 s. During execution, the maximum linear velocity was limited to 1.0 m/s and the maximum angular velocity to 1.5 rad/s, while the low-level controller operated with a control time step of $\Delta t = 0.05$ s. Across five independent trials with varying start–goal configurations, no collisions or tracking instabilities were observed during execution. The average planning time was 0.12 s, with a maximum of 0.15 s across all runs, demonstrating real-time feasibility on the onboard compute platform.

D. Unmanned Aireal Vehicle Physical Experiments

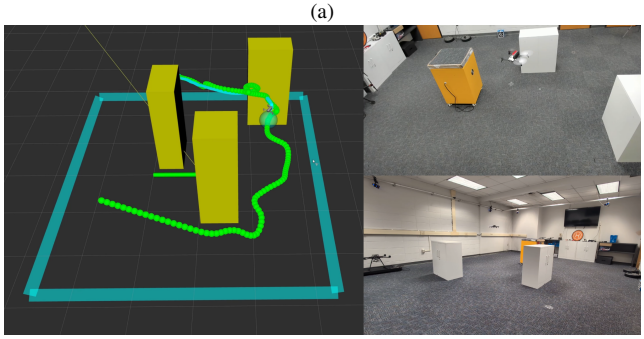
We implemented *BOWConnect* on a 6-DOF quadrotor UAV in a three-dimensional environment with 3D obstacles to demonstrate the capability of the proposed planner in high-dimensional planning problems, as shown in Fig. 3. The experiments were conducted using a Parrot Bebop 2 UAV. The UAV state is modeled using an 8-dimensional vector $\mathbf{x} = (x, y, z, \theta, \dot{x}, \dot{y}, \dot{z}, \dot{\theta})$, which includes position, yaw orientation, and linear and angular velocities. The control input is defined as a 4-dimensional vector $\mathbf{u} = (\dot{x}_c, \dot{y}_c, \dot{z}_c, \dot{\theta}_c)$, representing commanded body-frame linear velocities and yaw rate. In Fig. 3, the left panels show the RViz visualization with the real UAV state and the planned trajectory generated by BOWConnect (green), while the cyan curve denotes the actual trajectory followed by the UAV during execution. The goal position is sent interactively from RViz using the *Nav Goal* tool. The experiments were conducted in a bounded workspace of 6.5 m × 5.5 m × 2.5 m with three box-shaped obstacles placed to replicate the RViz configuration: two identical obstacles of dimensions 0.4 m × 0.8 m × 0.92 m and one obstacle of dimensions 0.64 m × 0.64 m × 1.08 m. The close alignment between the planned trajectory (green) and the executed trajectory (cyan) demonstrates accurate tracking and confirms the dynamic feasibility of the generated paths. Multiple trials were conducted from different start and goal configurations, and in all cases BOWConnect consistently computed collision-free trajectories in under 0.1 s, demonstrating computational efficiency and robustness in high-dimensional kinodynamic planning.

VI. Conclusion

This paper presented BOWConnect, a bidirectional parallel kinodynamic motion planner that combines Bayesian Optimization over Windows with spatial hashing and boundary value problem solving to overcome the sample inefficiency, heuristic limitations, and narrow passage failures of existing kinodynamic planners. By replacing random control sampling with GP-guided acquisition and growing trees from both ends of the planning problem in parallel, BOWConnect consistently achieves 100% success across diverse robotic platforms and challenging environments where classical methods including SST, EST, and KPIECE fail or time out. Real-world experiments on ground and aerial vehicles confirm real-time



(a)



(b)

Fig. 3: BOWConnect quadrotor experiments in representative cluttered environments. The top image shows the simulation setup illustrating the obstacle configuration and the generated kinodynamic trajectory (green). The bottom image presents the corresponding real-world experiment in an indoor test arena, demonstrating consistent trajectory execution and safe navigation under equivalent obstacle placement.

performance under 0.15 seconds, validating the practical applicability of the approach. Future work will explore adaptive worker allocation, planning under uncertainty, and extension to dynamic environments with moving obstacles.

REFERENCES

[1] 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.

[2] 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, 2002.

[3] F. Meng, J. Liu, H. Shi, H. Ma, H. Ren, and M. Q.-H. Meng, "Online time-informed kinodynamic motion planning of nonlinear systems," *IEEE Robotics and Automation Letters*, vol. 9, no. 11, pp. 9589–9596, 2024.

[4] D. Zheng and P. Tsiotras, "Accelerating kinodynamic rrt* through dimensionality reduction," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 3674–3680.

[5] 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.

[6] I. A. Şucan and L. E. Kavraki, "Kinodynamic motion planning by interior-exterior cell exploration," in *Algorithmic foundation of robotics VIII: selected contributions of the eight international workshop on the algorithmic foundations of robotics*. Springer, 2009, pp. 449–464.

[7] 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, pp. i–vii, 2018.

[8] T. Lai, W. Zhi, T. Hermans, and F. Ramos, "Neural kinodynamic planning: Learning for kinodynamic tree expansion," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 11 789–11 795.

[9] M. Faroni and D. Berenson, "Motion planning as online learning: A multi-armed bandit approach to kinodynamic sampling-based planning," *IEEE Robotics and Automation Letters*, vol. 8, no. 10, pp. 6651–6658, 2023.

[10] A. Boeuf, J. Cortés, R. Alami, and T. Siméon, "Enhancing sampling-based kinodynamic motion planning for quadrotors," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 2447–2452.

[11] C. H. Huang, P. Jadhav, B. Plancher, and Z. Kingston, "prrtc: Gpu-parallel rrt-connect for fast, consistent, and low-cost motion planning," *arXiv preprint arXiv:2503.06757*, 2025.

[12] N. Perrault, Q. H. Ho, and M. Lahijanian, "Kino-pax: Highly parallel kinodynamic sampling-based planner," *IEEE Robotics and Automation Letters*, vol. 10, no. 3, pp. 2430–2437, 2025.

[13] 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.

[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, no. 4, pp. 566–580, 2002.

[15] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "Chomp: Gradient optimization techniques for efficient motion planning," in *2009 IEEE international conference on robotics and automation*. IEEE, 2009, pp. 489–494.

[16] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "Stomp: Stochastic trajectory optimization for motion planning," in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 4569–4574.

[17] 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, no. 9, pp. 1251–1270, 2014.

[18] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Proceedings 2000 ICRA. Millennium conference. IEEE international conference on robotics and automation. Symposia proceedings (Cat. No. 00CH37065)*, vol. 2. IEEE, 2000, pp. 995–1001.

[19] 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 *2014 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2014, pp. 2997–3004.

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

[21] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Batch informed trees (bit*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs," in *2015 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2015, pp. 3067–3074.

[22] A. J. La Valle, B. Sakcak, and S. M. LaValle, "Bang-bang boosting of rrts," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 2869–2876.

[23] D. Q. Mayne and H. Michalska, "Model predictive control of nonlinear systems," *1991 American Control Conference*, pp. 2343–2348, 1991.

[24] F. Zhang, N. Li, T. Xue, Y. Zhu, R. Yuan, and Y. Fu, "An improved dynamic window approach integrated global path planning," in *IEEE International Conference on Robotics and Biomimetics*. IEEE, 2019, pp. 2873–2878.

[25] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[26] J. Snape, J. Van Den Berg, S. J. Guy, and D. Manocha, "The hybrid reciprocal velocity obstacle," *IEEE Transactions on Robotics*, vol. 27, no. 4, pp. 696–706, 2011.

[27] C. Quintero-Pe na, C. Chamzas, V. Unhelkar, and L. E. Kavraki, "Motion planning via bayesian learning in the dark," in *International Conference on Intelligent Robots and Systems*. IEEE, 2021, pp. 5634–5641.

[28] J. Ungredda and J. Branke, "Bayesian optimisation for constrained problems," *ACM Transactions on Modeling and Computer Simulation*, vol. 34, pp. 1–26, 2024.

[29] M. Mukadam, J. Dong, X. Yan, F. Dellaert, and B. Boots, "Continuous-time gaussian process motion planning via probabilistic inference," *The International Journal of Robotics Research*, vol. 37, no. 11, pp. 1319–1340, 2018.

[30] S. Raxit, A. A. R. Newaz, P. Padrao, J. Fuentes, and L. Bobadilla, "Bow: Bayesian optimization over windows for motion planning in complex

environments," *IEEE Robotics and Automation Letters*, 2025.
[31] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.