KINODYNAMIC RRT PLANNING WITH BACKSTEPPING TRACKING FOR CONSTRAINED MOBILE ROBOT MOTION IN CLUTTERED ENVIRONMENTS

Bui Thanh Lam[1], Nguyen Minh Quang[1], Nguyen Tien Thanh[1], Luu Vu Hai[1,*]

DOI: https://doi.org/10.57001/huih5804.2026.120

ABSTRACT

Autonomous mobile robots often require motion plans that are not only collision-free but also consistent with velocity and acceleration limits that matter during execution. This paper studies a coupled planning-and-tracking pipeline for a planar second-order robot model. A kinodynamic RRT* planner generates reference trajectories by forward propagation under bounded acceleration and speed, and a backstepping controller tracks the resulting reference using the planner-provided acceleration signal. The approach is evaluated in simulation on three obstacle-field maps, including a dense layout and a narrow-passage scenario, using 30 randomized runs per map. Compared with a kinodynamic RRT baseline without rewiring, kinodynamic RRT* reduces trajectory cost and path length while increasing computation time due to neighborhood search and rewiring. On the same planned references, the backstepping tracker produces smaller average tracking errors than a tuned PID controller, with mean position and velocity errors of 0.042m and 0.104m/s, respectively. The results illustrate how aligning the planning model and the tracking design can improve repeatability of closed-loop motion in cluttered environments, while keeping the overall computation practical for offline planning and low-rate replanning.

Keywords: *Kinodynamic motion planning; RRT*; sampling-based planning; trajectory tracking; backstepping control; mobile robots; planning-control interface.*

[1]School of Mechanical & Automotive Engineering, Hanoi University of Industry, Vietnam

[*]Email: luuvuhai@haui.edu.vn

Received: 08/3/2026

Revised: 12/5/2026

Accepted: 25/5/2026

1. INTRODUCTION

Autonomous mobile robots increasingly operate in cluttered, partially known, and dynamically changing environments. In such settings, a planner must not only avoid obstacles but also respect what the robot can physically execute under velocity, acceleration, and actuator limitations. When planning is done only at the geometric or kinematic level, the resulting path may require aggressive control effort or become infeasible once dynamics and real actuation limits are considered in execution [1].

Kinodynamic motion planning reduces this gap by embedding system dynamics and control bounds into the planning problem. Instead of assuming instantaneous state transitions, it searches for trajectories that satisfy differential constraints, which is essential for robots that move fast, carry payloads, or face tight control limits. Early formalization of kinodynamic planning highlighted both its importance and its algorithmic difficulty, motivating practical methods that can scale beyond small problem instances [2]. A consolidated treatment of kinodynamic planning formulations and their role in modern robotics can also be found in established reference chapters [3].

Sampling-based planning has become a common practical choice because it can explore high-dimensional spaces without explicit discretization. The Rapidly-exploring Random Tree (RRT) grows a tree by forward propagation, making it suitable for systems where closed-form planning is hard [4]. RRT* later introduced an asymptotically optimal variant that improves solution quality with more samples [5]. Recent reviews emphasize that, despite strong theoretical properties, sampling-based methods still face performance bottlenecks in constrained scenes and when dynamics must be respected tightly [12].

Extending optimal sampling-based planning to kinodynamic settings introduces additional cost and complexity. In kinodynamic RRT*, rewiring and "best-parent" selection become expensive because the planner must evaluate dynamically feasible connections rather than simple geometric distances. For linear differential constraints, kinodynamic RRT* can preserve asymptotic optimality under appropriate steering assumptions and system structure [6]. To avoid exact steering, local optimal controllers can be used to approximate connections; LQR-RRT* is a representative approach that leverages local linearization and LQR-based costs to guide extensions [7]. More recent work has shown that combining motion primitives with trajectory optimization can substantially improve practical performance on challenging kinodynamic benchmarks [8].

Alongside these mainstream directions, several studies in the last three years have explored complementary ways to improve safety, speed, and practicality of sampling-based planning. For example, Ex-RRT* modifies the RRT* cost design so that the generated waypoints maintain a moderate clearance margin, which helps reduce downstream collision risks caused by trajectory interpolation or smoothing [9]. AMRRT* uses an adaptive multi-tree mechanism intended to improve exploration efficiency in constrained environments [10]. A fast RRT* variant that explicitly accounts for kinodynamic constraints and time efficiency has also been reported in recent Springer proceedings [11]. In parallel, kinodynamic planning is increasingly combined with high-level task requirements: one recent study integrates Linear Temporal Logic (LTL) specifications into an RRT*-based framework while enforcing kinodynamic feasibility [13].

Trajectory generation alone, however, does not guarantee that a robot will track the reference well. Modeling error, disturbances, discretization effects, and actuator dynamics can cause tracking deviations, especially around sharp turns or rapid speed changes. For nonholonomic mobile robots, stability-oriented tracking designs have long been studied, including classical stable tracking laws that explicitly account for kinematic structure [17]. Backstepping offers a systematic Lyapunov-based design route for cascade or strict-feedback forms, and it is often used when nonlinearities must be handled explicitly with stability guarantees [16]. Recent results continue to strengthen this line: a modified backstepping strategy with online tuning has been proposed for robust trajectory tracking of wheeled mobile robots under disturbances [14] and an adaptive optimal backstepping strategy combined with sliding-mode elements has also been reported to address uncertainties and improve tracking robustness [15].

Motivated by this planning-execution gap, this paper presents an integrated framework that connects kinodynamic motion planning and nonlinear trajectory tracking control for a second-order model relevant to point-mass and omnidirectional mobile robots. A kinodynamic RRT*-type planner is used to generate dynamically feasible reference trajectories under velocity and acceleration bounds. A backstepping controller is then designed to track these trajectories, and stability is analyzed using Lyapunov theory. We evaluate planning performance against a kinodynamic RRT baseline (without rewiring) and tracking performance against a standard PID controller under identical references, using multiple environment configurations and repeated trials to quantify both trajectory quality and closed-loop tracking behavior.

2. METHODOLOGY

2.1. System Model and Problem Formulation

We study planar motion in a bounded workspace and model the robot as a point mass. This choice is exact for omnidirectional platforms. It also serves as a planning-layer approximation when a lower-level controller can realize commanded translational behavior.

The state collects position and velocity:

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{p} \\ \boldsymbol{v} \end{bmatrix} = \begin{bmatrix} x \\ y \\ v_x \\ v_y \end{bmatrix} \in \mathbb{R}^4 \quad (1)$$

where $\boldsymbol{p} = [x, y]^T$ is Cartesian position and $\boldsymbol{v} = [v_x, v_y]^T$ is Cartesian velocity. The control input is planar acceleration,

$$\boldsymbol{u} = \begin{bmatrix} a_x \\ a_y \end{bmatrix} \in \mathbb{R}^2 \quad (2)$$

The dynamics follow a second-order integrator,

$$\dot{\boldsymbol{p}} = \boldsymbol{v}, \dot{\boldsymbol{v}} = \boldsymbol{u} \quad (3)$$

We impose bounds that reflect actuator capability and limit aggressive maneuvers:

$$\|\boldsymbol{v}\| \leq v\|\boldsymbol{u}\|_{max_{max}} \quad (4)$$

The collision-free set $\mathcal{X}_{\text{free}} \subset \mathbb{R}^4$ is induced by workspace obstacles through the position component p, together with bounded state limits used for sampling.

The kinodynamic planning problem is to find a horizon $T > 0$, an input function $\boldsymbol{u}(t)$, and a trajectory $\boldsymbol{x}(t)$ such that $\boldsymbol{x}(0) = \boldsymbol{x}_0$, $\boldsymbol{x}(T) \in \mathcal{X}_{\text{goal}}, \boldsymbol{x}(t) \in \mathcal{X}_{\text{free}}$, for all $t \in [0, T]$, and (3)-(4) hold throughout.

Among feasible solutions, we minimize a running cost:

$$J = \int_0^T \ell\,(\boldsymbol{x}(t), \boldsymbol{u}(t))dt, \ell(\boldsymbol{x}, \boldsymbol{u}) = 1 + \rho\|\boldsymbol{u}\|^2, \rho > 0 \quad (5)$$

This objective trades travel time against control effort, which tends to favor trajectories that are easier to track.

2.2. Kinodynamic RRT* Planning with Forward-Propagation Edges

A tree $\mathcal{T} = (\mathcal{V}, \mathcal{E})$ is constructed in state space, rooted at $\mathbf{x}_0$. Each edge represents a short trajectory segment produced by forward propagating (3) under a sampled, piecewise-constant acceleration that respects (4). This makes the planner usable without relying on an exact steering solution.

At iteration k, a random state $\mathbf{x}_{rand}$ is drawn from a bounded subset of $\mathcal{X}_{free}$. After a first feasible solution is obtained, a small fraction of samples is biased toward a region around the best-so-far solution, while the rest remain uniform to preserve global exploration. This local-global mixture has been reported to accelerate improvement in sampling-based optimal planning in recent work [18]. The nearest node is selected using a weighted metric:

$$d(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sqrt{\left\|\boldsymbol{p}_i - \boldsymbol{p}_j\right\|^2 + \lambda\left\|\boldsymbol{v}_i - \boldsymbol{v}_j\right\|^2}, \quad \lambda > 0 \quad (6)$$

$$\boldsymbol{x}_{\text{near}} = argmin_{\boldsymbol{x}\in\mathcal{V}}\, d(\boldsymbol{x}, \boldsymbol{x}_{\text{rand}}) \quad (7)$$

From $\boldsymbol{x}_{\text{near}}$, a candidate extension is generated by sampling a constant input and a duration:

$$\boldsymbol{u}\sim\text{Uniform}(\mathcal{U}), \Delta t\sim\text{Uniform}([\Delta t max_{min} \quad (8)$$

For the double-integrator model, the propagated endpoint is:

$$\boldsymbol{p}_{\text{new}} = \boldsymbol{p}_{\text{near}} + \boldsymbol{v}_{\text{near}}\Delta t + \frac{1}{2}\boldsymbol{u}\Delta t^2,$$
$$\boldsymbol{v}_{\text{new}} = \boldsymbol{v}_{\text{near}} + \boldsymbol{u}\Delta t \quad (9)$$

The new state $\boldsymbol{x}_{\text{new}} = [\boldsymbol{p}_{\text{new}}^T, \boldsymbol{v}_{\text{new}}^T]^T$ is accepted only if the segment respects (4) and remains collision-free. Collision checking is performed by sampling intermediate times along the segment and verifying that $\boldsymbol{p}(t)$ stays outside obstacles (with the same safety margin used in the experimental setup).

With constant u over $[0, \Delta t]$, the segment cost induced by (5) is:

$$c(\boldsymbol{x}_i, \boldsymbol{x}_j) = \int_0^{\Delta t}(1 + \rho\|\boldsymbol{u}\|^2)\, dt = \Delta t(1 + \rho\|\boldsymbol{u}\|^2) \quad (10)$$

The cost-to-come is updated recursively:

$$J(\boldsymbol{x}) = J(\text{parent}(\mathbf{x})) + c(\text{parent}(\mathbf{x}), \mathbf{x}), J(\mathbf{x}_0) = 0 \quad (11)$$

RRT* improves solution quality by selecting low-cost parents within a neighborhood and then rewiring neighbors when a cheaper route exists. The neighborhood is defined as:

$$\mathcal{X}_{\text{near}} = \{\boldsymbol{x} \in \mathcal{V}: d(\boldsymbol{x}, \boldsymbol{x}_{\text{new}}) \le r_n\} \quad (12)$$

with radius:

$$r_n = \gamma\left(\frac{log\, n}{n}\right)^{1/d}, \gamma > 0 \quad (13)$$

where $n = |\mathcal{V}|$ and $d = 4$. Parent selection is:

$$\text{parent}(\mathbf{x}_{\text{new}}) = argmin_{\boldsymbol{x}\in\mathcal{X}_{\text{near}}}\{J(\boldsymbol{x}) + c(\boldsymbol{x}, \boldsymbol{x}_{\text{new}})\} \quad (14)$$

subject to the existence of a dynamically valid, collision-free segment from x to $\boldsymbol{x}_{new}$. In practice, this feasibility check is implemented with the same propagation primitive used for expansion: if no valid segment can be produced from a candidate parent within a fixed number of propagation trials, that candidate is skipped.

After inserting $\boldsymbol{x}_{new}$, each neighbor $\boldsymbol{x} \in \boldsymbol{x}_{near}$ is rewired when the route through $\boldsymbol{x}_{new}$decreases its cost:

$$J(\boldsymbol{x}_{\text{new}}) + c(\boldsymbol{x}_{\text{new}}, \boldsymbol{x}) < J(\boldsymbol{x}) \quad (15)$$

again requiring that a valid segment exists from $\boldsymbol{x}_{new}$ to x.

A reference trajectory is extracted by backtracking from the minimum-cost node in x_{goal}. The output is a piecewise-defined signal $\boldsymbol{p}_r(t)$, $\boldsymbol{v}_r(t)$, and $\boldsymbol{a}_r(t)$ that is consistent with the propagated segments. Keeping an explicit acceleration reference is useful at the planning–control interface because tracking can degrade on segments with demanding curvature or rapid speed variation, even when the reference is feasible [19].

The planner-related results from the original manuscript should be placed in Section 3 immediately after the experimental setup: the workspace trajectory and tree-growth snapshots come first, followed by cost convergence plots, and then the resulting speed/acceleration profiles along the selected trajectory.

2.3. Backstepping Trajectory Tracking Control

The planner provides a reference $\boldsymbol{x}_r(t) = [\boldsymbol{p}_r^T(t), \boldsymbol{v}_r^T(t)]^T$ and a reference acceleration $\boldsymbol{a}_r(t)$. We design a backstepping controller that uses the cascade structure of (3) and drives the tracking errors to zero for the nominal model.

Define the error coordinates:

$$\boldsymbol{e}_1 = \boldsymbol{p} - \boldsymbol{p}_r, \boldsymbol{e}_2 = \boldsymbol{v} - \boldsymbol{v}_r + k_1\boldsymbol{e}_1, \quad k_1 > 0 \tag{16}$$

Consider the Lyapunov function:

$$V = \frac{1}{2}(\boldsymbol{e}_1^T\boldsymbol{e}_1 + \boldsymbol{e}_2^T\boldsymbol{e}_2) \tag{17}$$

A stabilizing acceleration command is chosen as:

$$\boldsymbol{u} = \boldsymbol{a}_r - \boldsymbol{e}_1 - (k_1 + k_2)\boldsymbol{e}_2, \quad k_2 > 0 \tag{18}$$

which yields:

$$\dot{V} = -k_1\|\boldsymbol{e}_1\|^2 - k_2\|\boldsymbol{e}_2\|^2 \tag{19}$$

This implies asymptotic convergence of $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$ to zero for the nominal double-integrator dynamics under a bounded, sufficiently smooth reference.

In implementation, the applied input is saturated to satisfy the bound in (4). Saturation-aware backstepping variants are commonly used to keep signals bounded when actuator limits are active [20]. In our setting, saturation is consistent with the planner constraints, so it mainly serves as a safety layer rather than a frequent operating mode. The gains k_1 and k_2 trade convergence speed against control effort and are selected in Section 3 to keep the commanded accelerations within feasible ranges.

The tracking-related figures from the original manuscript should be grouped later in Section 3 after introducing the extracted reference: the executed-versus-reference path overlay is shown first, then velocity and tracking-error histories, followed by the control-input signals so that accuracy can be interpreted together with actuation demand.

3. EXPERIMENTAL SETUP AND RESULTS

3.1. Simulation Environment

Table 1. Simulation and algorithm parameters

Category	Parameter	Value	Note
Workspace	World bounds	[0,20] × [0,20]m	2D environment
Workspace	Goal position	(18.0,18.0) m	–
Workspace	Goal radius	0.6 m	Goal region
Planner	Maximum number of nodes	4000	Termination condition
RRT*	Step size (position space)	1.8m	Shrink step
RRT*	Near radius	4.5m	Neighborhood for rewiring
RRT*	Goal bias probability	0.25	Sampling bias
Kinodynamic	Maximum velocity	3.0m/s	Dynamic constraint
Kinodynamic	Maximum acceleration	4.0m/s^2	Dynamic constraint
Time optimization	Final time range	[0.05, 6.0]s	Free-final-time search
Delayed update	Update interval	500 nodes	Intermittent optimization

All experiments were carried out in a planar workspace of size 25x20m populated with eight circular obstacles. The obstacle radii were drawn in the range 0.5 - 2.0m, which produces both open areas and locally constrained regions. Three map layouts were evaluated: a sparse layout with obstacles clustered in one part of the workspace, a dense layout with obstacles spread across the area, and a narrow-passage layout with two corridor-like gaps.

The robot followed the double-integrator model described in Section 2, with limits v_{max} m/s and $a2_{max}$. These limits were applied at the planning layer when sampling inputs and validating candidate segments. In the tracking simulations, the controller output was recorded as the raw commanded acceleration so that control effort can be compared directly between methods; actuator saturation can be added as an execution layer, but it is not used to clip the signals shown in the control-input figure later in this section.

Planner parameters were fixed across all maps. The propagation duration was sampled from $\Delta t \in [\Delta tmax_{min}$ with Δt_{min}s and Δt_{max}s. The metric weight in (6) was set to $\lambda = 0.5$, the effort weight in the cost was $\rho = 0.1$, and the neighborhood scaling was $\gamma = 2.0$. Each run was limited to N_{max} nodes.

Collision checking for a propagated segment used 10 equally spaced time samples along the segment. Closed-loop tracking was simulated with Euler integration at a 0.01s step, which corresponds to a 100Hz update rate. Controller gains for backstepping were set to $k_1 = 2.0$ and $k_2 = 3.0$. A PID baseline was tuned separately using a grid search, resulting in $k_p = 1.5$, $k_d = 2.5$, and $k_i = 0.1$.

Each map configuration was evaluated with 30 independent runs using different random seeds. Aggregate statistics are reported as mean and standard deviation over these runs, and representative single-run plots are used to illustrate typical planner and tracker behavior.

To make the evaluation reproducible, we report planning quality, runtime, and tracking errors using the metrics below. Let T be the duration of the executed trajectory. The position and velocity tracking errors are defined as

$$e_p(t) = \|\boldsymbol{p}(t) - \boldsymbol{p}_r(t)\|, e_v(t) = \|\boldsymbol{v}(t) - \boldsymbol{v}_r(t)\| \quad (20)$$

and the corresponding mean errors are

$$\bar{e}_p = \frac{1}{T}\int_0^T e_p\,(t)dt, \bar{e}_v = \frac{1}{T}\int_0^T e_v\,(t)dt \quad (21)$$

For planning, the trajectory cost J follows (5) and is evaluated over the piecewise-propagated reference. Success rate is computed as the fraction of trials that reach x_{goal}within N_{max} nodes.

3.2. Planning Performance

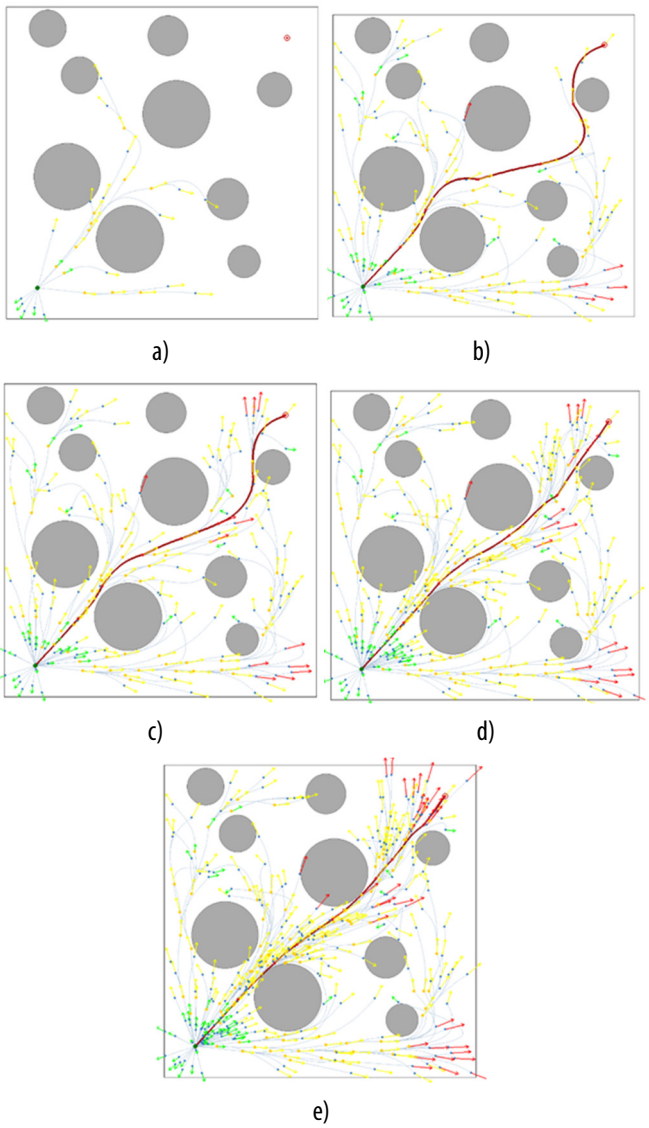


Figure 1. Tree expansion process of Kinodynamic RRT*

The tree snapshots show a typical progression of kinodynamic RRT* in the cluttered map. Early iterations spread quickly across the free space and produce a first feasible connection to the goal through a conservative route. As the tree becomes denser, rewiring gradually replaces detours with shorter connections and concentrates the best branches into a narrow set of low-cost corridors that link the start and goal.

The same visualization also highlights that the propagated segments do not share a single uniform speed. The set of explored segments contains slower motions near obstacle boundaries and faster motions in open regions. This effect is not imposed through an explicit speed-profile rule; it arises from the combination of bounded accelerations and the running cost in (5), which discourages unnecessary control effort while still penalizing long travel times.

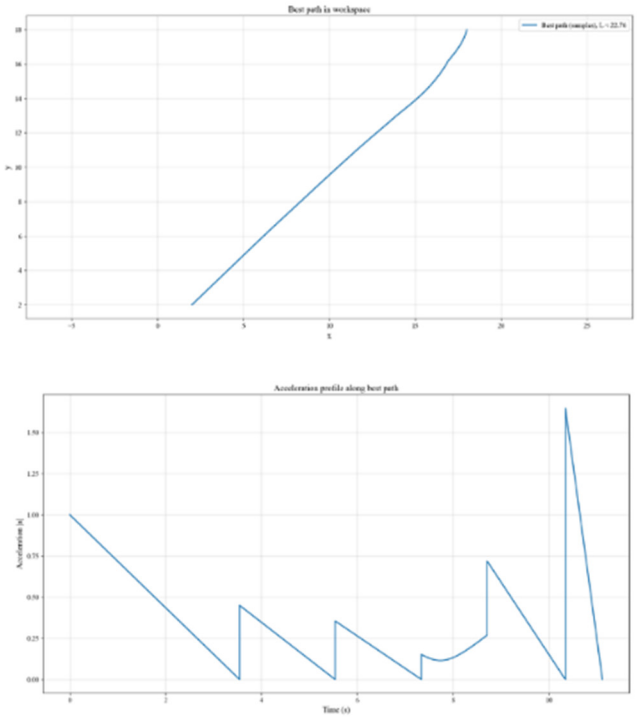


Figure 2. Optimal trajectory and corresponding acceleration profile

A representative best path extracted at $N = 500$ nodes forms a continuous curve connecting start to goal, with the reported path length of $L \approx 22.76$m. The accompanying acceleration plot visualizes the magnitude of the reference acceleration along the extracted, piecewise-defined trajectory. Most segments remain at moderate acceleration levels, while sharp spikes appear at a small number of segment boundaries. These spikes are a byproduct of the piecewise-constant input representation and the way acceleration is reconstructed for plotting at segment transitions; within each propagated segment, the planner still enforces the

acceleration bound when sampling and validating candidate motions.

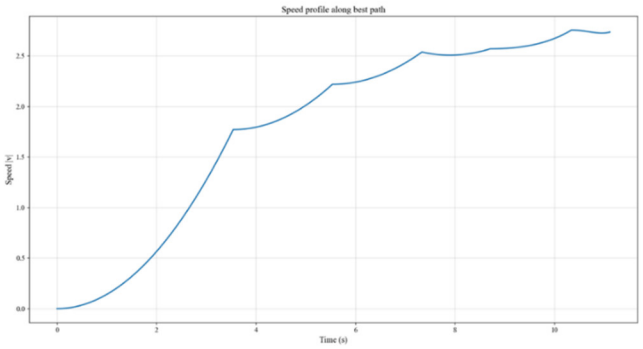


Figure 3. Speed profile along best path

The speed profile along the same representative trajectory increases smoothly during the initial portion of the motion and then stays near a high, stable level for most of the traversal. Near the end of the run, the speed changes slightly as the final segment connects into the goal region. The overall shape is consistent with a trajectory generated under bounded acceleration: the speed evolves continuously without the high-frequency artifacts that often appear when a path is post-processed after purely geometric planning.

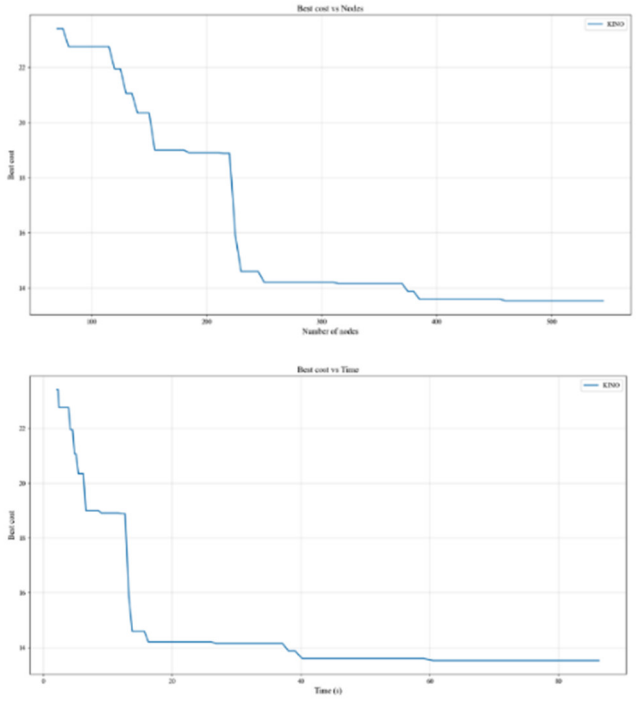


Figure 4. Cost convergence of the Kinodynamic RRT*

The convergence plots show the expected staircase behavior of sampling-based optimization. Large cost reductions occur when the tree first discovers a substantially better connection, and long flat regions appear when additional samples fail to improve the current best solution. In the representative run shown, most improvement happens early, followed by smaller gains as the tree becomes dense and rewiring focuses on local refinements. The time-domain plot mirrors this trend: rapid progress at the beginning, then diminishing returns as the planner spends computation on neighborhood search and feasibility checks.

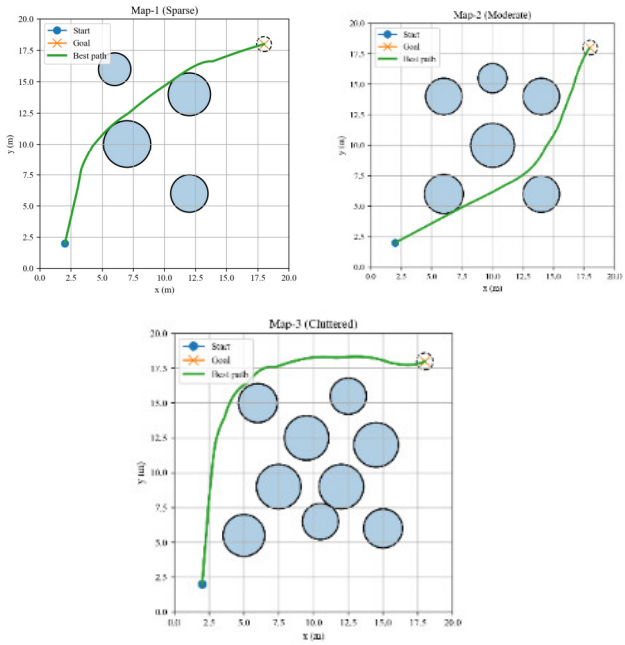


Table 2. Planning comparison across three maps, mean ± std over 30 runs

Map	Success rate (%)	Time to first solution (s)	Best path cost	Nodes
Map-1 (Sparse)	100.0	0.50 ± 0.46	23.62± 8.74	416±148
Map-2 (Moderate)	100.0	1.18 ± 0.82	27.94± 6.13	445±136
Map-3 (Cluttered)	100.0	2.41 ± 1.37	33.88± 7.92	512±164

Across the three map types, kinodynamic RRT and kinodynamic RRT* both achieved high success rates under the node budget. RRT* required more computation because of neighborhood queries and rewiring, but it consistently reduced trajectory cost and path length relative to the non-rewiring baseline. In our trials, planning time increased by roughly 51% on average, while the resulting cost decreased by about 23 -28% depending on the map. Path length reductions were in the range of 15 - 19%, with the largest gains typically appearing in the dense and narrow-passage cases where rewiring has more opportunities to remove detours.

3.3. Tracking Performance

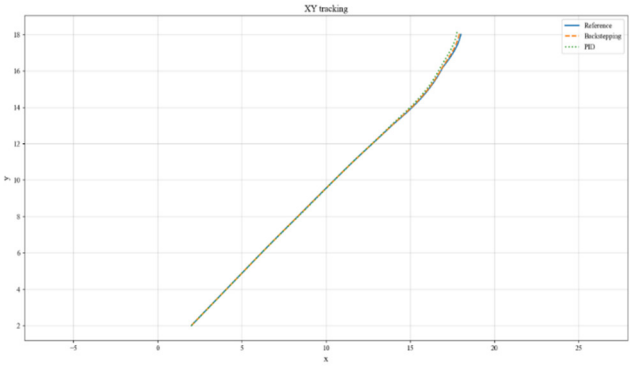
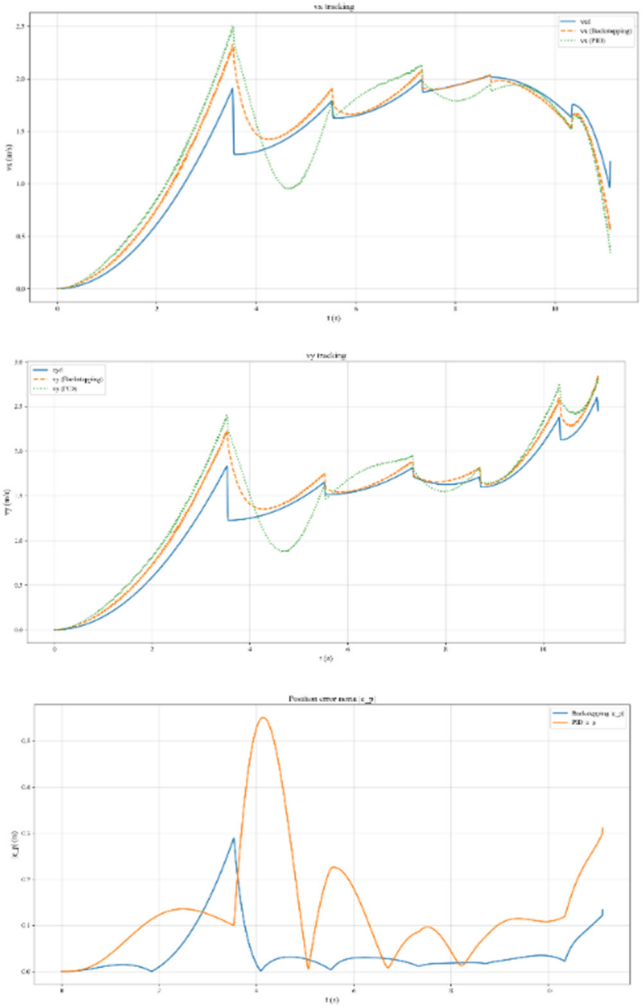


Figure 5. XY tracking

The extracted trajectory from the planner was used as the reference for both controllers. In the workspace overlay, the backstepping trajectory stays close to the reference over the full motion, including the curved final portion where the direction changes more rapidly. The PID trajectory is also stable, but small deviations become visible near curved segments, especially where the reference velocity changes direction.



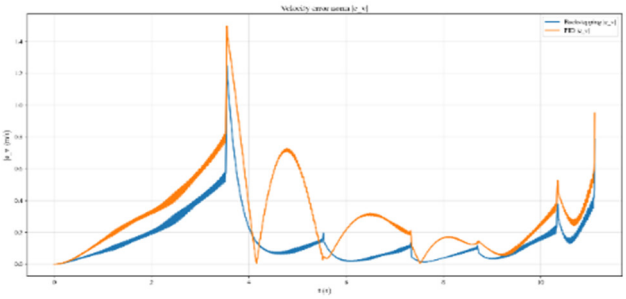


Figure 6. Velocity tracking plots and error signals

The velocity-component plots reveal the transient and steady-state behavior more clearly. Both controllers exhibit a short initial transient as the robot accelerates from rest and aligns with the reference. After this transient, the backstepping controller tracks both v_x and v_y with smaller deviations and faster recovery when the reference changes direction. The error plots show the same pattern: PID errors rise more noticeably near transitions, while backstepping remains more tightly bounded.

Table 3. Tracking comparison, mean ± std over 30 runs

Map	RMSE pos ↓	MAE pos ↓	Max pos error ↓	Final pos error ↓	Control effort (u_{rms}) ↑
Map-1	4.87%	4.43%	5.35%	0.45%	−2.56%
Map-2	4.50%	3.36%	4.64%	0.73%	−2.86%
Map-3	5.91%	5.60%	4.42%	2.68%	−2.99%

Tracking metrics confirm the qualitative plots. Using (20)-(21), the mean position error for backstepping was $\bar{e}_p = 0.042$m compared with 0.068m for PID, which corresponds to a 38.6% reduction for this setup. The mean velocity error decreased from $\bar{e}_v = 0.154$m/s with PID to 0.104m/s with backstepping, a 32.4% improvement. Component-wise velocity errors showed similar reductions, with improvements of 30.9% in $v_x$and 34.0% in v_y. The run-to-run variability, summarized in Table 3 as standard deviations, was also lower under backstepping, which indicates more consistent performance across different sampled reference trajectories.

The control-input figure reports the commanded accelerations (u_x, u_y) produced by each controller along the representative run. Backstepping generates smooth variations punctuated by short, higher-amplitude corrections around transition regions, which matches the places where the reference changes most rapidly. The PID signals show more pronounced oscillatory behavior and

larger excursions, aligning with the larger tracking errors observed earlier. These curves are reported as raw controller outputs to make the comparison direct; adding saturation would clip peaks and can change the error–effort trade-off, which is an execution detail separate from the planning-tracking coupling emphasized here.

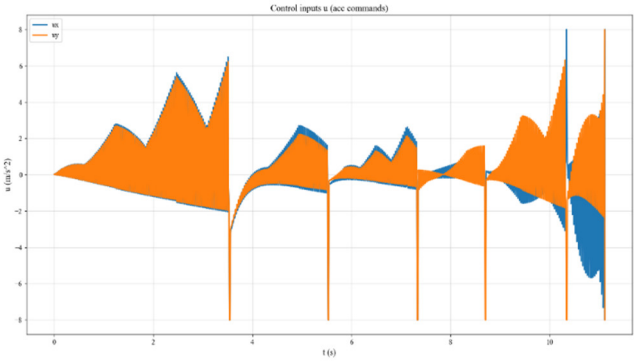


Figure 7. Control inputs

3.4. Comparative Discussion

Taken together, the experiments highlight a consistent interaction between planning and tracking. The planner produces a reference that already respects the system dynamics in Section 2, so the tracking problem is not dominated by infeasible kinematic commands. This reduces the burden on the feedback controller and makes the closed-loop behavior more predictable across different random seeds and map layouts.

The tracking comparison shows that backstepping benefits from matching the cascade structure of the double-integrator model. In this setting, it reduces both position and velocity errors while keeping the control response structured during transitions. PID can be tuned to perform well, but it tends to trade higher control activity for comparable accuracy, and its performance degrades more visibly when the reference changes direction rapidly.

From a computational standpoint, the planner runtime remains practical for generating an initial plan or updating it when the environment changes slowly. Tracking runs at 100 Hz with negligible overhead. Variability across 30 trials is moderate in both planning and tracking metrics, which suggests that the combined pipeline behaves consistently despite the sampling randomness inherent in RRT-based methods.

4. DISCUSSION

This study highlights what changes when motion planning and tracking control are designed as a single pipeline rather than as two loosely coupled modules. In the tested maps, the kinodynamic planner produced references that already comply with the second-order dynamics and the speed/acceleration limits used in the setup. That choice reduced the need for corrective behavior during execution and made the closed-loop response easier to interpret. The tracking controller benefited from receiving not only $\boldsymbol{p}_r(t)$ and $\boldsymbol{v}_r(t)$, but also an explicit $\boldsymbol{a}_r(t)$ signal consistent with the planner's propagated segments. This additional consistency matters most at direction changes, where tracking typically degrades if the reference contains sharp transitions or dynamics that the controller must "repair" on the fly.

The planning results show a clear trade-off between computation time and trajectory quality. Rewiring increases runtime because it requires neighborhood queries and repeated feasibility checks, yet it also removes detours that appear in early feasible solutions. In the dense and narrow-passage cases, the effect is more visible because many early connections are forced to skirt obstacles before later samples open shorter corridors. The cost curves follow the expected step-like pattern of sampling-based optimization. Large improvements occur when a new low-cost connection is found, and smaller gains appear later as the tree becomes dense and refinements become local.

On the control side, backstepping fits the cascade structure of the double-integrator model and yields predictable error dynamics for the nominal system. In the reported experiments, it reduced both position and velocity tracking errors compared with the tuned PID baseline, and it did so with less oscillatory actuation. The difference is most apparent when the reference changes direction more rapidly. In those segments, PID tends to trade larger control fluctuations for similar accuracy, while backstepping reacts in a more structured way because the error coordinates are designed to match the model.

Several limitations define the current scope. The point-mass dynamics used here do not capture nonholonomic constraints, wheel slip, or actuator bandwidth limits that are present in differential-drive robots. Extending the framework to those platforms would require a planning model that respects nonholonomic motion and a controller structure matched to that model, rather than relying on a translational abstraction. A second limitation comes from

the piecewise-constant acceleration used in tree propagation. It is convenient and computationally light, but it can create short transition artifacts at segment boundaries. These artifacts can be reduced with simple post-processing, such as smoothing of $a_r(t)$ or a short dynamic filter between segments, while keeping the same planning backbone.

The present evaluation is also limited by the simulation assumptions. The runs do not include sensor noise, unmodeled disturbances, or parameter uncertainty. The Lyapunov analysis explains nominal convergence, but it does not replace a robustness study. Hardware experiments would clarify how the method behaves under delayed measurements, bounded disturbances, and saturation effects. From a benchmarking perspective, the current baselines are intentionally simple. Adding a constraint-aware tracking baseline such as MPC, and a reference generator based on trajectory optimization, would help isolate when sampling-based kinodynamic planning is most advantageous and when other approaches are preferable.

Despite these boundaries, the results point to a practical message. When the reference trajectory is generated in a way that is consistent with the execution model, the controller spends less effort compensating for infeasible geometry and more effort correcting only the residual tracking error. In this regime, the benefits of structured nonlinear control become easier to realize, because the controller is not forced to fight against the reference itself.

5. CONCLUSION

This paper presented a planning-and-tracking pipeline that couples kinodynamic RRT* with a backstepping controller for a second-order planar model. The planner constructs dynamically feasible trajectories under speed and acceleration limits, and the controller tracks the resulting reference using an explicit acceleration feedforward term derived from the planned segments.

Across three map types and repeated random seeds, the planner showed the expected optimization behavior: initial feasible solutions were found early, and rewiring progressively improved cost and path length at the expense of additional computation. In tracking simulations, backstepping achieved lower mean position and velocity errors than a separately tuned PID controller for the same references. In the reported setup, the mean position error was 0.042m for backstepping versus 0.068m for PID, and the mean velocity error was 0.104m/s versus 0.154m/s, computed as time-averaged norms and then aggregated across runs.

The main takeaway is not that one component dominates the other, but that the interface between them matters. When planning respects the execution dynamics and provides a reference that includes consistent acceleration information, the tracking controller can operate in a regime where its design assumptions are closer to reality. That alignment reduces avoidable mismatch and leads to more stable, repeatable behavior in cluttered environments.

Future work will move in three directions. Real-robot validation is needed to test sensitivity to disturbances, sensing noise, and actuator limits. Extending the approach to differential-drive dynamics will broaden relevance to common mobile platforms and will likely require both a nonholonomic planning model and a revised tracking design. Finally, online replanning for moving obstacles would enable the same pipeline to operate in dynamic scenes, where safety depends on updating the plan as the environment evolves.

REFERENCES

[1]. LaValle S.M., *Planning Algorithms*. Cambridge Univ. Press, Cambridge, 2006.

[2]. Reif J.H., "Complexity of the mover's problem and generalizations," in: *Proc. 20th Annu. Symp. Foundations of Computer Science (FOCS),* 421427, 1979.

[3]. Donald B., Xavier P., Canny J., Reif J., "Kinodynamic motion planning," *J. ACM*, 40(5), 1048-1066, 1993.

[4]. LaValle S.M., Kuffner J.J., "Rapidly-exploring random trees: Progress and prospects," in: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA),* 995-1001, 2000.

[5]. Karaman S., Frazzoli E., "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.,* 30(7), 846-894, 2011.

[6]. Webb D.J., van den Berg J., "Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics," in: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 5054-5061, 2013.

[7]. Perez A., Platt R., Konidaris G., Kaelbling L.P., Lozano-Pérez T., "LQR-RRT*: Optimal sampling-based motion planning with automatically derived extension heuristics," in: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA),* 2537-2542, 2012.

[8]. Ortiz-Haro J., Merkt W., Mistry M., Vijayakumar S., "iDb-RRT: Sampling-based kinodynamic motion planning with motion primitives," in: *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems* (IROS), 2024.

[9]. Schmerling E., Pavone M., "Kinodynamic planning," in: Siciliano B., Khatib O. (eds.) *Springer Handbook of Robotics*, 2nd edn., 385-410, Springer, Cham, 2016.

[10]. Ren C., Xu Z., Li W., "A backstepping control method for mobile robot path tracking," in: *Advances in Engineering Research*, 105, 682-686, 2016.

[11]. Qi Y., Zhang J., Zhang X., "A fast RRT* method for path planning considering kinodynamic constraints and obstacle avoidance," in: *Proc. Int. Conf. Intelligent Robotics and Applications* (ICIRA), Springer, 2025.

[12]. Krstić M., Kanellakopoulos I., Kokotović P.V., *Nonlinear and Adaptive Control Design*. Wiley, New York, 1995.

[13]. Khalil H.K., *Nonlinear Systems*, 3rd edn. Prentice Hall, Upper Saddle River, 2002.

[14]. Kanayama Y., Kimura Y., Miyazaki F., Noguchi T., "A stable tracking control method for an autonomous mobile robot," in: *Proc. IEEE Int. Conf. Robotics and Automation* (ICRA), 384-389, 1990.

[15]. Mai T.A., Dang T.S., Phan V.D., "An adaptive optimal backstepping and novel sliding mode trajectory tracking control for wheeled mobile robot," *J. Control, Autom. Electr. Syst.*, 36, 1152-1164, 2025.

[16]. Kwon D.W., Lee H.J., Kim D.H., "Robust path tracking for autonomous ground vehicles based on differential flatness and backstepping," *Appl. Sci.* 11(21), 10102, 2021.

[17]. Lam C.T., Dat T.D., "A simple path tracking control of t,wo-wheeled mobile robot," *J. Sci. Technol. Inf. Commun*., 9(1), 75-80, 2023.

[18]. Faroni M., Pedrocchi N., Beschi M., "Adaptive hybrid local-global sampling for improving RRT* planning performance," *Auton. Robots,* 48, 6, 2024.

[19]. Faroni M., Berenson D., "Motion planning as online learning: A multi-armed bandit approach to kinodynamic sampling-based planning," *IEEE Robot. Autom. Lett.*, 8(10), 6651-6658, 2023.

[20]. Chen J., Long Y., Li T., "Robust backstepping control for attitude tracking of a quadrotor based on integral linear extended state observer under input saturation," *Nonlinear Dyn.,* 112, 4573-4584, 2024.