Modeling, Identification and Control, Vol. 45, No. 1, 2024, pp. 15–28, ISSN 1890–1328

CostTrust: A Fast-Exploring, Iteratively Expanding Frontier-Based Kinodynamic Motion Planner

Fetullah Atas [1] Grzegorz Cielniak [2] Lars Grimstad [1]

[1] *Faculty of Science and Technology (REALTEK), Norwegian University of Life Sciences, N-7491 Aas, Norway. E-mail:* {fetullah.atas,lars.grimstad}@nmbu.no

[2] *Lincoln Centre for Autonomous Systems (LCAS), University of Lincoln, Lincoln, UK. E-mail:* gcielniak@lincoln.ac.uk

Abstract

Sampling-based motion planning has recently experienced considerable advancements, particularly in the domain of geometric motion planning for diverse robotic systems. Nonetheless, kinodynamic motion planning, which additionally considers a robot's kinematics and dynamics to generate a motion plan, remains an open challenge, necessitating further research. Kinodynamic planning, inherently more complex than geometric planning, mandates that the planner not only adheres to motion constraints but also account for system dynamics, including limitations in velocity and acceleration.

Furthermore, kinodynamic planning often requires the navigation of extensive state and control spaces, rendering the process both computationally demanding and time-consuming. To effectively tackle kinodynamic motion planning, our proposed approach introduces a dynamic balance between exploration and exploitation, continuously adjusted throughout the execution. Our bi-directional and multi-threaded algorithm is specifically tailored to fulfill the efficiency requisites of kinodynamic motion planning. Our comprehensive benchmarks, conducted on an Ackermann-steered robot and a dynamic quadrotor, demonstrate that our method notably outperforms state-of-the-art baselines in terms of solution rate percentage and path cost.

To facilitate accessibility and further research within the community, we have made the implementation of our method available *. It is integrated with the Open Motion Planning Library (OMPL), a widely utilized resource in the field, enhancing our approach's practical applicability †

Keywords: Motion Planning, Kinodynamic Planning, Sampling-Based Motion Planning

1 Introduction

Motion planning constitutes a fundamental aspect of robotics, focusing on developing algorithms and techniques that enable a robot or an autonomous agent to have a trajectory from an initial to a goal state. This process must adhere to motion constraints and avoid obstacles. Broadly, motion planning is categorized into geometric and kinodynamic planning, differentiated by the nature of the constraints involved. Geometric planning is primarily concerned with generating a collision-free path, disregarding the dynamic constraints of the

*https://github.com/NMBURobotics/ompl
†https://github.com/NMBURobotics/vox_nav

doi:10.4173/mic.2024.1.2

robot. Conversely, kinodynamic planning delves into optimizing a robot's motion by meticulously considering its kinematics and dynamics, such as limitations on acceleration, velocity, and specific motion constraints.

Kinodynamic planning finds its application in various domains, including robotic manipulation, aerospace systems, unmanned aerial vehicles, and autonomous vehicles. The dynamics of these systems typically involve constraints on velocity, acceleration, and other parameters, which inherently limit their maneuverability. This limitation presents a significant challenge for kinodynamic planners, necessitating a keen focus on the system's maneuverability during method development. Consequently, this elevates the complexity and intricacy of kinodynamic planning problems.

Thanks to their efficiency, sampling-based methods have been the mainstay of kinodynamic planning. The basic idea behind these methods is to explore the state space by generating many samples and then use the information gathered to guide the search for a feasible trajectory. These methods benefit systems with high-dimensional state spaces and complex constraints, as they can efficiently explore the state space without needing an explicit representation of the entire state space.

Our method represents a progressive stride in kinodynamic motion planning, focusing on a strategy that harmoniously balances exploration and exploitation. Recent advancements in geometric motion planning, exemplified by contributions such as ABIT* and BIT* Strub and Gammell (2020b); Gammell et al. (2020), emphasize exploiting established solutions through informed sampling. This technique selectively samples regions with the potential to enhance the current solution, incorporating these samples into the underlying data structure, typically a tree or graph. However, adapting this strategy to kinodynamic motion planning presents unique challenges, primarily due to the added complexity of motion constraints inherent in dynamic systems. For instance, establishing a connection between two states in such systems is not straightforward, as it requires a steering function capable of solving a two-point boundary value problem (BVP). Addressing these limitations, we introduce a tree-search methodology that leverages random forward propagation of system states while conforming to the kinematic and dynamic constraints of the system. A notable advantage of our approach is its independence from boundary value problems (BVPs) that necessitate solving complex differential equations for diverse dynamic systems. Furthermore, our method excels in rapid exploration, attributed to its selective mechanism for choosing *frontiers*. Simultaneously, it can continuously refine and improve the current solution, demonstrating effective exploitation.

Our research introduces several significant advancements in kinodynamic planning:

- Development of a heuristic-like function that effectively balances rapid pre-solution exploration with post-solution exploitation, enhancing solution refinement—a capability not prevalent in existing state-of-the-art planners.
- Implementation of a multi-threaded, bi-directional tree search method, significantly surpassing the capabilities of current state-of-the-art kinodynamic planners in terms of efficiency.
- Integration of our planner into the widely-used Open Motion Planning Library (OMPL), enhancing its usability and practical application, and provision of benchmarking code to facilitate further research and comparisons.

2 Related Work

For the last two decades, sampling-based motion planning algorithms have gained popularity in robotic motion planning due to their effectiveness and flexibility in handling complex systems and environments. Among the most well-known and widely-used algorithms in this category are the Probabilistic RoadMap (PRM) algorithm Kavraki et al. (1996) and the Rapidly-exploring Random Tree (RRT) algorithm Lavalle and Kuffner (2000), along with a plethora of their variants. To gain a deeper understanding of the optimality properties of sampling-based planners, the authors in Karaman and Frazzoli (2011) conducted a comprehensive examination of their completeness and guarantees. Several libraries and frameworks have been developed to facilitate the use, and implementation of sampling-based motion planning algorithms, such as the Open Motion Planning Library (OMPL) Sucan et al. (2012). This library contains a wide range of sampling-based planners, including RRT* Karaman and Frazzoli (2011) and PRM Karaman and Frazzoli (2011), as well as more recent algorithms that incorporate both graph and tree structures, such as BIT* Gammell et al. (2020), ABIT Strub and Gammell (2020b) and AIT* Strub and Gammell (2020a). Additionally, some planners utilize parallelized approaches, such as CFOREST Otte and Correll (2013) and AnytimePartShortening (APS) Luna et al. (2013), which allow for concurrent execution of multiple planners on different threads, resulting in improved performance overall.

To extend the applicability of sampling-based planners to systems with dynamics (e.g., velocities, ac-

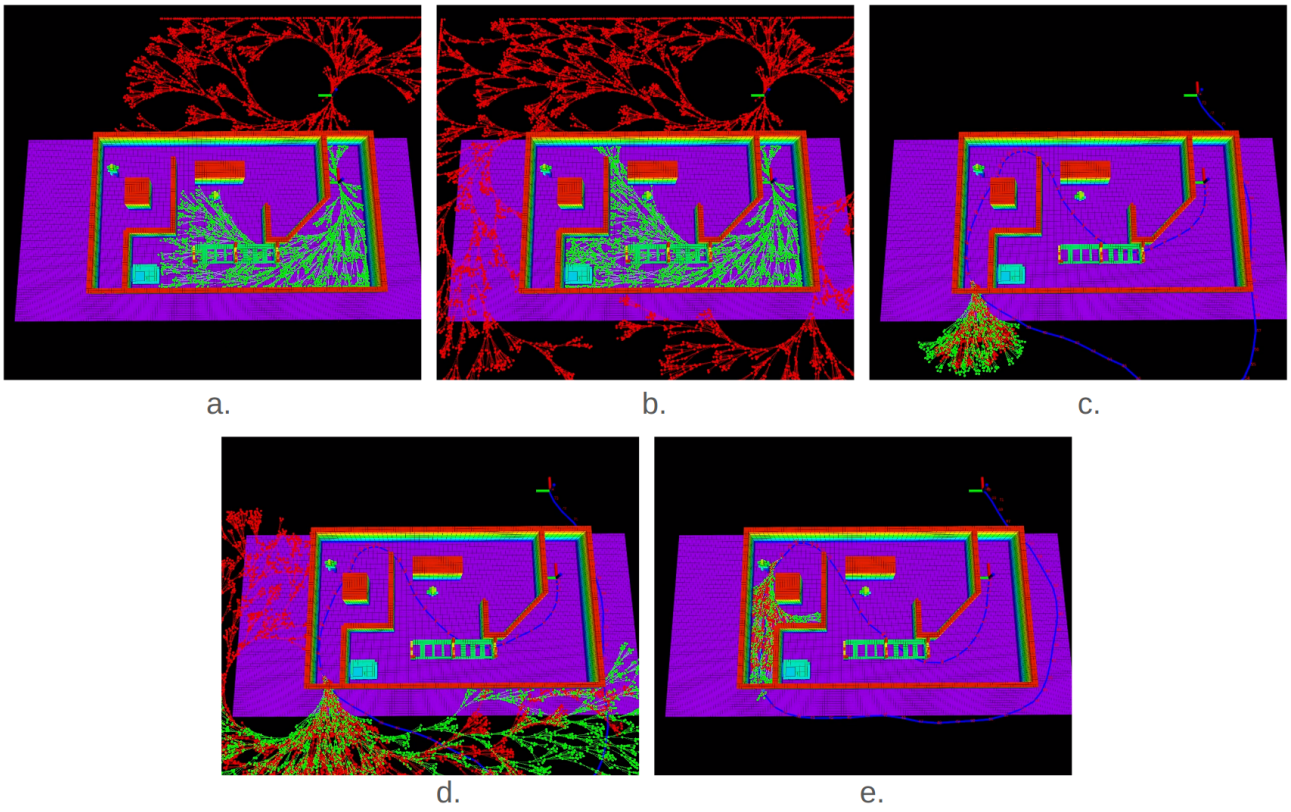


Figure 1: A series of snapshots illustrating the progression of our planner’s execution is presented. In snapshot a., we observe the initial phase where multiple trees start their bi-directional search to connect the start and goal vertices. Upon finding an initial solution, as shown in snapshot c., the focus of the trees shifts towards exploitation of the current solution. This phase involves continuous optimization of the path. Snapshot e. displays the culmination of this process, showcasing the optimized motion plan. Throughout this sequence, the planner demonstrates its capability to dynamically transition from exploration to exploitation, effectively refining the path to achieve an optimal solution.

celerations, etc.), i.e., kinodynamic planning, several variants of RRT algorithm have been proposed. Two notable examples include the RRT-Extend algorithm LaValle and James J. Kuffner (2001) and the Expansive Space Trees (EST) algorithm Hsu et al. (1997). These methods focus on the forward propagation of dynamics and aim to efficiently and evenly explore the state space, regardless of obstacles. Additionally several works extended on Karaman and Frazzoli (2011) to benefit from asymptotic optimality, e.g. Karaman and Frazzoli (2013); Perez et al. (2012); Xie et al. (2015). However, these works relied on the availability of a steering function. A steering function computes the optimal path between two states in a scenario with no obstacles. Implementing a steering function is often associated with solving a two-point boundary value problem (BVP). This problem involves solving a differential equation while adhering to specific boundary conditions. It is widely acknowledged in the field that this can be challenging, considering the complexity of some systems (e.g., a quadrotor with 15 DOF). Recently, there has been an intense effort to remove the need for the steering function in the motion planning community. In Li et al. (2014), a kinodynamic asymptotically optimal planner named as SST was proposed based on a simplified random forward search tree that does not rely on a steering function. A theoretical analysis of the proposed algorithm was provided, which showed that as long as an adequate number of Monte Carlo propagations (random controls and duration) are added to the search tree, the algorithm ensures asymptotic optimality. A meta-algorithm that produces an asymptotically optimal kinodynamic planner was presented in Hauser and Zhou (2016), given any feasible kinodynamic planner as a subroutine without using steering functions or numerical boundary-value problem solvers. The increase in the use of heuristics has also become prevalent in kinodynamic planners Littlefield and Bekris (2018). Authors in Kleinbort et al. (2018, 2019) have led to a deeper understanding of

the properties of asymptotically optimal kinodynamic planning with milder assumptions, which the proposed method also takes advantage of. Most recently, authors of Shome and Kavraki (2021) proposed a new approach for kinodynamic motion planning using sampling techniques to estimate the connectivity of high-dimensional configuration spaces. The approach uses bundles of kinodynamic edges to cover the state space before a query arrives and is shown to be asymptotically optimal and to find high-quality solutions quickly in experimental validation. Upon analyzing the existing literature in sampling-based motion planning, a noticeable trend emerges: the increasing adoption of heuristics in the most effective geometric motion planners. Yet, this trend has not been as prevalent in kinodynamic motion planning, likely due to its added complexity. Our work distinguishes itself from existing kinodynamic planners by placing a significant emphasis on the use of heuristics. This strategy is employed to rapidly explore the extensive state space. Once an initial solution is found, our approach shifts towards exploitation, which can also be construed as a heuristic-driven strategy. This dual focus on exploration followed by exploitation, underpinned by heuristic principles, sets our approach apart in kinodynamic motion planning.

3 Problem Setup

In the present work, we delve into the domain of kinodynamic motion planning, a detailed exposition of which is provided in Sec. 4. Kinodynamic motion planning is a specialized subset of motion planning that intricately incorporates the dynamics of a system into the planning process. This includes considering constraints imposed by the system's velocity and acceleration, as well as its equations of motion while charting a trajectory from an initial state to a designated goal state. The primary objective of kinodynamic planning is to identify a path that is not only feasible but also optimal, adhering to both geometric and dynamic constraints.

We assume a d-dimensional smooth manifold for the state space X. The goal region is assumed to be in the free region of state space X_F such that $X_F \subset X$, $x_{goal} \in X_F$, $\delta_{goal} > 0$ and $X_{goal} = B_{\delta_{goal}}(x_{goal})$. δ_{goal} denotes obstacle clearance and $B_{\delta_{goal}}(x_{goal})$ is ball (Euclidean) centered at $x_{goal} \in R^d$ with radius r. Since we consider a dynamic system, a control space is denoted with $U \subseteq R^D$. The considered system evolves from the current state with given control inputs in the following manner;

$$\dot{x(t)} = f(x(t), u(t)), x(t) \in X, u(t) \in U \tag{1}$$

The system is assumed to be Lipschitz continuous for both x and u such that; $\exists K_x, K_u > 0$, $\forall x_0, x_1 \in X, u_0, u_1 \in U$;

$$\begin{aligned} ||f(x_0, u_0) - f(x_0, u_1)|| &\leq K_u ||u_0 - u_1|| \\ ||f(x_0, u_0) - f(x_0, u_1)|| &\leq K_x ||x_0 - x_1|| \end{aligned} \tag{2}$$

Definition 1. A valid kinodynamic trajectory π is produced by propagating system forwards (see Eq. 1) starting from $\pi(0)$ by applying control function $\Upsilon : [0, t_\pi] \subset U$ resulting in $\pi : [0, t_\pi] \subset F$.

Definition 2. A piecewise constant control function Υ is composed of multiple constant control functions $\overline{\Upsilon_i}$, each defined over a specific interval of time $[0, \Delta t]$, where each constant control function $\overline{\Upsilon_i}$ maps to a specific control value u_i that belongs to set U. The number of constant control functions is represented by a natural number $k \in N$.

In optimal kinodynamic motion planning, the goal is to identify a control function Υ and a trajectory π that lives in subset F. The planner should also minimize the overall cost of the trajectory, which is calculated as the integral of a cost function $g(\pi(t), \Upsilon(t))$ over the entire duration of the trajectory, from time 0 to t_π. More compactly the cost of π, g is denoted as $COST(\pi)$.

4 Approach

Our algorithm is designed to identify an optimal motion plan for a specified start-goal pose, utilizing random forward propagation of the system's states. To achieve this, the algorithm accommodates input constraints of the system, such as limits on velocity and acceleration. A prerequisite for the algorithm's effective operation is the prior knowledge of the system model, which enables the calculation of subsequent system states based on the current state and control inputs. This approach often necessitates navigating extensive state spaces, presenting a significant challenge to the algorithm's time efficiency. In our novel approach, the algorithm concurrently grows 2N trees, with N trees originating from the start pose and the remaining N from the goal pose. Here, N represents the number of threads, a configurable parameter in our method. This design inherently renders our planner multi-threaded, a distinction that, to the best of our knowledge, is unique in this field of study. The rationale for allocating an equal number of trees to both the start and goal poses is to enhance the likelihood of discovering an initial solution faster. This strategy addresses the inherently stochastic nature of the process, stemming from the non-deterministic control sampling, which significantly influences the initial solution discovery. Upon finding

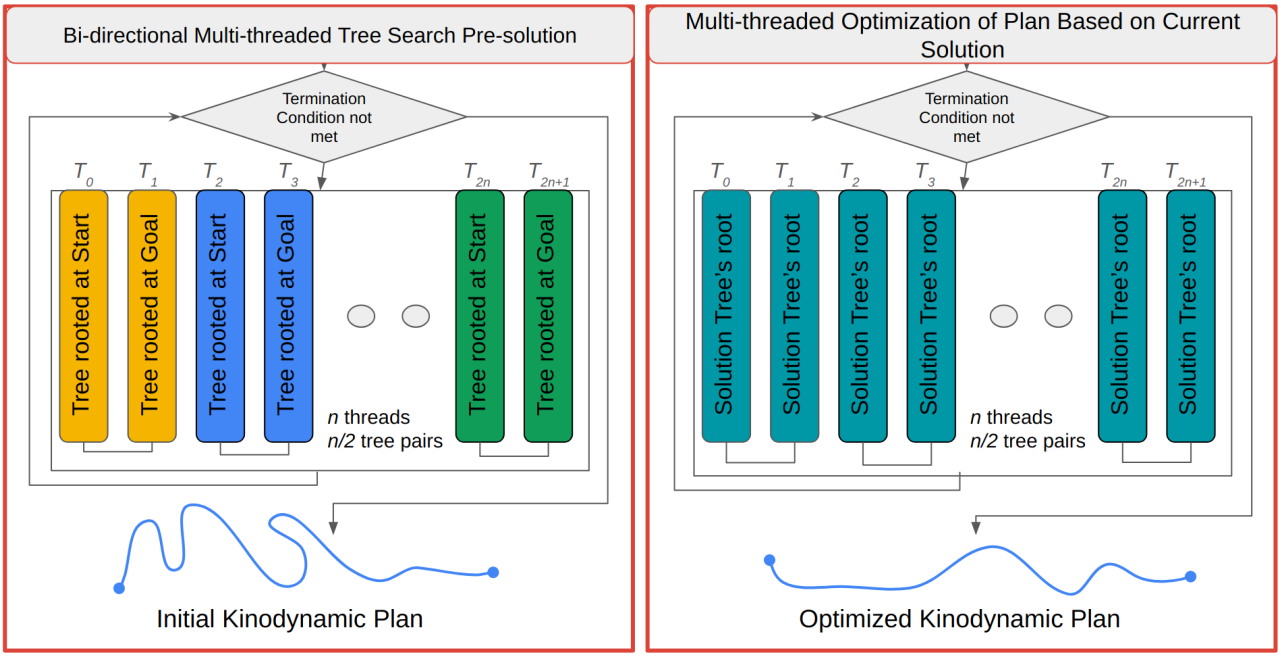


Figure 2: On the left side, we visualize the exploratory phase of the algorithm. Before finding an initial solution, the algorithm employs a predominantly explorative heuristic alongside multiple bi-directional trees. On the right side, following the discovery of an initial solution, all trees adapt to focus on the current solution. They aim to enhance the solution's cost-effectiveness, guided by the direction of the tree that first identified the initial solution.

the initial solution, the algorithm enters a phase of continuous exploitation and optimization of this solution. This optimization process persists until a predefined termination condition is satisfied. In the subsequent subsections, we will elaborate on the specific subcomponents of our planner, which we have named 'CostTrust'.

4.1 Definitions

A tree structure is initiated from a root vertex and systematically extends outward. This expansion aims to establish a linkage between the root and the designated target vertex. Subsequently, we explain the concept of a *vertex*, elaborating on its intrinsic properties and significance.

Vertex in Motion Planning: In the context of our motion planning algorithm, a vertex is conceptualized through a structure encapsulating key attributes essential to the planning process:

Our motion planning algorithm utilizes this structured vertex representation for the search space; see Tab. 1. Furthermore, these vertices are organized within a nearest-neighbor structure, facilitating efficient query processing Sucan et al. (2012).

It is crucial to note that the terms 'root-target' and 'start-goal' are not used interchangeably in our approach. Our bi-directional strategy means that for some trees, the 'target' vertex may actually be the '*start*' vertex. Similarly, the 'root' vertex could be synonymous with the 'goal' vertex in certain contexts. In any given problem setup, while the 'start' and 'goal' vertices remain consistent, the trees, depending on their direction, may treat either of these vertices as the 'root' or 'target'.

4.2 Selecting Frontier Vertexes

After initializing a tree with the root vertex, we create *branches* by applying random control samples to the root vertex's states. As the algorithm progresses, the vertex count escalates, making selecting vertices for further expansion increasingly critical. Our objective is initially to identify and select vertices most beneficial to rapid exploration. In this context, such vertices are termed 'frontiers.' Random control sampling is applied to these frontiers, a process we will present in the following subsection.

In Fig. 3, we illustrate the opposing trees utilized in our approach. The criteria for selecting frontier vertices are based on several key properties: firstly, the number of branches in a vertex n_b; secondly, the

Table 1: Vertex Attributes in Our Planner

Attribute	Description
State	The current state of the system, capturing its position, orientation, and other relevant parameters.
Control	The control input applied to the system, influencing its transition to the next state.
Cost	A numerical value representing the cumulative cost incurred to reach this vertex from the root.
Control Duration	The time period for which a specific control input is applied.
Blacklisted Status	A flag indicating whether the vertex is excluded from further exploration due to factors like infeasibility or collision.
Root Status	A designation that identifies whether the vertex serves as the starting point of a search tree.
Solution Relevance	A boolean status indicating whether the vertex is part of the path identified by the planner.
Branches	A collection of subsequent vertices originating from the current vertex, representing possible paths forward.
Parent	A reference to the preceding vertex, forming the connection in the path or search tree.

accumulative cost associated with a vertex c, calculated from the root; and thirdly, the density of vertices within its immediate neighborhood d.

The selection process is conducted in two steps. First, we calculate an initial score, denoted by s, for each vertex, based on the parameters: the number of branches (n_b) and the cumulative cost (c).

$$s = 1/(n_b + 0.001) + c \tag{3}$$

In the subsequent phase of the selection process, the vertices are arranged in descending order based on their score, denoted as s, see Eq. 3. Vertices with higher scores are typically those at the interface of explored and unexplored spaces. The second step involves considering the neighborhood density, represented as d,

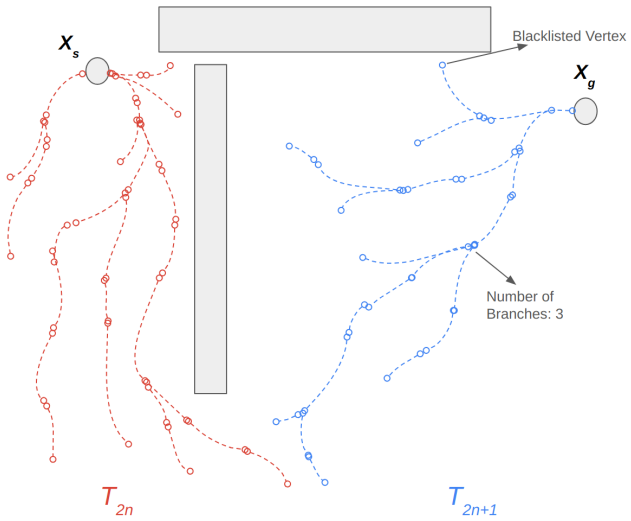


Figure 3: The presented figure depicts the bi-directional tree expansion strategy, which operates by simultaneously expanding two distinct trees rooted at the start and goal to enhance the algorithm's efficiency. The dashed paths represent the propagation of states forward with a given control input. Note that the branches of one tree do not need to be connected to the branches of the incoming tree. The aim is to connect X_s to X_g with any tree.

for a predefined subset of frontiers, identified in the initial phase. This subset is then re-sorted, taking into account the cumulative distance of each vertex to its K nearest neighbors. Following this re-sorting, vertices located in sparser neighborhoods are given higher priority. The two-step nature of the frontier selection process is fundamentally driven by considerations of efficiency. Calculating the nearest K vertices for every vertex in the tree becomes increasingly inefficient as the tree expands. Therefore, the initial step serves as an efficient preliminary process, identifying frontiers with a higher likelihood of enhancing the algorithm's exploratory capability. The subsequent step is designed to ensure efficiency by focusing on these pre-selected frontiers, thereby streamlining the overall process.

4.3 Frontier Expansion

As indicated previously, the frontiers designated for tree expansion act as a heuristic. In the ensuing phase, the algorithm expands these frontiers, employing random control inputs with random duration alongside a random number of branches. As outlined in Tab. 1, control inputs and their durations are applied to the states of these frontier vertices. Upon confirmation of

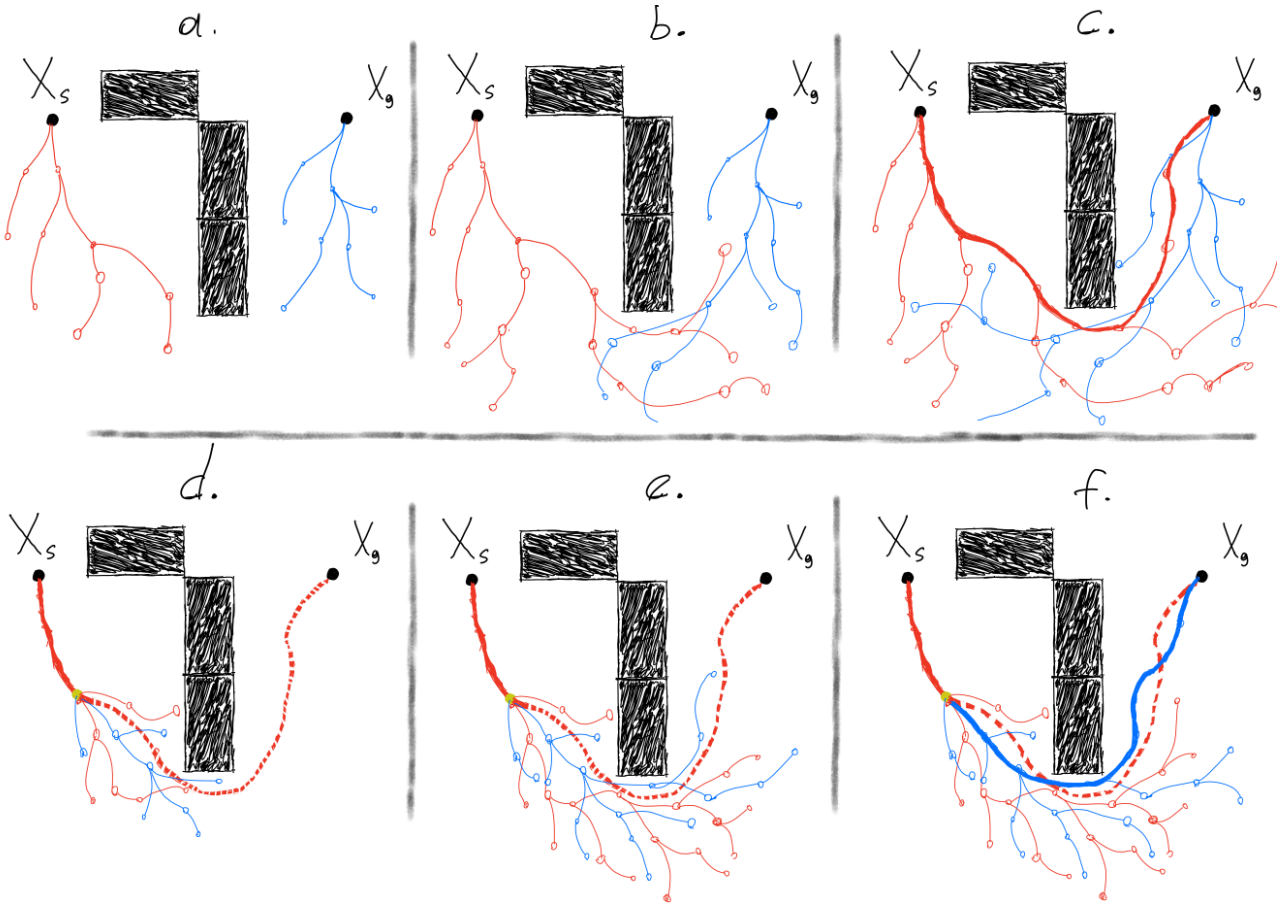


Figure 4: Initially, our algorithm initiates tree growth from both the start and goal vertices, as illustrated in panels a., b., and c. Following discovering a solution, all trees and their respective threads shift focus to exploit this initial solution. The objective is to enhance the solution, aligning with the direction established by the tree that first identified the solution. For instance, as shown in panel f., a path with a more favorable cost is achieved by strategically exploiting the current optimal solution. The solution exploitation can run indefinitely.

being collision-free, the new vertices are integrated into the tree. In instances where all branches originating from a particular frontier vertex do not progress due to a collision, such a vertex is designated as 'blacklisted.' Consequently, this status leads to its exclusion from being selected in the subsequent iteration of frontier identification.

After the successful expansion of a frontier, an additional critical evaluation is conducted. This assessment determines whether the newly added vertex can be connected to the goal at a lower cost than that of the existing solution if one exists. If this condition is met, the trees are reset, and the current best solution is updated accordingly.

4.4 Initial Solution and Plan Optimization

Prior to the discovery of an initial solution, the trees, which operate independently, undergo bi-directional growth. Each tree is rooted either at the start or the goal, as depicted in Fig. 3. The tree represented in red color originates from the start pose, denoted as X_s, while the blue tree originates from X_g, the goal pose. It is important to note that, although these trees are independent, the multi-threaded architecture of our method enables their simultaneous growth, increasing the efficiency dramatically as evidenced by findings in Sec. 5. Upon the identification of an initial solution, all trees are reinitialized to commence from the same root vertex as the tree that initially identified the solution. A detailed illustration of the planner's progression is provided in Fig. 4.

The algorithm persistently exploits the current opti-

mal solution, intending to improve its cost. This persistent refinement process underpins the asymptotic optimality of our approach. The stages of this ongoing solution exploitation are visually detailed in d., e., and f. of Fig. 4.

4.5 Algorithms

In this subsection, we delve into the core algorithms of our approach, accompanied by a detailed discussion of the corresponding pseudocode.

Algorithm 1 Solve Planning Problem

Require: *ptc* {Planner Termination Condition}
Ensure: PlannerStatus $\in$ {EXACT, APPROXIMATE, INVALID}
Initialize *start_state* and *goal_state*
if isValid(*start_state*) = FALSE $\vee$ isValid(*goal_state*) = FALSE **then**
  return INVALID
end if
for $t = 1$ to numThreads **do**
  direction $\leftarrow$ $t\%2 = 0$? *start_to_goal* : *goal_to_start*
  CALL **Algorithm 2**
  CALL **Algorithm 3**
end for
Wait for all threads to complete
bestPath $\leftarrow$ findBestPath()
if *bestPath* $\neq$ NULL **then**
  Update *bestControlPath* and *controlTrees*
  VisualizePaths
  PlannerStatus $\leftarrow$ EXACT or APPROXIMATE
end if
Clear data structures
return PlannerStatus

In the proposed multi-threaded control planning framework, three core algorithms interact to refine the solution path in a kinodynamic planning problem iteratively. The primary algorithm, **Solve Planning Problem** (Algorithm 1), orchestrates the overall planning process. It initializes the planning scenario, including the start and goal states, and then iteratively invokes parallel threads. Each thread independently executes a cycle of explorative frontier selection and expansion, managed by Algorithms 2 and 3, respectively.

Select Explorative Frontiers (Algorithm 2) plays a critical role in guiding the search process. Within each thread cycle, this algorithm selects a subset of frontier nodes from the rapidly-exploring tree. The selection is based on a scoring mechanism that considers both the cost associated with each node and the num-

Algorithm 2 Select Explorative Frontiers

Require: *max_number*, *nn_structure*
Ensure: Selected set of frontier nodes
frontier_nodes $\leftarrow$ listNodes(*nn_structure*)
Remove blacklisted nodes from *frontier_nodes*
for each *node* $\in$ *frontier_nodes* **do**
  score[*node*] $\leftarrow$ computeScore(*node*)
end for
Sort *frontier_nodes* based on *score*
frontier_nodes $\leftarrow$ clip(*frontier_nodes*, *max_number*)
for each *node* $\in$ *frontier_nodes* **do**
  density[*node*] $\leftarrow$ computeDensity(*node*)
end for
Sort *frontier_nodes* based on *density*
frontier_nodes $\leftarrow$ clip(*frontier_nodes*, *max_number*)

Algorithm 3 Expand Frontiers

Require: *frontier_nodes*, *num_branch_to_extend*, *nn_structure*, *ptc*, *target_property*, *path*, *control_paths_vertices*, *exact_solution_found*, *should_stop_exploration*, *current_best_path*
Ensure: Expanded set of frontier nodes
for each *node* $\in$ *frontier_nodes* **do**
  for $i = 1$ to *num_branch_to_extend* **do**
    if *ptc* = TRUE **then**
      return
    end if
    (*new_state*, *valid*) $\leftarrow$ propagate(*node*, randomControl())
    if *valid* **then**
      vertex $\leftarrow$ createVertex(*new_state*)
      vertex.cost $\leftarrow$ computeCost(*node*, *new_state*)
      if isCloseToGoal(*vertex*, *target_property*) **then**
        path $\leftarrow$ updatePath(*vertex*)
      end if
      Add *vertex* to *nn_structure*
    end if
  end for
end for
EvaluateBranches(*frontier_nodes*, *current_best_path*, *should_stop_exploration*)

ber of existing branches, thereby balancing exploration and exploitation. The algorithm prioritizes nodes that present a beneficial trade-off between these two factors, ensuring an efficient yet thorough exploration of the search space.

Expand Frontiers (Algorithm 3) is then called to expand the selected frontiers. This algorithm generates new states by applying random controls to the chosen frontier nodes, effectively growing the search tree. It ensures that new states are valid and checks for their proximity to the goal, updating the solution path whenever a more cost-effective route is discovered. Extending frontiers is pivotal in exploring unseen areas of the state space and getting closer to an optimal solution.

Together, these algorithms enable a dynamic and adaptive approach to kinodynamic planning. By utilizing multiple threads, the **Solve Planning Problem** algorithm ensures diverse search directions and a robust state space exploration. The interplay between **Select Explorative Frontiers** and **Expand Frontiers** algorithms allows for a guided yet flexible search strategy, continually adapting to the evolving topology of the tree. This synergy effectively balances exploration and exploitation, significantly enhancing the planner's ability to find optimal or near-optimal solutions for systems with complex kinodynamics.

5 Experimental Results

Our evaluation methodology encompassed two distinct dynamic systems: an Ackermann-steered vehicle-like robot and a dynamic quadrotor with 15 state dimensions. Both systems included state and control components, with the controls for each system being sampled within a range specified by the user. To assess the efficacy of our proposed method in addressing kinodynamic planning problems, we employed two maps featuring a variety of objects. The map illustrated in Fig. 2, for instance, spans an area of 18x6 meters.

Our study included all prominent kinodynamic planners available in the Open Motion Planning Library (OMPL) to ensure a comprehensive evaluation. These planners were compared against our proposed planner. The ensuing sections detail the results, generated on a laptop equipped with an Intel Core i7-6700HQ CPU, operating at 2600 MHz with eight cores, and supplemented by 16GB of memory.

Our primary metric for evaluation is the resulting path cost, quantified by the accumulative control duration of the solutions generated by the planners. The exact solution rate is also presented as a metric in the benchmarks. Each planner is allowed an identical time frame to produce its results. Given the larger state and control dimensions of the Quadrotor compared to the Ackerman robot, a different amount of time allocation is employed for these two benchmarks. Specifically, planners are provided 10 seconds for the Ackerman robot and 40 seconds for the Quadrotor.

A majority of the baseline planners, including our own, fall under the category of *optimizing* planners. This implies that these planners are designed to utilize the entirety of their allotted time to enhance their solutions, provided they achieve a solution within this timeframe. To ensure a fair comparison, all planners are assigned a fixed time duration for operation. However, due to the significantly larger state space encountered in dynamic quadrotor planning, these planners are allocated four times the amount of time given for planning with the Ackermann-steered robot. This adjustment acknowledges the increased complexity and computational demands of the quadrotor's state space.

5.1 Ackermann-steered vehicle-like robot

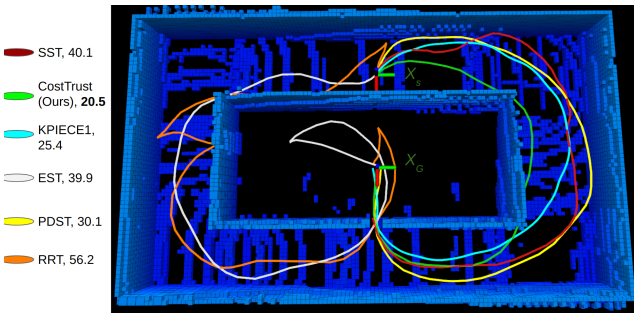


Figure 5: The figure illustrates a sample from our benchmark results for kinodynamic motion planning of an Ackermann-steered robot. The motion plans generated by the various contestant planners are depicted using distinct colors for clarity. Alongside each planner's name, the corresponding path costs are enumerated. Notably, our planner, CostTrust, achieves the lowest cost path, which is marked with a green color.

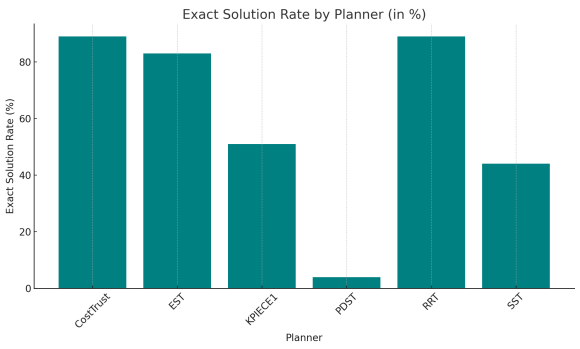
The Ackermann steering system, often used in vehicle-like robots, is characterized by a unique geometry that allows the front wheels to turn at different angles. This system is ideal for ensuring that all wheels follow their respective circular paths during a turn, minimizing slippage. The typical equations governing an Ackermann-steered vehicle are:

$$
\begin{aligned}
x_{\text{next}} &= x + \Delta t \cdot v \cdot \cos(\psi + \beta) \\
y_{\text{next}} &= y + \Delta t \cdot v \cdot \sin(\psi + \beta) \\
\psi_{\text{next}} &= \psi + \Delta t \cdot \frac{v}{L_b} \cdot \sin(\beta) \\
v_{\text{next}} &= v + \Delta t \cdot a \\
\beta &= \arctan\left(\frac{L_a}{L_a + L_b} \cdot \tan(d_f)\right)
\end{aligned}
\tag{4}
$$

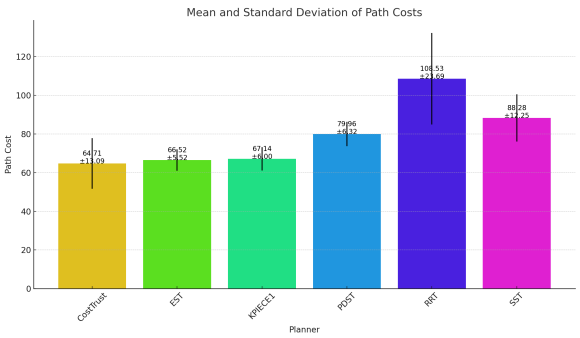
In these equations: Δt is the time step. x, y, ψ, v are the current state variables representing position, orientation, and velocity. d_f is the front wheel steering angle. a is the acceleration. L_a, L_b are distances from the vehicle's center of gravity to the front and rear axles, respectively. β is the slip angle, which is a function of the steering angle d_f and the vehicle's geometric parameters L_a, L_b.

The forward propagation of a vehicle-like robot system states is described in Eq. 4. The system's state vector, $X_c = [x, y, \psi, v]$, is subjected to a randomly sampled control vector, $U_c = [a,\ d_f]$, for a randomly selected duration in the interval $[0, T_{max\_prop}]$, resulting in the system transitioning to next state X_n. This process is repeated to expand the tree as outlined in Subsec. 4.5. The environment used for this use-case is a house-like setting, as depicted in Fig. 1.

The start and goal states are placed such that the algorithm is forced to explore the state space since the only connection point where the planners can connect start-goal states is a gate at the bottom right of the map. For the start-goal states depicted in Fig. 1 with 3D axes, a benchmark is run 100 times. The results, presented in Fig. 6, indicate that the proposed planner's solve rate for 100 problems is comparable to that of RRT*, both showing a solve rate of 89 % while outperforming all other baselines. However, the proposed method's efficiency lowers the resulting path cost within the given time, depicted in the bottom image of Fig. 6. Regarding the solve rate, the proposed planner failed to find a solution in 11 cases. In rare instances, the planner may struggle to establish a connection as the number of nodes in the tree increases, leading to longer execution times for each iteration. We believe the cause of the missed cases could be attributed to the dense tree structure utilized in the proposed planner. Specifically, maintaining a dense tree structure may increase the likelihood of prolonged exploration of some narrow passages, potentially increasing the number of missed cases. To address this issue, we have modified the implementation of our approach such that the planner can be configured to maintain a sparse tree structure similar to that of SST, which may reduce the occurrence of missed cases.



(a)



(b)

Figure 6: In (a), we present the exact solve rate percentages for kinodynamic motion planning of an Ackermann-steered vehicle-like robot. In (b), we focus on the resulting path costs, taking into account only those paths derived from exact solutions. Our planner, CostTrust, demonstrates superior performance over baseline planners in terms of path cost efficiency. It achieves a comparable solve rate to RRT*, registering an 89% success rate in finding solutions while outperforming the rest of the baselines.

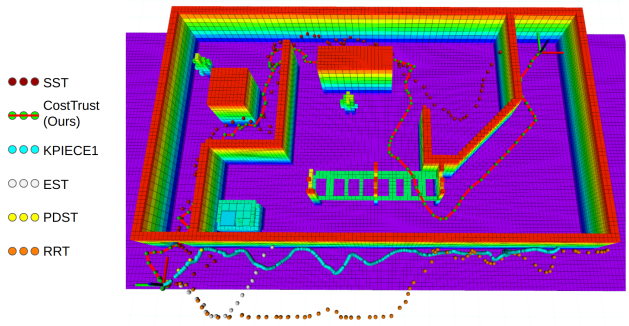


Figure 7: The figure showcases a representative sample focusing on kinodynamic motion planning for a quadrotor. Notably, as depicted in the figure, CostTrust surpasses existing methods in its performance. It stands out as the sole approach that consistently finds the *exact* solution. In contrast, the other baseline methods predominantly yield infeasible or *approximate* solutions.

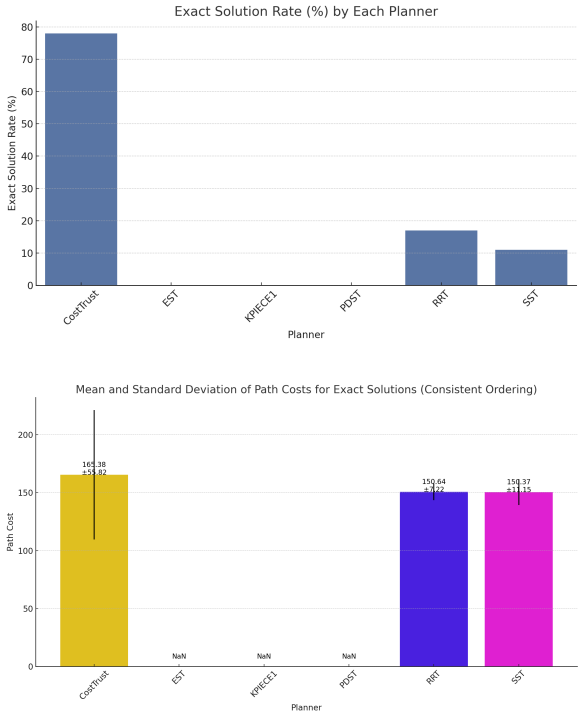


Figure 8: In (a), we present the exact solve rate percentages, and in (b), resulting path costs for kinodynamic motion planning of a quadrotor. Our planner, CostTrust, surpasses other planners significantly in terms of the number of exact solutions achieved.

5.2 Quadrotor

Sampling-based motion planning demonstrates marked effectiveness in higher-dimensional state spaces, where discretization-based methods become computationally infeasible due to exponential computation cost escalation. To showcase the efficacy of our kinodynamic motion planner, we extended our benchmarking to include kinodynamic planning for a quadrotor characterized by 15 state dimensions and six control dimensions. Control sampling for the quadrotor was confined within a user-defined range, conforming to the design limitations of quadrotors. The forward propagation model employed for the quadrotor is detailed in Eq. 5.

In Eq. 5; The system's state vector comprises 15 elements. The positions are denoted $[x, y, z]$. T is the thrust. m is the mass. g is the gravitational acceleration. a_x, a_y, a_z are the accelerations in respective axes. $K_p, K_d, K_\phi, K_\theta, K_\psi$ are proportional gains for z, roll, pitch, and yaw. z_d, ψ_d are the desired z position and yaw angle. z, ϕ, θ, ψ are the current z position and orientation angles. R is the rotation matrix. $\vec{a}$ is the acceleration vector. $\ddot{\vec{r}}, \dot{\vec{r}}, \vec{r}$ are the acceleration, velocity, and position vectors. Δt is the time step. In comparison, the control vector comprises six elements denoted by $[z_d, \dot{z}_d, \psi_d, a_x, a_y, a_z]$. These control inputs correspond to the desired z position, z velocity, yaw angle, and x, y, z accelerations, respectively. As the system's complexity increases, the performance disparity between our proposed planner and other planners becomes more pronounced, as illustrated in Fig. 8.

$$
\begin{aligned}
T &= m \cdot (g + a_z + K_p \cdot (z_d - z) + K_d \cdot (\dot{z}_d - \dot{z})), \\
\tau_\phi &= K_\phi \cdot \left(\frac{(a_x \cdot \sin(\psi) - a_y \cdot \cos(\psi))}{g} - \phi \right), \\
\tau_\theta &= K_\theta \cdot \left(\frac{(a_x \cdot \cos(\psi) - a_y \cdot \sin(\psi))}{g} - \theta \right), \\
\tau_\psi &= K_\psi \cdot (\psi_d - \psi), \\
\dot{\phi} &+= \tau_\phi \cdot \Delta t, \\
\dot{\theta} &+= \tau_\theta \cdot \Delta t, \\
\dot{\psi} &+= \tau_\psi \cdot \Delta t, \\
R &= \text{RotationMatrix}(\phi, \theta, \psi), \\
\vec{a} &= \left(R \cdot \begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ m \cdot g \end{bmatrix} \right) / m, \\
\ddot{\vec{r}} &= \vec{a}, \\
\dot{\vec{r}} &+= \ddot{\vec{r}} \cdot \Delta t, \\
\vec{r} &+= \dot{\vec{r}} \cdot \Delta t
\end{aligned}
\tag{5}
$$

The results unequivocally indicate that our pro-

posed planner achieves a considerably higher solve rate compared to baseline planners. In this analysis, we exclusively consider exact solutions, disregarding any approximate results produced by the planners. Specifically, the solve rate achieved by our planner is 78%, markedly outperforming the next best-performing planner, RRT*, which registers a solve rate of 17%. The resulting path costs for all the planners that are able to provide an exact solution seem reasonably similar. Based on these findings, we draw that our planner is particularly advantageous for high-dimensional systems requiring extensive sampling and rapid state space exploration to obtain a solution. For example, as illustrated in Fig. 8, existing planners except RRT* and SST cannot navigate the sole passage (lower left) to the goal. As the number of samples increases, it becomes increasingly time-consuming to identify the passage due to the growing number of nodes in the tree structure. Our method's bi-directional approach offers benefits in navigating sharp turns, preventing tree growth from becoming trapped, and prolonging exploration. The proposed planner achieves a threefold improvement in mean trajectory cost compared to the next best-performing planner, SST.

5.3 Discussion

The results distinctly demonstrate the advantages of our newly introduced motion planner over current state-of-the-art techniques, particularly in scenarios involving a dynamic quadrotor characterized by high-dimensional state and control spaces. The effectiveness of our proposed planner can be attributed to several pivotal characteristics:

Dynamic Balance between Exploration and Exploitation: The planner adeptly navigates between exploring new areas of the state space and exploiting known paths. This is achieved through the strategic selection of frontier vertices within the tree, which are instrumental in guiding the exploration process, coupled with the continuous refinement of existing solutions.

Multi-threaded Bi-directional Tree Approach: A significant strength of our planner is its implementation of multi-threaded, bi-directional trees. This architecture has proven particularly beneficial in enhancing the solution rate for the dynamic quadrotor, outperforming traditional single-threaded or uni-directional approaches in terms of both efficiency and effectiveness.

These attributes underscore our planner's robustness and adaptability in complex motion planning scenarios. The combination of an intelligent exploration-exploitation balance and the innovative use of multi-threaded bi-directional trees not only facilitates a higher solution rate but also contributes to more nuanced and effective navigation through challenging state spaces.

In comparing the benchmark results for the two systems, namely the Ackermann-steered robot and the dynamic quadrotor, a notable observation emerges. Despite allocating four times more time for the quadrotor, the solution rate for all planners, except ours, significantly diminishes. This phenomenon underscores the non-linear correlation between the state dimension complexity and the time required to find a solution. The exceptional performance of our planner can be attributed to its inherent exploratory instinct and the efficiency of its multi-threaded, bi-directional tree search approach. These characteristics are pivotal in effectively navigating the complex state space of the quadrotor, thereby facilitating a higher solution rate compared to other planners.

While our proposed motion planner exhibits notable strengths, it is not without limitations, as reflected in the solution rate results for both the Ackerman robot and the quadrotor. In certain cases, the planner fails to derive an exact solution. This limitation is suspected to stem from the planner's inherent tendency towards extensive exploration. Before identifying an initial solution, the planner prefers areas with fewer branches and lower vertex density. Consequently, in scenarios where numerous vertices are proximal to the goal yet remain unconnected to the tree, the planner may inadvertently prioritize frontiers distant from the goal region. Over time, this reduces the likelihood of establishing a connection to the goal, as vertices near the goal are less frequently selected as frontiers.

However, this limitation could be mitigated by implementing goal biasing. Such an approach would involve incorporating a proportion of the nearest neighbors of the goal at each iteration of frontier selection. This strategy could enhance the planner's ability to identify and prioritize goal-proximate vertices, thereby improving the likelihood of establishing a connection to the goal.

We have made the source code for our proposed motion planner, as well as the benchmarking code, publicly available as open-source resources. This initiative aims to facilitate further experimentation by researchers and allows for the comparison of newly developed planners against our proposed method. From a practical standpoint, users intending to deploy our motion planner (or other baseline methods) must possess an understanding of the system model. Crucially, they need to define a *propagate* function, which estimates the next state of the system based on current state parameters and control inputs. To assist users,

we provide exemplar codes for two system models: an Ackermann-steered robot and a quadrotor. Additionally, for effective utilization of the sampling-based motion planner, a collision-check function is required to determine whether a given state is in a collision or is collision-free.

6 Conclusion

This paper introduces a novel approach to kinodynamic planning, showcasing enhanced performance compared to existing state-of-the-art methods, particularly in high-dimensional contexts such as dynamic quadrotors. Distinctive in its strategy, our method employs a balanced interplay between exploration and exploitation, complemented by a multi-threaded, bi-directional tree search process. This dual-pronged approach yields significant performance improvements, as evidenced by our results in two benchmarks. The potential of our method to address real-world problems in autonomous robotics is evident from the results. In certain instances, our planner may not yield an exact solution, a limitation primarily attributed to its devotion to fast exploration of unknown spaces initially. To address this issue, future iterations of our planner will explore the incorporation of goal-biasing sampling. This enhancement would adjust the frontier selection criteria to balance the exploration with a more deliberate effort to connect to the goal in each iteration. Such a modification aims to refine the planner's effectiveness, ensuring a more targeted approach towards achieving the desired goal state.

References

Gammell, J. D., Barfoot, T. D., and Srinivasa, S. S. Batch informed trees (bit*): Informed asymptotically optimal anytime search. *The International Journal of Robotics Research*, 2020. 39(5):543–567. URL https://doi.org/10.1177/0278364919890396, doi:10.1177/0278364919890396.

Hauser, K. and Zhou, Y. Asymptotically optimal planning by feasible kinodynamic planning in a state–cost space. *IEEE Transactions on Robotics*, 2016. 32(6):1431–1443. doi:10.1109/TRO.2016.2602363.

Hsu, D., Latombe, J.-C., and Motwani, R. Path planning in expansive configuration spaces. In *Proceedings of International Conference on Robotics and Automation*, volume 3. pages 2719–2726 vol.3, 1997. doi:10.1109/ROBOT.1997.619371.

Karaman, S. and Frazzoli, E. Sampling-based algorithms for optimal motion planning. *CoRR*, 2011. abs/1105.1186. URL http://arxiv.org/abs/1105.1186.

Karaman, S. and Frazzoli, E. Sampling-based optimal motion planning for non-holonomic dynamical systems. In *2013 IEEE International Conference on Robotics and Automation*. pages 5041–5047, 2013. doi:10.1109/ICRA.2013.6631297.

Kavraki, L., Svestka, P., Latombe, J.-C., and Overmars, M. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 1996. 12(4):566–580. doi:10.1109/70.508439.

Kleinbort, M., Solovey, K., Bonalli, R., Bekris, K. E., and Halperin, D. RRT2.0 for fast and optimal kinodynamic sampling-based motion planning. *CoRR*, 2019. abs/1909.05569. URL http://arxiv.org/abs/1909.05569.

Kleinbort, M., Solovey, K., Littlefield, Z., Bekris, K., and Halperin, D. Probabilistic completeness of rrt for geometric and kinodynamic planning with forward propagation. *IEEE Robotics and Automation Letters*, 2018. PP:1–1. doi:10.1109/LRA.2018.2888947.

Lavalle, S. and Kuffner, J. Rapidly-exploring random trees: Progress and prospects. *Algorithmic and computational robotics: New directions*, 2000.

LaValle, S. M. and James J. Kuffner, J. Randomized kinodynamic planning. *The International Journal of Robotics Research*, 2001. 20(5):378–400. URL https://doi.org/10.1177/02783640122067453, doi:10.1177/02783640122067453.

Li, Y., Littlefield, Z., and Bekris, K. E. Asymptotically optimal sampling-based kinodynamic planning. *CoRR*, 2014. abs/1407.2896. URL http://arxiv.org/abs/1407.2896.

Littlefield, Z. and Bekris, K. E. Efficient and asymptotically optimal kinodynamic motion planning via dominance-informed regions. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. pages 1–9, 2018. doi:10.1109/IROS.2018.8593672.

Luna, R., Şucan, I. A., Moll, M., and Kavraki, L. E. Anytime solution optimization for sampling-based motion planning. In *2013 IEEE International Conference on Robotics and Automation*. pages 5068–5074, 2013. doi:10.1109/ICRA.2013.6631301.

Otte, M. and Correll, N. C-forest: Parallel shortest path planning with superlinear speedup. *IEEE Transactions on Robotics*, 2013. 29(3):798–806. doi:10.1109/TRO.2013.2240176.

Perez, A., Platt, R., Konidaris, G., Kaelbling, L., and Lozano-Perez, T. Lqr-rrt*: Optimal sampling-based motion planning with automatically derived extension heuristics. In *2012 IEEE International Conference on Robotics and Automation*. pages 2537–2542, 2012. doi:10.1109/ICRA.2012.6225177.

Shome, R. and Kavraki, L. E. Asymptotically optimal kinodynamic planning using bundles of edges. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*. pages 9988–9994, 2021. doi:10.1109/ICRA48506.2021.9560836.

Strub, M. P. and Gammell, J. D. Adaptively Informed Trees (AIT*): Fast asymptotically optimal path planning through adaptive heuristics. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. pages 3191–3198, 2020a. doi:10.1109/ICRA40945.2020.9197338.

Strub, M. P. and Gammell, J. D. Advanced bit* (abit*): Sampling-based planning with advanced graph-search techniques. *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020b. URL http://dx.doi.org/10.1109/ICRA40945.2020.9196580, doi:10.1109/icra40945.2020.9196580.

Sucan, I. A., Moll, M., and Kavraki, L. E. The open motion planning library. *IEEE Robotics Automation Magazine*, 2012. 19(4):72–82. doi:10.1109/MRA.2012.2205651.

Xie, C., van den Berg, J., Patil, S., and Abbeel, P. Toward asymptotically optimal motion planning for kinodynamic systems using a two-point boundary value problem solver. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*. pages 4187–4194, 2015. doi:10.1109/ICRA.2015.7139776.