# Guiding Sampling-Based Tree Search for Motion Planning with Dynamics via Probabilistic Roadmap Abstractions

Duong Le and Erion Plaku

***Abstract*— This paper focuses on motion-planning problems for high-dimensional mobile robots with nonlinear dynamics operating in complex environments. It is motivated by a recent framework that combines sampling-based motion planning in the state space with discrete search over a workspace decomposition. Building on this line of work, the premise of this paper is that the computational efficiency can be significantly improved by tightly coupling sampling-based motion planning with probabilistic roadmap abstractions instead of workspace decompositions. Probabilistic roadmap abstractions are constructed over a low-dimensional configuration space obtained by considering relaxed and simplified representations of the robot model and its feasible motions. By capturing the connectivity of the free configuration space, roadmap abstractions provide the framework with promising suggestions of how to effectively expand the sampling-based search in the full state space. Experiments with high-dimensional robot models, nonlinear dynamics, and nonholonomic constraints show significant computational speedups over related work.**

## I. Introduction

In motion planning, the objective is to plan motions that enable a robot to move from an initial state to a goal region while avoiding collisions with obstacles. While initially considered as a geometric problem [1], motion planning for complex robotic systems requires taking into account constraints imposed by the underlying dynamics in order to facilitate execution of the planned motions.

Robot dynamics describe how the robot moves as a result of applying external control inputs. As dynamics constrain the feasible motions, for example, by bounding the velocity, turning radius, directions of motions, it is difficult to find controls that result in both *feasible* motions that satisfy the physical constraints imposed by the dynamics and *collision-free* motions that drive the robot to the goal. The differential equations $\dot{s} = f(s, u)$, which model the robot dynamics, are often high dimensional and nonlinear. Additional challenges arise when incorporating more accurate simulations, such as those obtained by physics-based engines, e.g., Gazebo, Bullet [2], ODE [3], which model general rigid body dynamics.

To effectively plan collision-free and dynamically-feasible motions for mobile robots with complex dynamics, this paper builds on sampling-based motion planning, which has shown promise in solving challenging problems [4], [5]. To take dynamics into account, sampling-based approaches typically expand a motion tree, starting from the initial state, by adding new collision-free and dynamically-feasible trajectories as tree branches. RRT [6], [7] and other successful approaches have used nearest neighbors, probability distributions, reachable sets, transition costs, principal-component analysis, flood-filling behaviors, machine learning, and other techniques to guide the motion-tree expansion [4].

Another line of sampling-based motion planners has focused on using decompositions to guide the tree expansion. PDST [8] relies on subdivisions of a low-dimensional projection space to determine regions from which to expand the motion tree. IST [9] extends PDST by using roadmaps to compute heuristic costs for the cells in the subdivision. KPIECE [10] imposes a grid on a low-dimension projection and builds upon the notion of interior-exterior cells to guide the tree expansion. fRRT [11] conducts an A* search on a grid-based decomposition of the configuration space to bias the sampling in RRT to grid cells with low cost.

Syclop [12] introduced a framework that tightly couples sampling-based motion planning with discrete search over a workspace decomposition. The work in [13] considered different low-dimensional projections to employ within the Syclop framework. Further improvements to Syclop to minimize its dependence on long guides were offered in [14], which uses heuristic costs and partitions of the motion tree into equivalent groups to guide the tree expansion.

Building on our prior work in the Syclop framework, this paper shows that, when dealing with high-dimensional motion-planning problems with complex dynamics, computational efficiency can be significantly improved by tightly coupling sampling-based motion planning with probabilistic roadmap abstractions instead of workspace decompositions. This is motivated by the success of PRM [15]–[17] in solving challenging path-planning problems by using sampling to construct a roadmap that captures the global connectivity of the underlying collision-free space (see [4], [5] for discussions on PRM). However, PRM approaches have had limited applicability in motion planning with dynamics since each roadmap edge $(s_i, s_j)$ requires connecting the state $s_i$ to $s_j$ via a feasible trajectory that satisfies differential constraints imposed by dynamics. Exact solutions to this two-point boundary value problem are available only in limited cases, while numerical solutions impose significant computational cost, rendering roadmap construction impractical [18], [19].

To take advantage of the global exploration properties of PRM while avoiding issues related to two-point boundary value problems, the proposed approach constructs the roadmap abstractions over a low-dimensional configuration space obtained by considering simplified representations of the robot model and its feasible motions. In this way, the roadmap abstractions are not used to solve motion planning

The authors are with the Dept. of Electrical Engineering and Computer Science, Catholic University of America, Washington DC 20064.

with dynamics but rather to solve a simplified problem that can serve as a guide. By capturing the connectivity of the free configuration space, the roadmap abstractions provide the framework with more reliable suggestions than those obtained by workspace decompositions of how to effectively expand the motion tree toward the goal. In particular, the proposed approach uses the roadmap abstractions to induce a partition of the state space into equivalence classes. Heuristic costs based on the shortest-path distance along the edges of the roadmap abstraction are used to determine the feasibility of expanding the motion tree from the equivalence classes associated with the roadmap configurations. Heuristic costs are combined with selection penalties to provide the framework with the flexibility to discover new regions from which to expand the motion tree. In this way, the framework seeks to balance exploitation of the roadmap abstractions to greedily expand the search toward the goal with methodical exploration to discover alternative routes to the goal.

Experiments with models of a high-dimensional snake-like robot, a physics-based ground vehicle, and an aerial vehicle operating in complex environments demonstrate the efficiency of the proposed approach. Results show significant computational speedups when using roadmap abstractions instead of workspace decompositions as well as when comparing the proposed approach to other successful planners, e.g., RRT [6], [7], fRRT [11], PDST [8], IST [9], KPIECE [10].

## II. Problem Formulation

From a sampling-based motion-planning perspective, the robot's motions are encapsulated by a function

$$s_{new} \leftarrow \text{MOTION}(s, u, dt),$$

where the new state $s_{new}$ is obtained by applying the input control $u$ for one time step $dt$ starting at $s$. When dynamics are described as differential equations $\dot{s} = f(s, u)$, MOTION is implemented using numerical integration. This work also incorporates physics-based engines, which implement MOTION based on general rigid-body dynamics that take into account the physical interactions of the robot with the environment.

A function $\text{VALID} : S \rightarrow \{\texttt{true}, \texttt{false}\}$ determines if a state $s \in S$ is valid, e.g., no collisions, all values are within desired bounds. A function $\text{GOAL} : S \rightarrow \{\texttt{true}, \texttt{false}\}$ determines if $s \in S$ satisfies the motion-planning goal.

The objective is then to compute a control function $\bar{u} : [0, T] \rightarrow U$ such that the trajectory $\zeta : [0, T] \rightarrow$ obtained by applying $\bar{u}$ starting from the initial state $s_{init} \in S$ is collision-free and satisfies the motion-planning goal.

## III. Method

The approach uses roadmap abstractions in a low-dimensional configuration space to guide the expansion of a motion tree in the state space. Pseudocode is provided in Algorithm 1. The main steps are described below.

### A. Roadmap Abstraction

*1) Simplified Problem Representation:* The approach starts with a simplified representation of the overall motion-planning problem, which is obtained by considering planning in a related low-dimensional configuration space $C$ rather than the full state space $S$. The representation in $C$ could correspond to simplified geometric models as well as less constrained motions where the robot is free to translate and rotate without taking into account differential constraints imposed by dynamics. Section IV-B provides some examples.

**Algorithm 1** Pseudocode for the proposed approach

```
1:  C ← simplified problem in configuration space
2:  RM = (V_RM, E_RM) ← ConstructRoadmap(C)
3:  (hcost(c_1), ..., hcost(c_n)) ← HeuristicCosts(RM)
4:  [T, groups_T] ← CreateTree(s_init)
5:  while TIME < t_max and SOLVED = false do
6:      group_{T,c} ← SelectGroup(groups_T)
7:      s ← SelectState(group_{T,c})
8:      for several steps do
9:          u ← CONTROLLER(s)
10:         s_new ← MOTION(s, u, dt)
11:         if VALID(s) = true then break for loop
12:         AddVertex(T, s_new, u, dt)
13:         if GOAL(s_new) = true then
14:             return TRAJ(T, s_new)
15:         c ← NearestCfgRM(s_new)
16:         if (group_{T,c} ← FIND(groups_T, c)) = null then
17:             group_{T,c} ← NewGroup(c)
18:             ADD(groups_T, group_{T,c})
19:         ADD(group_{T,c}, s_new); s ← s_new
```

*2) Roadmap Construction:* The approach continues by building a roadmap that aims to capture the connectivity of the collision-free configuration space $C_{free}$ (Algorithm 1:2). As in PRM [15], the roadmap is constructed by sampling collision-free configurations and connecting neighboring configurations with collision-free paths. The collision-free configurations and paths are represented as vertices and edges in the roadmap graph $RM = (V_{RM}, E_{RM})$.

Each collision-free configuration is generated by uniform sampling, which samples a number of configurations at random in $C$ and adds to the roadmap only those configurations that are not in collision. Roadmap edges are obtained by attempting to connect each roadmap configuration to several of its nearest neighbors according to a distance metric $\rho : C \times C \rightarrow \mathbb{R}^{\geq 0}$. Good distance metrics over $C$ are generally easier to define than over $S$ since the robot motions are less constrained in $C$ [20]. Since $C$ is low dimensional, the computation of nearest neighbors does not present a bottleneck as efficient data structures that work with any distance metric are available [21]. A path between two neighboring configurations $c_i$ and $c_j$ is obtained by an appropriate interpolation in $C$. Subdivision or incremental collision checking is then used to determine whether the path avoids collisions in which case it is added to the roadmap.

While in PRM roadmap cycles are generally avoided to save computation time, as discussed in section III-B, they are needed in the proposed approach to improve the heuristic costs to the goal. Hence, path collision checking is performed

even when $c_i$ and $c_j$ belong to the same roadmap component. Since $C$ is low dimensional, the increase in computation time due to cycles is negligible. In fact, in the experiments in this paper, the roadmap construction took less than a second.

In order to ensure that the start and goal configurations belong to the same roadmap component, additional sampling and edge connections need to be used when the initial roadmap construction fails to connect the start and goal configurations. One common approach is to repeatedly keep adding a number of samples $n$ and then attempting edge connections until the initial and goal belong to the same roadmap component. The approach is likely to succeed since the roadmap is constructed over a simplified and relaxed problem. If such problem is too difficult to solve, then there is little hope that the original motion-planning problem with dynamics can be solved efficiently.

### B. Heuristic Costs based on the Roadmap Abstraction

The roadmap abstraction provides solutions to the simplified and relaxed motion-planning problem. These solutions are used to construct heuristics that effectively guide the expansion of the motion tree toward the goal. In particular, for a configuration $c \in V_{RM}$, the heuristic cost $h(c)$ is defined as the length of the shortest path in $RM = (V_{RM}, E_{RM})$ from $c$ to the goal. The heuristic costs can be computed by a single call to Dijkstra's shortest path algorithm with the goal configuration as the source (Algorithm 1:3).

### C. Using the Roadmap Abstraction to Guide the Expansion

In the state space $S$, a motion tree $\mathcal{T}$ is rooted at the initial state $s_{init}$ and is incrementally expanded by adding new collision-free and dynamically-feasible trajectories as branches. The roadmap abstraction is used to group the tree states according to their nearest roadmap configurations. More specifically, a function $cfg : S \to C$ uses the relaxed problem representation in the configuration space $C$ to provide a mapping from a state $s \in S$ to the corresponding configuration $cfg(s)$. As shown in the examples provided in Section IV-B, $cfg(s)$ could correspond to the position and orientation component of the state $s$.

Each roadmap configuration $c \in V_{RM}$ defines a group

$$group_{\mathcal{T},c} = \{s : s \in \mathcal{T} \wedge c = NearestCfgRM(s)\},$$

where $NearestCfgRM(s) = \operatorname{argmin}_{c' \in V_{RM}} \rho(cfg(s), c')$. In other words, the state $s$ is associated with the roadmap configuration closest to $cfg(s)$ according to $\rho$.

This induces a partition of $\mathcal{T}$ into different groups, labeled according to their nearest roadmap configuration, i.e.,

$$groups_{\mathcal{T}} = \{group_{\mathcal{T},c} : c \in V_{RM} \wedge |group_{\mathcal{T},c}| > 0\}.$$

Proceeding in an incremental fashion, each iteration consists of (i) selecting a group $group_{\mathcal{T},c}$ from $groups_{\mathcal{T}}$, and (ii) adding a collision-free and dynamically-feasible trajectory from a vertex associated with $group_{\mathcal{T},c}$.

*1) Selecting a Tree Group (Algorithm 1:6):* A weight $w(group_{\mathcal{T},c})$ is associated with each group based on the corresponding heuristic cost $h(c)$ and the number of times $group_{\mathcal{T},c}$ has been selected for expansions in the past, i.e.,

$$w(group_{\mathcal{T},c}) = \alpha^{NrSel(group_{\mathcal{T},c})}/(\epsilon + h(c)),$$

where $0 < \alpha < 1$ and $\epsilon > 0$ is used to avoid division by zero in case $h(c) = 0$. The motion tree is then expanded from the group with the maximum weight. Note that this scheme gives preference to tree groups associated with low heuristic costs. Since $h(c)$ is based on the shortest-path distance in the roadmap from $c$ to the goal, it is estimated that progress can be made in reaching the goal by expanding $\mathcal{T}$ from vertices in $group_{\mathcal{T},c}$. The term $\alpha^{NrSel(group_{\mathcal{T},c})}$ serves as a penalty factor to avoid becoming stuck when expansions from $group_{\mathcal{T},c}$ are not feasible due to constraints imposed by obstacles and robot dynamics. As an implementation note, $groups_{\mathcal{T},c}$ is maintained as a maximum heap data structure to allow efficient retrievals and updates.

*2) Expanding from a Tree Group (Algorithm 1:7–19):* The motion tree $\mathcal{T}$ is expanded from $group_{\mathcal{T},c}$ by first selecting a state $s \in group_{\mathcal{T},c}$ and then generating a collision-free and dynamically-feasible trajectory starting at $s$. A target configuration $p$ is sampled and the closest state in $group(\mathcal{T}, c)$ to $p$ is selected for expansion, i.e.,

$$s = \operatorname{argmin}_{s' \in group_{\mathcal{T},c}} \rho(p, cfg(s')).$$

To promote expansions toward the goal, $p$ is sampled near roadmap configurations along the shortest path from $c$ to the goal. More specifically, recall that the shortest roadmap path has already been computed by Dijkstra's algorithm during the computation of the heuristic costs (section III-B). A roadmap configuration $c_i$ along this path is selected uniformly at random and a target $p$ is generated uniformly at random inside a small ball centered at $c_i$. To also promote expansions in new directions, $p$ is sampled at other times uniformly at random from the entire configuration space $C$. To combine these two strategies, with probability $b$ (set to 0.85 in the experiments), $p$ is generated according to the first strategy and with probability $1 - b$ according to the second.

A collision-free and dynamically-feasible trajectory is generated from the selected state $s$ by applying control inputs $u$ for several time steps, stopping when a collision is detected or a maximum number of steps is exceeded. As it is common in sampling-based motion planning, the control $u$ is sampled uniformly at random in order to expand $\mathcal{T}$ in new directions. PID controllers can also be used, when available, to generate a trajectory from $s$ toward the sampled target $p$. Note that exact steering is not required, which makes it easier to design such controllers. Intermediate states along the generated trajectory are added as new vertices to $\mathcal{T}$. When a new state $s_{new}$ is added to $\mathcal{T}$, the corresponding tree group is also added to $groups_{\mathcal{T}}$ if not already there (Algorithm 1:16–19). In this way, the approach has the flexibility to expand the search from new tree groups in future iterations.

## IV. Experiments and Results

### A. Robot Models and Scenes

Experimental validation is provided using a physics-based vehicle model, a high-dimensional snake-like robot model, and an aerial-vehicle model operating in complex environments, as shown in Fig. 1.

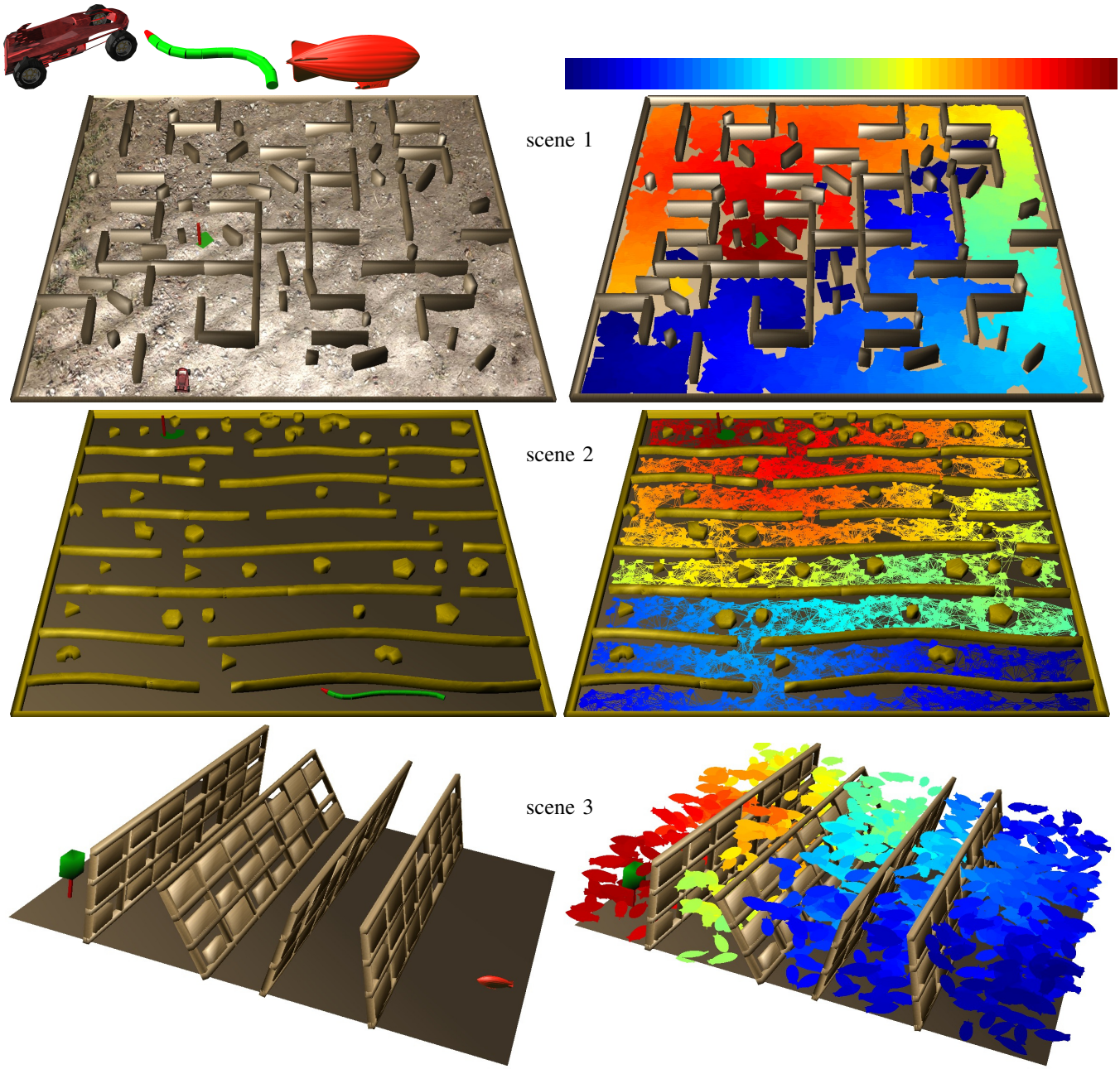


Fig. 1. Robot models and scenes. Figures on the right show the roadmap abstraction. Roadmap configurations are color-coded based on their roadmap distance to the goal (blue: far, red: near, scale provided at the top of the figure). Roadmap edges are shown in the case of scene 2, but not for the other scenes as it clutters the figure. Figures better viewed in color and on screen. Videos showing solutions are provided as supplementary material.

*1) Physics-based Vehicle Model:* In the first scene, a vehicle is required to drive over a bumpy terrain and avoid numerous obstacles scattered throughout the environment in order to reach the goal region. The vehicle is controlled by setting the engine force and changing the steering wheel. The physics-based engine, Bullet [2], is used to model the rigid body dynamics and to compute the motions resulting from applying these external forces.

*2) Snake-Like Robot Model:* In the second scene, a snake-like robot model is required to move through numerous narrow passages and maneuver in tight spaces in order to reach the goal region. This high-dimensional model with nonlinear dynamics is obtained by attaching several trailers to a car that pulls them as it moves. Specifically, the differential equations of motion are as follows (adapted from [5, pp. 731]):

$$\dot{x} = v\cos(\theta_0)\cos(\psi) \quad \dot{y} = v\sin(\theta_0)\sin(\psi)$$
$$\dot{\theta_0} = v\sin(\psi)/L \quad \dot{v} = a \qquad \dot{\psi} = \omega$$
$$\dot{\theta_i} = \tfrac{v}{d}(\sin(\theta_{i-1}) - \sin(\theta_0)) \textstyle\prod_{j=1}^{i-1} \cos(\theta_{j-1} - \theta_j),$$

where $x, y, \theta_0, v, \psi$ denote the position, orientation, velocity, and steering angle of the car; $\theta_i$ denotes the orientation of the $i$-th trailer; $N$ denotes the number of trailers; $L$ and $d$

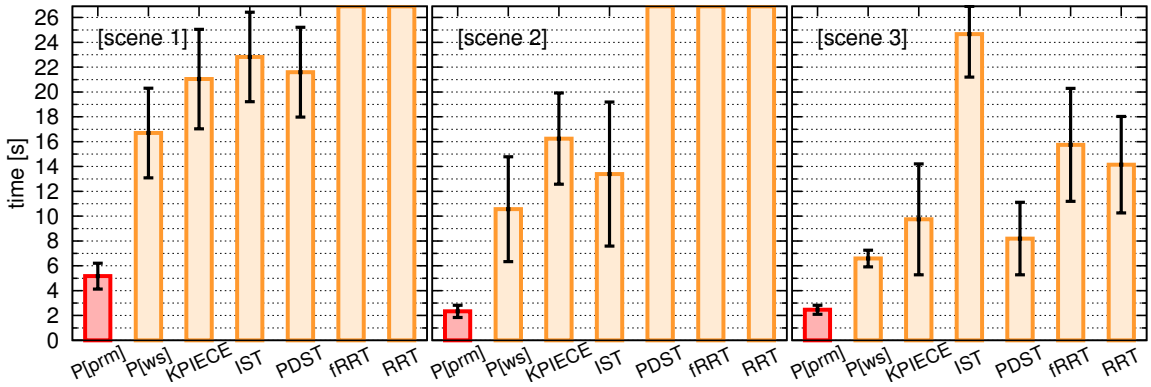


Fig. 2. Results of the comparisons. *P[prm]* refers to the proposed approach when using roadmap abstractions. Running time for *P[prm]* includes the time used for roadmap construction, which constituted a negligible fraction of the overall time. The framework when using workspace decompositions is referred to as *P[ws]*. Segments indicate one standard deviation.

denote the body and hitch lengths. The robot is controlled by setting $a$ (acceleration) and $\omega$ (steering rotational velocity).

*3) Aerial-Vehicle Model:* In the third scene, an aerial vehicle is required to fly through several small openings placed at different heights in order to reach the goal. The differential equations of motion are defined as

$$\dot{x} = v\cos(\theta)\cos(\psi) \qquad \dot{y} = v\sin(\theta)\cos(\psi)$$
$$\dot{\theta} = v\sin(\psi)/L \quad \dot{v} = a \quad \dot{\psi} = \omega \quad \dot{z} = v_z \quad \dot{v}_z = a_z$$

where the new components $z, v_z, a_z$ express the position, velocity, and acceleration along the $z$-axis, as the vehicle flies parallel to the $xy$-plane.

### B. Roadmap Abstractions used in the Experiments

In the case of the snake-like robot model, the roadmap is constructed over $SE(2)$, where a configuration is defined by $(x, y, \theta_0)$. Only the head link is considered. For the physics-based vehicle, the relaxed robot model is again allowed to freely translate and rotate. As this is a ground vehicle model, the roadmap is also constructed over $SE(2)$. For the aerial vehicle, a configuration is defined by $(x, y, z, \theta)$.

In all cases, straight-line interpolations $(1-t)q_i + tq_j$ are used to generate each roadmap edge. Edge collision checking is done by subdivision, as advocated in the literature [4, Chapter 7]. The number of initial samples is set to $n = 10000$ and the number of neighbors is set to $k = 10$. Note that, as described in Section III, an additional 5000 samples are added to the roadmap when the initial roadmap construction fails to connect the start and goal configurations (this is repeated until start and goal are connected). Illustrations of roadmap abstractions are provided in Fig. 1.

### C. Methods used in the Comparisons

Experiments compare the approach when using sampling-based motion planning in combination with roadmap abstractions instead of workspace decompositions. When using roadmaps, the approach is referred to as *P[prm]*. When using workspace decompositions, it is referred to as *P[ws]*. The approach is also compared to other successful motion planners, namely, RRT [6], [7], fRRT [11], PDST [8], IST [9], KPIECE [10]. All the planners use the same code base. Implementations of KPIECE and PDST are based on OMPL [22]. Implementations of fRRT and IST follow the technical descriptions in their respective papers. The RRT implementation uses the connect version with goal bias. Efficient data structures are used for nearest neighbors [21].

### D. Problem Instances

Due to the probabilistic nature of sampling-based motion planning, the computational efficiency of each approach on a particular scene is based on results from 60 different runs, where the $i$-th run uses the $i$-th query. Sixty different queries are generated for each scene before hand and used by all the planners. The five worst and the five best runs are discarded to avoid the influence of outliers. Results report the average time and standard deviation of the remaining runs. For the first scene, queries are generated by placing the car and the goal at random positions. For the second and third scenes, the robot is placed at random on one end of the scene and the goal is placed at random at the opposite end. Running time for each run includes everything from reading the input to reporting that a solution is found. Experiments are run on an Intel Core i7 machine (CPU: 1.90GHz, RAM: 4GB) using Ubuntu 13.04. Code is compiled with GNU g++-4.7.2.

### E. Results

Fig. 2 provides a summary of the results. Results indicate that the proposed approach performs significantly better than the other planners used in the comparisons. Also, *P[prm]*, which refers to the proposed approach when using roadmap abstractions, shows significant improvements over its counterpart that uses workspace decompositions (referred to as *P[ws]*). The reasons for these improvements come from the fact that the roadmap abstractions more effectively guide the motion-tree expansions as the roadmap uses a geometric shape that more closely relates to the full robot model as opposed to just using a point. Moreover, the roadmap sampling and edge connections help identify collision-free regions from which the tree search can be expanded. By capturing the connectivity of the underlying free configuration space, the roadmap abstractions provide the planner with more reliable suggestions of how to effectively reach the goal.

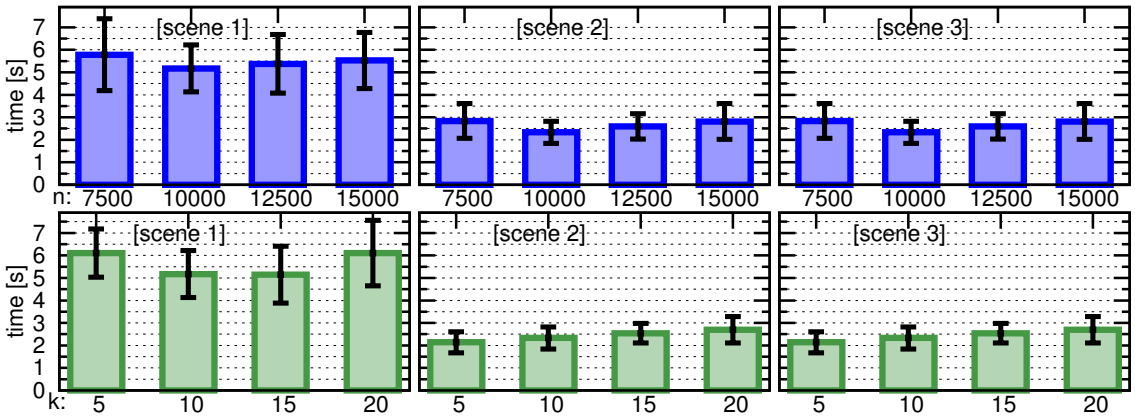


Fig. 3. Impact of the roadmap on the overall computational efficiency of the proposed approach. Top figure shows the results when varying the number of initial roadmap samples. Bottome figure shows the results when varying the number of nearest neighbors.

Fig. 3 shows the results when varying the number of roadmap configurations and nearest neighbors. As the results indicate, *P[prm]* performs well for a wide range of values.

## V. Discussion

This paper proposed using probabilistic roadmap abstractions in combination with sampling-based motion planning to effectively plan collision-free and dynamically-feasible robot motions to reach a given goal. Roadmap abstractions are constructed over a low-dimensional configuration space obtained by considering relaxed and simplified representations of the robot model and its underlying motions. By capturing the connectivity of the corresponding free configuration space, roadmap abstractions provide the approach with promising suggestions of how to effectively expand the tree-based search in the full state space of the robot.

The approach is geared toward high-dimensional motion-planning problems for mobile robots with nonlinear dynamics. Experimental results showed significant computational speedups when using roadmap abstractions instead of workspace decompositions as well as when comparing the proposed approach to other successful motion planners.

The combination of roadmap abstractions in configuration spaces with tree-based exploration of the state space opens up new venues for research. In fact, roadmap planners have been extensively improved over the years to handle increasingly complex motion-planning problems in configuration spaces. By taking advantage of this extensive body of literature, we plan to improve the sampling and connection strategies used in the roadmap construction to more effectively capture the connectivity especially inside narrow passages.

## References


[1] J. T. Schwartz and M. Sharir, "A survey of motion planning and related geometric algorithms," *Artificial Intelligence*, vol. 37, pp. 157 – 169, 1988.

[2] E. Coumans, "Bullet physics engine," 2012, http://bulletphysics.org/.

[3] M. Smith, "Open dynamics engine," 2006, www.ode.org.

[4] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, 2005.

[5] S. M. LaValle, *Planning Algorithms*. Cambridge, MA: Cambridge University Press, 2006.

[6] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[7] S. M. LaValle, "Motion planning: The essentials," *IEEE Robotics & Automation Magazine*, vol. 18, no. 1, pp. 79–89, 2011.

[8] A. M. Ladd and L. E. Kavraki, "Motion planning in the presence of drift, underactuation and discrete system changes," in *Robotics: Science and Systems*, 2005, pp. 233–241.

[9] K. E. Bekris and L. E. Kavraki, "Informed and probabilistically complete search for motion planning under differential constraints," in *International Symposium on Search Techniques in Artificial Intelligence and Robotics*, 2008.

[10] I. A. Şucan and L. E. Kavraki, "A sampling-based tree planner for systems with complex dynamics," *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 116–131, 2012.

[11] S. Kiesel, E. Burns, and W. Ruml, "Abstraction-guided sampling for motion planning," in *Symposium on Combinatorial Search*, 2012, pp. 162–163, also as UNH CS Technical Report 12-01.

[12] E. Plaku, L. E. Kavraki, and M. Y. Vardi, "Motion planning with dynamics by a synergistic combination of layers of planning," *IEEE Transactions on Robotics*, vol. 26, no. 3, pp. 469–482, 2010.

[13] M. Maly and L. E. Kavraki, "Low-dimensional projections for SyCLoP," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 420–425.

[14] E. Plaku, "Robot motion planning with dynamics as hybrid search," in *AAAI Conference on Artificial Intelligence*, 2013, pp. 1415–1421.

[15] L. E. Kavraki, P. Švestka, 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, 1996.

[16] D. Hsu, J. C. Latombe, and H. Kurniawati, "On the probabilistic foundations of probabilistic roadmap planning," *International Journal of Robotics Research*, vol. 25, no. 7, pp. 627–643, 2006.

[17] J. Denny and N. M. Amato, "Toggle PRM: A coordinated mapping of C-free and C-obstacle in arbitrary dimension," in *International Workshop on Algorithmic Foundations of Robotics*, ser. Springer Tracts in Advanced Robotics, 2013, vol. 86, pp. 297–312.

[18] H. Keller, *Numerical Methods for Two-Point Boundary-Value Problems*. New York, NY: Dover, 1992.

[19] P. Cheng, E. Frazzoli, and S. LaValle, "Improving the performance of sampling-based motion planning with symmetry-based gap reduction," *IEEE Transactions on Robotics*, vol. 24, no. 2, pp. 488–494, 2008.

[20] N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones, and D. Vallejo, "Choosing good distance metrics and local planners for probabilistic roadmap methods," in *IEEE International Conference on Robotics and Automation*, Leuven, Belgium, 1998, pp. 630–637.

[21] S. Brin, "Near neighbor search in large metric spaces," in *International Conference on Very Large Data Bases*, 1995, pp. 574–584.

[22] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, pp. 72–82, 2012, http://ompl.kavrakilab.org.