A Motion Planner for a Hybrid Robotic System with Kinodynamic Constraints

Erion Plaku Lydia E. Kavraki Moshe Y. Vardi

***Abstract*— The rapidly increasing complexity of tasks robotic systems are expected to carry out underscores the need for the development of motion planners that can take into account discrete changes in the continuous motions of the system. Completion of tasks such as exploration of unknown or hazardous environments often requires discrete changes in the controls and motions of the robot in order to adapt to different terrains or maintain operability during partial failures or other mishaps.**

The contribution of this work toward this objective is the development of an efficient motion planner for a hybrid robotic system. The controls and motion equations of the robot could change discretely in order to enable the robot to operate in different terrains. The framework in this paper blends discrete searching with sampling-based motion planning for continuous state spaces and is well-suited for robotic systems modeled as hybrid systems with numerous discrete modes and transitions. This multi-layered approach offers considerable improvements over existing methods addressing similar problems, as indicated by the experimental results.

I. INTRODUCTION

Nowadays robotic systems are designed to perform increasingly complex tasks. Robots are expected to explore unknown, dynamic, or possibly hazardous environments, quickly modifying their controls to respond to unanticipated changes in the environment, mishaps, or other failures. For example, a vehicle may be required to employ different controls over different terrains due to safety issues, and a reconfigurable robot may change its shape and use different gaits to climb, crawl, or walk fast. Such changes in the robot behavior are often realized by instantaneously switching to a different operating mode.

A challenging yet important problem is the development of motion planners for these *hybrid* robotic systems. The challenge lies in that a hybrid system combines discrete and continuous dynamics by associating continuous dynamics with each operating mode and using discrete logic to switch between modes. Fig. 1 provides an illustration of a hybrid robotic system, where controls and motion equations depend on the operating mode. Necessary modifications in controls and motion equations to adapt to changes in environment are modeled as discrete transitions between different modes. The objective is to find a sequence of continuous trajectories interleaved with discrete transitions that enable the robot to move from an initial to a final state associated with possibly different modes, while respecting collision-avoidance, kinodynamic, and other constraints.

Work on this paper has been supported in part by NSF CNS 0615328 (EP, LEK, MYV), NSF 0308237 (EP, LEK), GM078988 (EP, LEK), a Sloan Fellowship (LEK), and NSF CCF 0613889 (MYV). Experiments reported in this paper have been obtained on equipment supported by NSF CNS 0454333, and NSF CNS 0421109 in partnership with Rice University, AMD, and Cray. The authors are with the Department of Computer Science, Rice University, Houston, TX 77005, USA {plakue, kavraki, vardi}@cs.rice.edu

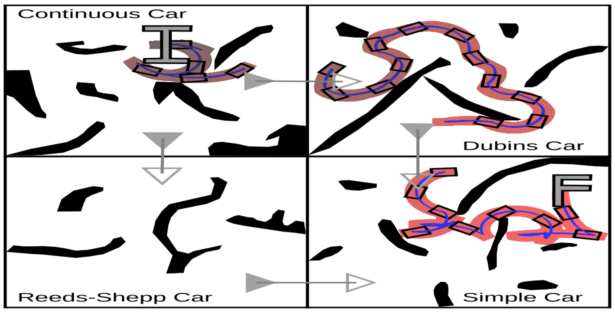


Fig. 1. The controls and motion equations of the robot could change discretely to quickly respond to changes in terrain and other driving conditions associated with different regions of a workspace. In the top-left region, the robot drives as a car controlled by accelerating and slowly turning the steering wheel. In the other regions, controls and motion equations are modified to respond to changes in terrain. The robot can leave the current region only when it reaches a guard set, which indicates a discrete transfer to the next region (from a gray triangle to an empty triangle). After the discrete transfer, the robot moves according to the controls and motion equations associated with the new region. The objective is to find a collision-free trajectory (obstacles shown in black) from some initial to some final state, indicated by polygons shaped as the "I" and "F" letters, respectively. The solution (shown as a gradient) consists of continuous trajectories interleaved with discrete transitions. The blue line traces out the path of the centroid of the box representing the car's geometry. Intermediate configurations certain distance apart are also shown along the solution trajectory.

Methods developed in hybrid systems that are also applicable to motion planning generally rely on symbolic reasoning, decompositions of the state space, or simplified abstractions [1]–[3]. The exponential dependency on the dimension, complexity of dynamics, and other factors limit the practicality of these methods only to simple systems with five to six dimensions, linear dynamics, or no controls, underscoring the need for alternative approaches [1]–[3].

In the absence of discrete modes, sampling-based motion planning has successfully been used for complex robotic systems with high-dimensional continuous state spaces [4]–[9]. In addition to collision avoidance, sampling-based methods can take into account kinodynamic constraints and plan even for robots whose motion is governed by nonlinear dynamics.

The use of the Rapidly-exploring Random Tree (RRT) [5], [10], a sampling-based method, has recently shown promise as a motion planning method for hybrid robotic systems with few discrete states [11], [12]. The applicability of RRT to more complex hybrid systems, especially systems with a large number of discrete modes and transitions, remains however challenging, since (i) an RRT relies heavily on distance metrics that should indicate how easily the hybrid system can transition from one state to another – the definition of such distance metrics is difficult, even in the case of continuous systems, since it is not even clear that a distance metric can express this property [13]; (ii) the growth of an RRT significantly slows down as the number of nodes in the tree increases [4], [5], [12], limiting the ability of RRT to successfully explore the continuous state spaces of systems with a large number of discrete modes and transitions; (iii) the exploration of continuous state spaces by an RRT is local and frequently gets stuck in certain regions [4], [6], [7], [12].

More recent work, such as the Sampling-based Roadmap of Trees (SRT) [6], [7], which may be relevant in the context of hybrid systems, shows how to combine RRT and other tree-based methods [14] with PRM [15] in order to address some of the issues observed with single-tree planners.

The contribution of this work is the development of an efficient motion planner for hybrid robotic systems that blends in novel ways discrete and continuous sampling-based searching. The work in this paper complements and extends the SRT framework by using a discrete component to guide the exploration. The discrete component uses the graph of discrete transitions and information collected during previous explorations to guide the sampling-based component to explore relevant regions of the continuous state spaces. In contrast to previous work [11], [12], the multi-layered approach developed in this paper is well-suited for systems with many discrete states and transitions and offers considerable computational improvements over existing methods, as indicated by the experimental results.

The rest of the paper is as follows. Sections II and III describe the problem considered in this work and the proposed motion planning framework. Experiments and results are described in section IV and the conclusion is in section V.

II. Problem Description

A given workspace is divided into a number of nonoverlapping regions $R_1, \ldots, R_N$. The robot motion inside each region R_i is governed by a set of, possibly nonlinear, differential equations $f_i : X_i \times U_i \rightarrow \mathrm{Tg}X_i$, where X_i, $\mathrm{Tg}X_i$, and U_i are the continuous state space, tangent of X_i, and the set of control inputs. Each continuous state space X_i could include derivatives of different orders, e.g., velocity and acceleration of a car. As an example consider the case of a robot which behaves as a car controlled by accelerating and turning the steering wheel in one region, while in another region the robot behaves as a car which cannot reverse and cannot make sharp turns. A discrete transition from R_i to R_j occurs when the robot enters a part of R_i referred to as the guard set $G_{ij} \subset R_i$. It is only through a discrete transition that the robot could move from one region to the other. If there are no discrete transitions, the robot cannot move outside the current region. The guard set could be thought of as indicating necessary changes in the way the robot should be controlled. For example, when approaching sharp turns or driving in rough terrain the robot should reduce the speed. Upon entering the guard set the robot is discretely transferred onto some part $J_{ij} \subset R_j$ where it continues to move according to f_j. Thus, a trajectory of the robot consists of one or more continuous trajectories interleaved with discrete transitions. Fig. 1 provides an illustration.

The robot behavior is modeled as a hybrid system represented by the tuple $H = (S, f, I, F, E)$, where $S = Q \times X$; Q is a discrete and finite set; $X = \{X_1, \ldots, X_N\}$ represents the different continuous state spaces; $f = \{f_1, \ldots, f_N\}$ indicates the motion equations associated with the continuous state spaces; $I, F \subset S$ are the sets of initial and final states, respectively. The discrete state $q_i \in Q$ is associated with the region R_i. A discrete transition $(q_i, q_j) \in E \subset Q \times Q$ is associated with the guard $G_{ij} \subset R_i$ and reset $J_{ij} \subset R_j$ sets.

The objective is to construct one or more feasible trajectories that enable the robot to move from some initial state $s_0 \in I$ to some final state $p \in F$.

III. Methods

The proposed framework is a multi-layered approach that interleaves two components, GUIDE and EXPLORE, which can be categorized as search methods for discrete and continuous spaces, respectively. GUIDE searches the discrete space (Q, E) for discrete solution sequences, i.e., sequences of discrete transitions from some $s_0 \in I$ to some $p \in F$, and EXPLORE is a sampling-based approach for exploring continuous state spaces. GUIDE and EXPLORE work in tandem and are iteratively called until a solution trajectory is obtained or the allocated computation time is exceeded. Details of EXPLORE and GUIDE are found in sections III-A and III-B, respectively. Pseudocode for the interplay of GUIDE and EXPLORE is given in Fig. 2 and an illustration is provided in Fig. 3.

The efficiency of motion planners for difficult problems depends on their ability to focus the exploration on important parts of the state space [16]. For this reason, the objective of the motion planner in this work is to estimate the importance of different regions of the state space and allocate more exploration time to the more promising regions. The motion planner associates a weight Imp_{ij} with each $(q_i, q_j) \in E$ which expresses an estimate on the importance of $S_{ij} = S_i \cup S_j$ (see section III-B for details). The time allocated for the exploration of S_{ij} is proportional to Imp_{ij}. The estimate Imp_{ij} is updated after each exploration to reflect new information gathered from the exploration. In this way, the motion planner spends most of the time exploring the most promising regions. At the same time, the motion planner does not completely ignore less promising regions, but instead spends less time exploring them. In this way, the framework aims to strike a desired balance between greedy and methodical search, which is important in making the motion planner efficient and robust.

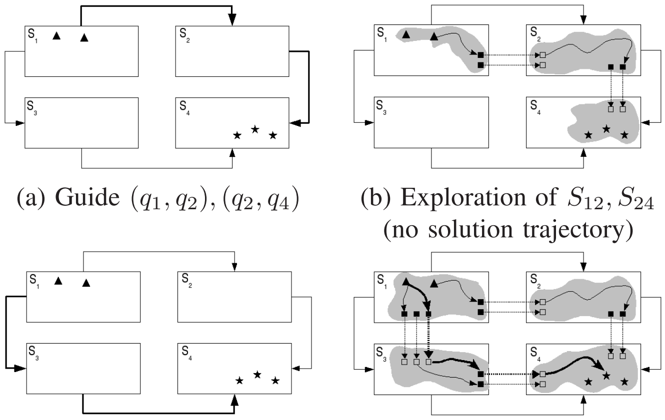
Initially, all the weights Imp_{ij} are set to zero (lines 1-2). Each step (lines 4-14) consists of a computation of a sequence σ of discrete transitions by GUIDE (lines 5-6) followed by exploration of the continuous state spaces by EXPLORE (lines 7-14). A step ends when a solution

A motion planner for hybrid robotic systems

Input:
$H = (S, f, I, F, E)$, a hybrid robotic system
$T \in \mathbb{R}^{>0}$, upper bound on computation time
Output:
A solution trajectory τ or FAILURE if no trajectory is found

```
for each (q_i, q_j) ∈ E do
    Imp_ij ← 0, initial estimate expressing importance of S_ij
t ← 0
while t < T and no solution is found do
    σ ← GUIDE(H), a relevant sequence of discrete transitions
    t ← t + t_g, where t_g is the computation time used by GUIDE
    t_c ∈ [0, T − t] ← time allocated for exploration using σ
    for each (q_i, q_j) ∈ σ do
        t_ij ← t_c Imp_ij / Σ_{(q_k, q_ℓ) ∈ σ} Imp_kℓ
        use EXPLORE(H) to explore S_ij for t_ij units of time
        Imp_ij ← update estimate based on exploration of S_ij
    t ← t + t_c
    τ ← search roadmap for solution trajectory
    if τ ≠ NIL then return τ
return FAILURE
```

Fig. 2. High-level illustration of the proposed motion planner. GUIDE utilizes the graph of discrete transitions E and information gathered from the exploration of the continuous state spaces to compute a relevant sequence of discrete transitions σ. EXPLORE uses σ and the SRT framework to guide the exploration toward promising regions of the continuous state spaces. A solution trajectory is obtained when, as part of the exploration, a trajectory is constructed from an initial state $s_0 \in I$ to a final state $p \in F$.



(a) Guide $(q_1, q_2), (q_2, q_4)$ (b) Exploration of S_{12}, S_{24} (no solution trajectory)
(c) Guide $(q_1, q_3), (q_3, q_4)$ (d) Exploration of S_{13}, S_{34} (solution trajectory)

Fig. 3. High-level illustration of the interplay between GUIDE and EXPLORE. Elbow arrows indicate edges of E. Triangles and stars indicate some initial and final states, respectively. (a) GUIDE computes the sequence $\sigma_1 = (q_1, q_2), (q_2, q_4)$, indicated by the thick elbow arrows. (b) EXPLORE uses σ_1 to focus the exploration on S_{12} and S_{24}. The gray areas indicate explored parts of the continuous state spaces. The arrows from black squares to empty squares indicate discrete transitions from one continuous state to another. (c) GUIDE computes another sequence $\sigma_2 = (q_1, q_3), (q_3, q_4)$, since EXPLORE was not able to construct a solution trajectory using σ_1. (d) As a result of the exploration guided by σ_2, a solution trajectory is obtained (thick trajectory lines).

trajectory is constructed or when the computation time allowed for one iteration is exceeded. GUIDE uses information collected from all previous calls to EXPLORE to update Imp_{ij} estimates and select more relevant discrete sequences. EXPLORE uses such sequences to focus the exploration on the more promising regions. Fig. 3 provides an illustration.

A. *Sampling-based Component:* EXPLORE

The discrete sequence σ computed by GUIDE indicates which regions should be explored. EXPLORE extends the SRT framework [6], [7] to efficiently explore these regions by generating and connecting trees via feasible trajectories. The time allocated for exploring each S_{ij}, associated with $(q_i, q_{i+1}) \in \sigma$, is proportional to the weight Imp_{ij}. Thus, more time is spent exploring important regions.

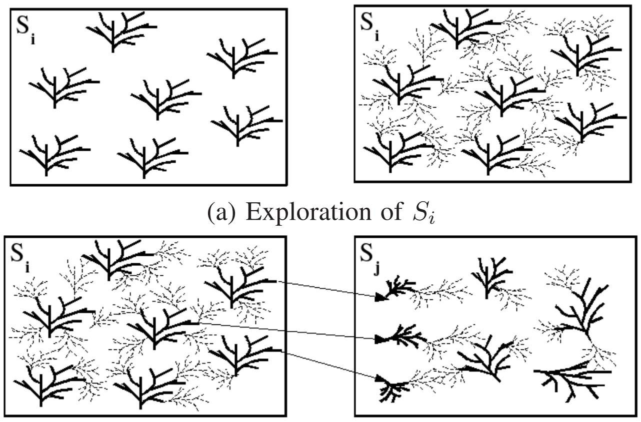
1) Exploration of S_{ij}: The exploration of S_{ij} is based on explorations of S_i and S_j. The exploration of S_i, similarly S_j, constructs and maintains a roadmap of trees. Trees are grown forward and backward in time in different parts of S_i by sampling roots and exploring around the roots using a tree-based method [9], [10], [14]. Neighboring trees are then connected by growing trees toward each-other. For each neighboring pair of trees (T', T''), several close pairs of states of T' and T'' are quickly checked to determine if local trajectories can be generated from one state in T' to another state in T''. If a local trajectory can be constructed, the trees are successfully connected and no further computation takes place. Otherwise, a more complex tree-connection algorithm is executed, e.g., bi-directional RRT. During the tree connection, additional states are typically added to the trees T' and T''. Possible remaining gaps due to the kinodynamic nature of the problem are closed using available steering or numerical methods [17]. The resulting roadmap of trees explores S_i, as Fig. 4(a) illustrates. The advantage of this approach is that it results in a global exploration method, and, unlike RRT, does not get stuck easily [6], [7].

When exploring S_{ij}, it is important to guide the exploration toward the guard G_{ij} to facilitate the connection of continuous trajectories from S_i to S_j. When trees in S_i reach G_{ij}, a discrete transition is made onto some state in S_j. Trees in S_i could thus have branches extending to S_j, as Fig. 4(b) illustrates. Such branches can be further grown to explore S_j and connect trajectories from S_i to S_j.

2) Quantifying Information Collected During Exploration: Information collected during exploration is used by GUIDE to select discrete transition sequences that are deemed important in the construction of solution trajectories. EXPLORE quantifies such information by assigning to each S_{ij} a score Score_{ij} which reflects how well S_{ij} is explored. One possibility is to define Score_{ij} as a function of (i) Exp_{ij}, the exploration of S_{ij}, and (ii) Conn_{ij}, the connections of trees in S_{ij}, as follows:

$$\mathrm{Score}_{ij} = w\mathrm{Exp}_{ij} + (1 - w)\mathrm{Conn}_{ij}, \quad 0 < w < 1,$$

where $\mathrm{Exp}_{ij} = (\mathrm{Exp}_i + \mathrm{Exp}_j)/2$ and Exp_i and Exp_j are the estimations of the exploration of S_i and S_j, respectively. Drawing from quasirandom sampling [4], this work uses a variant of the dispersion measure for estimating Exp_i (Exp_j). Intuitively, S_i is explored well if new samples are close to existing samples. The exploration of S_i is estimated by generating a sample s at random and determining if any of the existing samples are inside a small ball centered at



(a) Exploration of S_i

(b) Exploration of S_{ij}

Fig. 4. (a) (left) Several trees are grown in S_i. (right) Connections between trees are computed by growing trees toward each-other. Thinner lines indicate the resulting additional exploration. (b) As the growth of the trees in S_i is guided toward G_{ij}, several branches extend from S_i to S_j, as indicated by the arrows. Such branches are further grown to explore S_j. Connections are also made from these branches to other trees rooted in S_j.

s. Efficient data structures such as [18], [19] can be used to speed up such computations. The process is repeated several times and Exp_i is set equal to the fraction of times an existing sample was found to be close to a random sample.

A randomized approach is also used to compute Conn_{ij}. Let s' and s'' be two samples drawn uniformly at random from samples in S_i and determine if there is a trajectory from s' to s'' by searching the roadmap. Let ℓ_i denote the number of times a trajectory is found when the above process is repeated k times. The computation of ℓ_j is similar, but draws s' and s'' from samples in S_j, while the computation of ℓ_{ij} draws s' and s'' from samples in S_i and S_j, respectively. Then, $\mathrm{Conn}_{ij} = (\ell_i + \ell_j + \ell_{ij})/(3k)$. Thus, Conn_{ij} indicates how well samples in S_{ij} are connected.

B. Discrete Component: GUIDE

The objective of GUIDE is to compute sequences of discrete transitions to guide the exploration. Such sequences are computed using the graph of discrete transitions. The selection is biased toward transitions that, according to internal estimates, further improve the exploration. A probability p_{ij} is associated with each $(q_i, q_j) \in E$, expressing the running estimate of the relevance of S_{ij} for constructing solution trajectories. These probabilities are updated based on new information gathered at each exploration step.

1) Relevance of Discrete Transitions: An importance Imp_{ij} estimate is used to express how much additional exploration of S_{ij} facilitates the construction of solution trajectories. The probability p_{ij} on the relevance of S_{ij} is then defined by normalizing Imp_{ij}, i.e., $p_{ij} = \mathrm{Imp}_{ij} / \sum_{(q_\ell, q_m) \in E} \mathrm{Imp}_{\ell m}$. The importance estimate is tightly connected to information gathered during exploration of S_{ij}. Let Score_{ij} represent the quality of this information, as described in section III-A. The estimate Imp_{ij} is based on the history $\mathrm{Score}_{ij}^0, \ldots, \mathrm{Score}_{ij}^n$ of scores obtained after each exploration of S_{ij} for $t_0, \cdots, t_n$ units of time, where $\mathrm{Score}_{ij}^0 = 0$ and $t_0 = 0$. Then, Imp_{ij} is defined as an exponentially decaying sum of changes of Score_{ij}, i.e.,

$$\mathrm{Imp}_{ij} = \sum_{h=1}^{n} \alpha^{n-h} \frac{\mathrm{Score}_{ij}^h - \mathrm{Score}_{ij}^{h-1}}{t_h}, \quad 0 < \alpha \leq 1.$$

Recent changes have exponentially higher weights, since recent scores more accurately reflect the relevance of S_{ij}.

2) Selecting Discrete Solution Trajectories: The relevance of a discrete sequence is estimated as the product of the probabilities associated with its discrete transitions. In each stage of the testing process, a discrete solution sequence is selected according to its probability. In this way, the selection is biased toward discrete sequences that according to internal estimates facilitate the construction of solution trajectories. Variations of A*, Dijkstra's algorithm, and other graph search methods can be used to compute such sequences.

We note that only acyclic sequences of discrete transitions are used as guides, while in fact, solution trajectories may contain discrete cycles. To address this issue, as described in section III-A, the exploration of S_{ij} is biased not only toward G_{ij}, but also toward other guards. Once a guard is met, a trajectory is constructed from one discrete state to another. Guiding the exploration toward different guards allows the connection of continuous trajectories such that the resulting trajectory could contain discrete cycles.

IV. EXPERIMENTS AND RESULTS

For the initial validation of the framework, we have chosen problems where the motion equations of the robot change frequently as the robot moves from one region to the next, as described in section II. Such discrete changes considerably add to the complexity of the already complex motion planning for a continuous state space. As the experimental results indicate, the integration of GUIDE and EXPLORE results in an efficient motion planner for hybrid robotic systems.

The Rice Terascale Cluster, PBC Cluster, and Rice Cray XD1 Cluster ADA were used for code development. Experiments were run on ADA, where each processor runs at 2.2GHz and has 2GB of RAM.

A. Robot Models

Several motion equations are used to define how the robot moves in different regions of the workspace: a simple kinematic car, Reeds-Shepp car, Dubins car, and a continuous car. Fig. 1 provides an illustration. Detailed descriptions of these models can be found in [4], [5].

1) Simple Car: The motion equations are $\dot{x} = u_0 \cos(\theta)$; $\dot{y} = u_0 \sin(\theta)$; $\dot{\theta} = u_0 \tan(u_1)/L$, where (x, y, θ) is the car configuration; $u_0 \in [-1, 1]$ and $u_1 \in (-\pi/4, \pi/4)$ are the speed and steering wheel controls; and L, a constant, is the distance between the front and rear axles.

2) Reeds-Shepp Car: The speed control is limited only to "reverse," "park," and "drive," i.e., $u_0 \in \{-1, 0, 1\}$.

3) Dubins Car: An even more restricted model is the Dubins car which cannot reverse, i.e., $u_0 \in \{0, 1\}$.

4) Continuous Car: The velocity v and steering angle ϕ are controlled by changing the acceleration, $u_0 \in [-1, 1]$, and the rotational velocity of the steering wheel, $u_1 \in [-1, 1]$, as described below: $\dot{x} = v\cos(\theta)$; $\dot{y} = v\sin(\theta)$; $\dot{\theta} = v\tan(\phi)/L$; $\dot{v} = u_0$; $\dot{\phi} = u_1$.

In each experiment, the robot geometry is represented by a 0.086×0.051 box and each region R_i has unit dimension.

B. Results

1) Validating GUIDE *and* EXPLORE*:* Fig. 5 illustrates how GUIDE and EXPLORE tailor future explorations of the continuous state spaces based on information collected during previous explorations. As discussed earlier, the efficiency of motion planners for difficult problems depends on their ability to focus the exploration on difficult parts of the continuous state spaces.

Fig. 5(a) shows a workspace divided into three regions that have the same motion equations of the simple kinematic car model, but different number and distribution of obstacles. These regions can be considered as easy (R_0), medium (R_1), and hard (R_2). Since a solution trajectory requires the robot to move through all three regions, we would like the motion planner to spend most of the time exploring the hard region and spend little time exploring the easy region. This desired behavior is achieved by using information collected during previous explorations to measure the progress in different regions and bias future explorations toward promising regions. Fig. 5(b) shows the fraction of time allocated to the exploration of each region as a function of the total time. As shown in Fig. 5(b), the motion planner initially explores more the easy region, since this region yields the most progress. After quickly exploring the easy region, the motion planner explores the other regions more. This desired behavior is shown in Fig. 5(b) by the sharp drop on the fraction of the total time spent exploring the easy region. We also note that the exploration is now biased toward the medium region, since this region is currently the most promising. However, after spending some time exploring the medium region, the internal estimates indicate that the medium region has been adequately explored, and so the motion planner focuses the exploration on the hard region. As desired, the motion planner spends most of the time exploring the hard region and little time exploring the easy region.

Fig. 5(c, d) also illustrate the interplay between GUIDE and EXPLORE. The workspace in Fig. 5(c) is divided into five regions. The robot can reach a final state by following regions R_2, R_3, and R_4, referred to as guide A, or regions R_2, R_0, R_1, referred to as guide B. Fig. 5(d) shows the fraction of time the motion planner spends exploring each of these alternative routes as a function of the total exploration time. As Fig. 5(d) indicates, initially the motion planner makes more progress using guide A, since the regions of guide A have more open space. After the initial rapid progress, the motion planner has difficulty making further progress using guide A, since it requires connecting samples separated by a narrow passage in the top-right region. The internal estimates indicate that guide B is now more promising and thus the motion planner allocates more time to the exploration of guide B. After favoring guide B over guide A for quite some time, the motion planner switches back to guide A as the most promising. This is due to the fact that samples separated by the narrow passage have been successfully connected. Such connections indicate significant improvements in the progress estimates. The motion planner is then able to find a solution trajectory (not shown) using guide A.

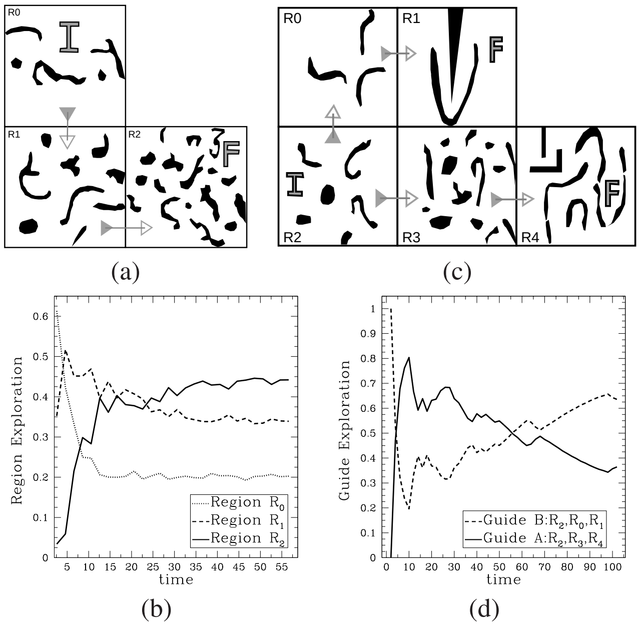


Fig. 5. Guiding the exploration toward promising regions. (a) The top, bottom-left, and bottom-right regions of the workspace can be considered as easy, medium, and hard, respectively. (b) The fraction of time (y axis) spent by the motion planner exploring different regions as a function of the total time (x axis). As desired, the motion planner spends most of the time exploring the hard region and little time exploring the easy region. (c) A trajectory from an initial to a final state can be obtained by following any of the two alternative routes. (d) The fraction of time (y axis) spent by the motion planner exploring each of the two alternative routes as a function of the total time.

The results in Fig. 5 indicate that GUIDE guides EXPLORE toward regions whose additional exploration would allow the motion planner to make further progress. EXPLORE, by drawing from the SRT framework, effectively explores these regions. At the same time, the motion planner does not completely ignore alternative routes, but instead spends less time exploring such routes.

2) Computational Efficiency: Fig. 1 and 6 show the problems used to test the efficiency of the motion planner for hybrid robotic systems. In Fig. 1, the workspace is divided into four regions and a different set of motion equations is used in each region. A trajectory from an initial to a final state can be obtained by following two alternative routes. The motion planner is capable of solving this problem in 42.5s. The computation time is obtained as an average of 20 runs initialized with different pseudorandom seeds. We also implemented the motion planner described in [12] to compare the performance of the motion planner in this work to existing methods addressing similar problems. Although we made every effort to implement the motion planner in [12]

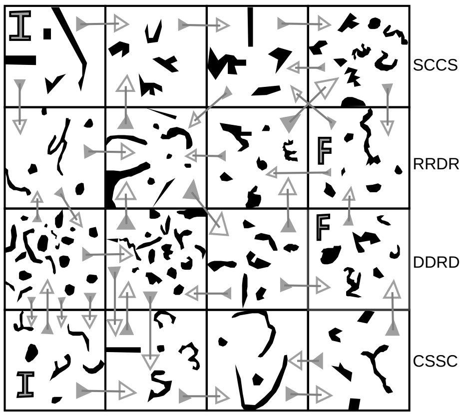


Fig. 6. The workspace is divided into $16 = 4 \times 4$ different regions corresponding to 16 discrete modes. The i-th letter in a four-letter string displayed at the end of each row indicates the dynamics of the robot associated with the i-th region of that row, counting from left to right. The letter S, R, D, or C denotes a simple, Reeds-Shepp, Dubins, or continuous car, respectively. The average total time to find a solution is 105s.

as best as we could, we also note that it is in general difficult to compare different algorithms since not all the details necessary to obtain the best implementation are available. Our implementation of [12] required an average of 110.5s to solve the same problem using the same experimental setup as our method. What is important to note are not the actual running times, which despite our best efforts are still subject to coding issues, but the computational efficiency of motion planners as the complexity of the problem increases.

A more challenging example is shown in Fig. 6. The hybrid system model has 16 discrete modes. There also many discrete transitions (34) and numerous alternative guides. The solution of this problem requires generating many trees with thousands of nodes. Even for this challenging problem, the motion planner in this work is able to find a solution trajectory in 89.3s. The motion planner based on the work in [12] requires a considerably longer time, i.e., 535.8s.

These results indicate that the integration of GUIDE and EXPLORE produces an effective motion planner for hybrid robotic systems that is at least as efficient, if not more, as recent motion planners [11], [12]. Furthermore, the results also indicate that the computational advantages offered by the motion planner in this work become more pronounced as hybrid robotic systems with larger numbers of discrete modes and transitions are considered.

V. DISCUSSION

Motion planning for hybrid robotic systems is a challenging yet important problem that requires taking into account the interplay between continuous behaviors associated with different discrete modes of operations.

The framework developed in this work carefully integrates in novel ways discrete searching with sampling-based motion planning for continuous state spaces. The discrete component uses the graph of discrete transitions and information gathered during previous explorations to guide the sampling-based component to explore relevant parts of the different continuous state spaces. The result is an effective motion planner for hybrid robotic systems especially well-suited for systems with numerous discrete modes and transitions whose continuous behavior in each discrete mode is defined by a possibly different set of complex motion equations. In future work, we will investigate further improvements to the initial framework developed in this work and applications to more realistic 3D examples.

REFERENCES

[1] C. J. Tomlin, I. Mitchell, A. Bayen, and M. Oishi, “Computational techniques for the verification and control of hybrid systems,” *Proceedings of the IEEE*, vol. 91, no. 7, pp. 986–1001, 2003.

[2] C. Chutinan and B. H. Krogh, “Computational techniques for hybrid system verification,” *IEEE Transactions on Automatic Control*, vol. 48, no. 1, pp. 64–75, 2003.

[3] E. Clarke, A. Fehnker, Z. Han, B. Krogh, O. Stursberg, and M. Theobald, “Verification of hybrid systems based on counterexample-guided abstraction refinement,” in *Inter. Conf. on Tools and Algorithms for the Construction and Analysis of Systems*, H. Garavel and J. Hatcliff, Eds., 2003, vol. 2619, pp. 192–207.

[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*. Cambridge, MA: MIT Press, 2005.

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

[6] E. Plaku, K. E. Bekris, B. Y. Chen, A. M. Ladd, and L. E. Kavraki, “Sampling-based roadmap of trees for parallel motion planning,” *IEEE Transactions on Robotics*, vol. 21, no. 4, pp. 597–608, 2005.

[7] E. Plaku and L. E. Kavraki, “Distributed sampling-based roadmap of trees for large-scale motion planning,” in *IEEE Inter. Conf. on Robotics and Automation*, Barcelona, Spain, 2005, pp. 3879–3884.

[8] A. M. Ladd and L. E. Kavraki, “Fast tree-based exploration of state space for robots with dynamics,” in *Workshop on Algo. Found. of Robot.*, 2005, pp. 297–312.

[9] G. Sánchez and J.-C. Latombe, “On delaying collision checking in PRM planning: Application to multi-robot coordination,” *International Journal of Robotics Research*, vol. 21, no. 1, pp. 5–26, 2002.

[10] S. M. LaValle and J. J. Kuffner, “Rapidly-exploring random trees: Progress and prospects,” in *New Directions in Algorithmic and Computational Robotics*, B. R. Donald, K. Lynch, and D. Rus, Eds., 2001, pp. 293–308.

[11] J. Kim, J. M. Esposito, and V. Kumar, “An RRT-based algorithm for testing and validating multi-robot controllers,” in *Robotics: Science and Systems*, Boston, MA, 2005, pp. 249–256.

[12] J. M. Esposito, J. Kim, and V. Kumar, “Adaptive RRTs for validating hybrid robotic control systems,” in *Workshop on Algo. Found. of Robot.*, Zeist, Netherlands, 2004, pp. 107–132.

[13] 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 Inter. Conf. on Robotics and Automation*, 1998, pp. 630–637.

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

[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] M. Morales, L. Tapia, R. Pearce, S. Rodriguez, and N. M. Amato, “A machine learning approach for feature-sensitive motion planning,” in *Workshop on Algo. Found. of Robot.*, Zeist, Netherlands, pp. 361–376.

[17] P. Cheng, E. Frazzoli, and S. M. LaValle, “Improving the performance of sampling-based planners by using a symmetry-exploiting gap reduction algorithm,” in *IEEE Inter. Conf. on Robotics and Automation*, 2004, pp. 4362–4368.

[18] S. Brin, “Near neighbor search in large metric spaces,” in *Inter. Conf. Very Large Data Bases*, 1995, pp. 574–584.

[19] E. Plaku and L. E. Kavraki, “Quantitative analysis of nearest-neighbors search in high-dimensional sampling-based motion planning,” in *Workshop on Algo. Found. of Robot.*, New York, NY, 2006.