Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence

Robot Motion Planning with Dynamics as Hybrid Search

Erion Plaku
Department of Electrical Engineering and Computer Science
Catholic University of America, Washington DC, 22064

Abstract

This paper presents a framework for motion planning with dynamics as hybrid search over the continuous space of feasible motions and the discrete space of a low-dimensional workspace decomposition. Each step of the hybrid search consists of expanding a frontier of regions in the discrete space using cost heuristics as guide followed by sampling-based motion planning to expand a tree of feasible motions in the continuous space to reach the frontier. The approach is geared towards robots with many degrees-of-freedom (DOFs), nonlinear dynamics, and nonholonomic constraints, which make it difficult to follow discrete-search paths to the goal, and hence require a tight coupling of motion planning and discrete search. Comparisons to related work show significant computational speedups.

Introduction

The objective in motion planning is to compute a motion trajectory from an initial state to a goal region that avoids collisions with obstacles. In addition, motion planning needs to take into account the underlying robot dynamics in order to plan dynamically-feasible motions that the robot can execute in the physical world. Robot dynamics express physical constraints on the feasible motions, such as bounding the velocity and directions of motions, ensuring a minimum turning radius, or keeping the wheels from sliding sideways. Robot dynamics are generally specified as differential equations of the form $\dot{s} = f(s, u)$, which describe how the continuous state s changes as a result of applying control inputs u. As an example, the differential equations of a car-like robot indicate changes in position, orientation, and velocity as a result of setting the acceleration and steering wheel controls.

Robot dynamics pose significant computational challenges to motion planning (Choset et al. 2005). The constraints on the feasible motions make it difficult to find controls that drive the robot to the goal while avoiding collisions. This is made even more challenging by the fact that often the constraints are nonholonomic due to the controllable DOFs being less than the total DOFs. In addition, the continuous state is generally high dimensional in order to model position, orientation, velocity, and other components related to motion. The differential equations are also often nonlinear as the continuous state includes the velocity.

The proposed approach builds on sampling-based motion planning, which has had success in addressing these challenges (Choset et al. 2005; LaValle 2006). The key insight is to search for a solution by selectively sampling and exploring the continuous space of collision-free and dynamically-feasible motions. Starting from the initial state, the search is incrementally expanded as a tree. Each branch is generated by applying a control u to an existing state s in the tree and integrating the differential equations of motion f for a number of steps or until a collision is found. This ensures that each branch corresponds to a collision-free and dynamically-feasible trajectory. The control u is sampled usually at random to expand the search along different directions. The search terminates when the tree reaches the goal region. A solution trajectory is then obtained by concatenating the trajectories associated with the tree branches connecting the initial state to the goal region.

Sampling-based motion planners rely on various heuristics to guide the search. Rapidly-exploring Random Tree (RRT) (LaValle and Kuffner 2001; LaValle 2011) and other successful approaches have used nearest neighbors, distance metrics, probability distributions, reachable sets, transition costs, and other heuristics to guide the search (Hsu et al. 2002; Ladd and Kavraki 2005; Dalibard and Laumond 2008; Jaillet, Cortes, and Simeon 2008; Berenson et al. 2009; Shkolnik, Walter, and Tedrake 2009; Karaman and Frazzoli 2010; Li and Bekris 2011; Şucan and Kavraki 2012; Kiesel, Burns, and Ruml 2012).

Even though considerable progress has been made, it still remains challenging to efficiently plan motions that take into account nonlinear dynamics, nonholonomic constraints, and high-dimensional continuous state spaces. The search in many of these sampling-based approaches has been noted to slow down as the dimensionality or the complexity of the dynamics increase (Choset et al. 2005; LaValle 2011).

To achieve computational efficiency, the proposed approach treats motion planning as hybrid search over the continuous space of feasible motions and the discrete space of a low-dimensional decomposition. This is motivated by Syclop (Plaku, Kavraki, and Vardi 2010), which has been shown to improve over RRT and other sampling-based mo-

tion planners by combining sampling-based motion planning with discrete search. The low-dimensional decomposition in the proposed approach is obtained by a triangulation of the workspace (the environment on which the robot operates). The use of workspace decompositions is motivated by their suitability for problems in mobile robotics, particularly when a challenge for motion planning is in avoiding workspace collisions. The decomposition serves as a simplified abstraction of the overall motion-planning problem that will be used to guide the search more effectively. Each region in the decomposition is associated with a heuristic cost, which is initially based on the shortest-path distance to the goal along the edges of the adjacency graph of the workspace triangulation. The proposed hybrid search starts by rooting a tree at the initial continuous state and adding the corresponding workspace region to the frontier. Each step of the hybrid search consists of selecting a region r from the frontier using cost heuristics as guide followed by sampling-based motion planning to expand the tree of feasible motions in the continuous space from r. Each new region reached by the tree is added to the frontier so that it becomes available for selection in the next expansion step. The heuristic cost associated with r is increased to allow expansions from different regions. The tree and the frontier of regions are expanded in this way until the goal region is reached.

The proposed approach offers several improvements over Syclop approaches (Plaku, Kavraki, and Vardi 2010; Plaku 2012). In particular, Syclop approaches compute at each iteration an entire discrete path from the initial to the goal region and then use sampling-based motion planning to expand the search from regions along this path. Due to collision avoidance and constraints imposed by dynamics, expansion along certain discrete paths may be difficult or even impossible. As the discrete paths also tend to be long, Syclop could waste some valuable computational time before realizing that it needs to abandon the current discrete path, due to lack of progress, and request a new one. In contrast, the proposed approach does not follow an entire discrete path but simply expands the search from one selected region using the cost heuristics as guide. In addition, the proposed approach is easier to implement as it does not require the complex estimates on coverage and free volume used by Syclop.

Decompositions in conjunction with sampling-based motion planning have also been used in (Kiesel, Burns, and Ruml 2012) to bias the sampling in RRT toward regions associated with low cost. Although the approach improves upon RRT, it still inherits its limitations due to the use of a distance metric and expansion from nearest neighbors, which cause the tree to get stuck when dealing with high-dimensional problems with nonlinear dynamics and nonholonomic constraints. As a result, the approach has been applied only to low-dimensional systems with essentially linear dynamics. Discretization of the continuous state space to solve motion-planning problems has also been used in (Likhachev and Stentz 2008; Gochev, Safonova, and Likhachev 2012). The discretization, however, limits the applicability of these approaches to low-dimensional systems and does not allow taking into account complex dynamics.

Experimental results show that the proposed approach considerably improves the computational efficiency over state-of-the-art sampling-based motion planners, such as RRT (LaValle and Kuffner 2001), fRRT (Kiesel, Burns, and Ruml 2012), and Syclop (Plaku, Kavraki, and Vardi 2010; Plaku 2012), in solving high-dimensional problems with nonlinear dynamics and nonholonomic constraints.

Problem Specification

The planner considers a robot and its interactions with the world as a black-box simulator that provides access to the necessary components for motion planning while hiding details that do not affect motion planning. Such approach provides a general formulation that makes it possible to consider a wide class of systems. The simulator is defined as

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

where the new state s_{new} is obtained by applying the control u to the state s for a short time duration dt. The simulator relies on state-of-the-art numerical integration of the differential equations f to compute s_{new}. To ensure accuracy, as advocated in the literature, Runge-Kutta methods with an adaptive step are used for the integration.

The motion-planning problem can now be stated as follows: Given an initial state, a goal region, and a simulator, compute controls such that the resulting trajectory reaches the goal, avoids collisions, and is dynamically feasible.

Example To facilitate presentation, this section provides an example of the robot model used in the paper. A high-dimensional model with nonlinear dynamics and nonholonomic constraints is obtained by attaching several links to resemble a snake (Fig. 1) and modeling the dynamics as a car pulling trailers (adapted from (LaValle 2006, pp. 731)):

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

where x, y, θ_0, v, ψ is the position, orientation, velocity, and steering-wheel angle of the car; θ_i is the orientation of the i-th trailer; N is the number of trailers; $d = 0.05$ is the hitch length; a and ω are the acceleration and steering controls.

Method

The proposed approach simultaneously conducts a sampling-based search over the continuous state space and a discrete search over a workspace decomposition. Pseudocode is provided in Algo. 1.

Workspace Decomposition

The low-dimensional decomposition is obtained by a triangulation of the workspace W. The triangulation treats the obstacles and the goal region r_{goal} as holes inside the workspace bounding box to ensure that the triangles in the decomposition do not intersect with obstacles or the goal region except at the boundaries. The open-source Triangle package (Shewchuk 2002) is used for the computations.

The physical adjacency of the regions in the workspace decomposition is captured by a graph $G = (R, E)$, where

$$R = \{r_1, \ldots, r_m, r_{goal}\}$$

denotes the regions of the workspace decomposition and

$$E = \{(r_i, r_j) : r_i, r_j \in R \text{ share a vertex or an edge}\}$$

Algorithm 1 Pseudocode for the proposed approach

$G = (R, E) \leftarrow$ WORKSPACEDECOMPOSITION(W)
$(\mathit{hcost}(r_1), \ldots, \mathit{hcost}(r_n)) \leftarrow$ HEURISTICCOSTS(G, r_{goal})
$\mathcal{T} \leftarrow$ CREATETREE(s_{init})
INSERT$(Q,$ LOCATEREGION$(s_{\mathit{init}}))$
while TIME$() < t_{\mathit{max}}$ **and** SOLVED$() =$ `false` **do**
  $r \leftarrow$ SELECTREGIONBASEDONCOST(Q)
  $v \leftarrow$ SELECTVERTEX$(\mathit{vertices}(\mathcal{T}, r))$
  $u \leftarrow$ SELECTCONTROL$(\mathit{state}(v))$
  for several steps **do**
    $[s_{\mathit{new}}, \mathit{collision}] \leftarrow$ SIMULATOR$(\mathit{state}(v), u, dt)$
    if $\mathit{collision} =$ `true` **then break** for loop
    $v \leftarrow$ ADDNEWVERTEX$(\mathcal{T}, s_{\mathit{new}}, u, dt)$
    if CONTAINS$(Q, \mathit{region}(v)) =$ `false` **then**
      INSERT$(Q, \mathit{region}(v))$
  $\mathit{hcost}(r) \leftarrow 2\mathit{hcost}(r)$; UPDATE$(Q, r)$

denotes the edges. The proposed approach also relies on a function LOCATEREGION : $W \rightarrow R$, which maps each workspace point to the corresponding region in the decomposition. The function returns an error code if the point is inside an obstacle or falls outside the bounding box of W. LOCATEREGION can run in polylogarithmic time with respect to the number of triangles in the triangulation (de Berg et al. 2008). To facilitate presentation, the shorthand notation LOCATEREGION(s) is used to denote running the function with the position component of the state s as the input.

Heuristic Costs

Each $r \in R$ is associated with a heuristic cost, denoted as $\mathit{hcost}(r)$, which estimates the difficulty of reaching r_{goal} from r. Since $G = (R, E)$ provides a simplified discrete abstraction of the problem, $\mathit{hcost}(r)$ is defined as the length of the shortest path from r to r_{goal} in G, where the cost of each $(r_i, r_j) \in E$ is set to the Euclidean distance between the centroids of r_i and r_j. The heuristic costs are computed all at once by running Dijkstra's shortest-path algorithm with r_{goal} as the source. As discussed later in the section, if the search is expanded from r, then $\mathit{hcost}(r)$ is increased to allow expansion from other regions in later iterations.

Search Data Structures

In the continuous state space S, the search is maintained as a tree $\mathcal{T}$ of continuous states and motion trajectories. In particular, each vertex $v \in \mathcal{T}$ is associated with a collision-free continuous state, denoted as $\mathit{state}(v)$. Each edge $(v_i, v_j) \in \mathcal{T}$ is associated with a collision-free and dynamically-feasible trajectory, denoted as $\mathit{traj}(v_i, v_j)$, which connects $\mathit{state}(v_i)$ to $\mathit{state}(v_j)$. The tree is rooted at the initial state s_{init} and is expanded by adding new vertices and new edges. A solution is found when a vertex v that satisfies the motion-planning goal is added to $\mathcal{T}$. In that case, the trajectory to the goal is obtained by concatenating the trajectories associated with the tree edges connecting the root vertex to v.

The vertices in $\mathcal{T}$ are also grouped according to the regions that they have reached, i.e.,

$$\mathit{vertices}(\mathcal{T}, r) = \{v : v \in \mathcal{T} \wedge \mathit{region}(v) = r\},$$

where $\mathit{region}(v)$ is computed as LOCATEREGION$(\mathit{state}(v))$. To save computation time, these sets are updated on-the-fly as new vertices are added to $\mathcal{T}$. In particular, when a vertex v is added to $\mathcal{T}$, it is also added to $\mathit{vertices}(\mathcal{T}, \mathit{region}(v))$.

In the discrete space, the search is maintained as a region frontier, denoted by Q, which includes all the regions that have been reached by $\mathcal{T}$, i.e.,

$$Q = \{r : r \in R \wedge |\mathit{vertices}(\mathcal{T}, r)| > 0\}.$$

The frontier is also updated on-the-fly by adding each region to Q as soon as it is reached by $\mathcal{T}$, if not already there. A hash map data structure is used to determine in almost constant time if a region is already in the frontier Q.

Overall Search

The search starts by adding the initial state s_{init} to $\mathcal{T}$. The corresponding region r_{init}, which is computed as LOCATEREGION(s_{init}), is added to Q. Each iteration consists of selecting a region $r \in Q$ from which to expand the search followed by sampling-based motion planning to expand $\mathcal{T}$ from $\mathit{vertices}(\mathcal{T}, r)$.

A probabilistic scheme is used to select a region r from Q with probability depending on $\mathit{hcost}(r)$, i.e.,

$$\mathit{prob}(r) = \frac{1/\mathit{hcost}(r)}{\sum_{r' \in Q} 1/\mathit{hcost}(r')}. \quad (1)$$

Such scheme selects more frequently regions associated with low costs, which provides the greedy component to the search. At the same time, each region has a nonzero probability of being selected, which provides the methodical component necessary to ensure that the search does not get stuck due to greedy choices. From an implementation perspective, the probabilistic selection can run in $O(\log(|Q|))$ time by arranging the regions in Q as leaf nodes in a self-balancing binary search tree. Each node b in the binary tree contains the sum of the costs of its children b_{left} and b_{right}. To select a region, a number w is generated uniformly at random from 0 to c, where c is the cost at the top of the binary tree. The selection moves recursively to the left or to the right subtree until it reaches a leaf node. If b is a leaf node, then the region associated with b is selected. Otherwise, if $w < \mathit{cost}(b_{\mathit{left}})$, the selection continues recursively onto b_{left} with w as the input number. If $w \geq \mathit{cost}(b_{\mathit{left}})$, the selection continues recursively onto b_{right} with $w - \mathit{cost}(b_{\mathit{left}})$ as the input number.

After selecting a region r from Q, sampling-based motion planning is invoked to expand a collision-free and dynamically-feasible trajectory from $\mathit{vertices}(\mathcal{T}, r)$. The sampling-based motion planner selects a vertex v from $\mathit{vertices}(\mathcal{T}, r)$ and then samples a control input u. The vertex selection as well as control sampling are done uniformly at random, as it is commonly the case in sampling-based motion planning. Other selection and sampling strategies are possible, as discussed in (Choset et al. 2005; LaValle 2006). The tree $\mathcal{T}$ is expanded from the vertex v by applying the control input u to $\mathit{state}(v)$ for several steps or until a collision is found. This gives rise to a collision-free and dynamically-feasible trajectory, which is computed by integrating the differential equations of motion. To ensure accuracy, as advocated in the literature, Runge-Kutta methods with an adaptive step are used for the integration. Intermediate collision-free states are added as new vertices to $\mathcal{T}$.

If a new region r_{new} is reached, then r_{new} is added to Q to give the approach the possibility to expand the search from new regions. The search continues to expand Q and $\mathcal{T}$ until r_{goal} is reached. In that case, the solution trajectory is obtained by concatenating the collision-free and dynamically-feasible trajectories associated with the tree edges that connect the root vertex to a vertex v in r_{goal}.

After each motion-planning expansion, hcost(r) is increased in order to provide the approach with the flexibility to expand the search from other regions. This is essential to ensure an effective exploration, since, due to collision avoidance and dynamics, it may

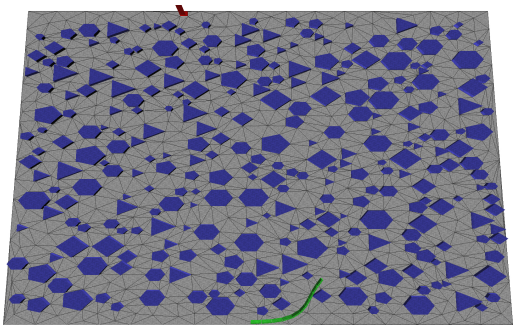
be difficult or even impossible to expand the search from a particular region r.

Probabilistic Completeness

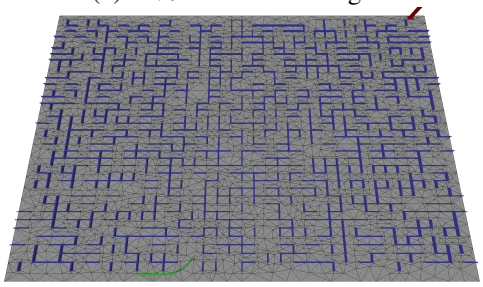
Due to space limitations, this section provides only an outline of the proof of probabilistic completeness for the proposed approach. The proof is based on the theoretical framework developed in (**?**; Ladd and Kavraki 2005), which shows that a motion planner that produces any random walk in S_{free} is probabilistically complete. To achieve the random-walk criterion, it suffices to show that, as time tends to ∞, every state in $\mathcal{T}$ is selected infinitely often and that all possible control inputs can be eventually applied from each state.

The condition on control inputs is satisfied by the uniformly-at-random selection strategy employed in the proposed approach. To show that each state in $\mathcal{T}$ is selected infinitely often, first note that each region $r \in Q$ has a nonzero probability of being selected at each iteration (see Algo 1:6 and Equation 1). Moreover, a state is selected from r uniformly at random. As a result, each state in $\mathcal{T}$ has a nonzero probability of being selected at each iteration, which ensures that it will be selected infinitely often as time tends to ∞.

Experiments and Results



(b) 26% obstacle coverage



(d) 48×48 maze

Figure 1: Instances of the "random obstacles" and "random mazes" benchmarks. Obstacles shown in blue, robot in green, goal in red, triangulation in gray.

Experimental validation is provided by comparing the proposed approach to related work. The experiments vary the benchmark type and difficulty as well as the DOFs of the robot to show how the methods scale.

Benchmarks The "random obstacles" and "random mazes" benchmarks are used in the experiments as they provide challenging motion-planning problems. Fig. 1 shows some examples. The benchmarks are parametrized in order to generate workspaces of increasing difficulty, where the robot has to wiggle its way through numerous obstacles and narrow passages to reach the goal. In addition to collision avoidance, the differential constraints imposed by the robot dynamics make these problems even more challenging.

The "random obstacles" benchmark is parametrized by the percentage p of the workspace area covered by obstacles. Random obstacles are added inside the workspace until the percentage p is reached. The experiments use random obstacles with coverage percentages varying from 20% to 32%. The geometry of each robot link is set to a rectangle with dimensions 1.2 and 0.2.

The "random mazes" benchmark is parametrized by the number of dimensions p. A $p \times p$ maze is generated using a randomized version of the Kruskal's algorithm. Additional passages are created by removing at the end 20% of the walls, selected at random. The experiments use random mazes with dimensions varying from 32×32 to 64×64. In the case of random mazes, the geometry of each robot link is set to a rectangle with dimensions 0.75 and 0.125.

Measuring Computational Efficiency Experiments vary the benchmark type, the parameter value associated with each benchmark, and the DOFs of the robot. For a fixed benchmark type, parameter value, and DOFs, 30 random instances are generated. In each instance, the robot is placed randomly at either the bottom or the top of the workspace while the goal region is placed randomly at the opposite side so that the robot has to move from one end to the other end of the workspace. Each method is run on each problem instance until the problem is solved or a timeout of 200s is reached. Due to the probabilistic nature of the methods, to avoid the influence of outliers, the 5 lowest and the 5 highest running times are removed, and the computational efficiency of a method for a fixed benchmark type, parameter value, and DOFs is measured as the average of the remaining times. All the experiments are run on an Intel Core 2 Duo machine (CPU: P8600 at 2.40GHz, RAM: 8GB) using Ubuntu 12.10. Code is compiled with GNU g++-4.7.2.

Methods used in the Comparisons The approach is first compared to RRT (LaValle and Kuffner 2001; LaValle 2011), which is one of the most successful sampling-based motion planners. The RRT implementation, as advocated in literature, uses the connect version, which extends each trajectory until it reaches the sampled state or finds a collision. In addition, goal bias is used to guide RRT toward the goal. Efficient data structures are used for nearest neighbors (Beygelzimer, Kakade, and Langford 2006).

Comparisons are included with a recent version of RRT, named fRRT, which biases the sampling toward regions associated with low cost (Kiesel, Burns, and Ruml 2012). The implementation follows the details in the technical report.

The approach is also compared to Syclop (Plaku, Kavraki, and Vardi 2010; Plaku 2012), which introduced the idea of using discrete search to guide sampling-based motion planning. Syclop was shown to solve challenging motion-planning problems with dynamics significantly faster than other state-of-the-art sampling-based motion planners. The experiments use the most recent version (Plaku 2012).

Results as Function of Benchmark Difficulty Fig. 2 provides a summary of the results when varying the benchmark difficulty. As the results show, RRT is able to efficiently solve the initial "random obstacles" problems. As the coverage percentage increases, however, the running time of RRT increases rapidly, eventually timing out. As RRT is guided by nearest neighbors, it becomes increasingly difficult to find new ways to expand the search toward the goal. After an initial growth, RRT gets stuck in the narrow passages. These problems become more pronounced in the case of the "random mazes" benchmark. By expanding the search from the nearest vertex to a randomly sampled state, RRT is likely to encounter maze walls, and thus get stuck trying to find a way

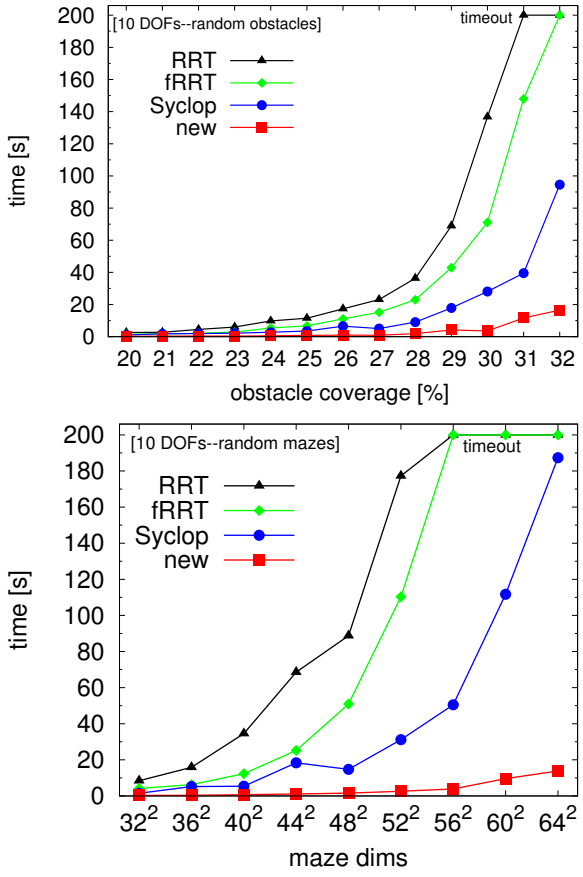


Figure 2: Impact of the benchmark difficulty on the computational efficiency. The label new indicates the results of the proposed approach.

out of the maze. Recall that 20% of the maze walls were knocked down when creating the benchmark instances in order to provide alternative ways out of the maze. Nevertheless, RRT remains stuck in the maze and the running time increases rapidly as the instances become more challenging.

fRRT (Kiesel, Burns, and Ruml 2012) improves upon RRT. By biasing the sampling toward regions with low-cost, fRRT more effectively guides the search toward the goal. fRRT, however, still inherits the problems of RRT in using a distance metric and nearest neighbors to expand the search. As a result, the exploration in fRRT starts being less effective as the problem difficulty increases.

Syclop considerably improves over the running time of RRT and fRRT. Similar improvements over RRT and other state-of-the-art sampling-based approaches were also shown in (Plaku, Kavraki, and Vardi 2010; Plaku 2012). By using discrete search to guide sampling-based motion planning, Syclop is able to more efficiently solve the problem instances. As the benchmark difficulty increases, Syclop starts to slow down but not as rapidly as RRT or fRRT. Due to collision avoidance and constraints imposed by dynamics, expansion along certain discrete paths may be difficult or even impossible. As the discrete paths also tend to be long, Syclop wastes some valuable computational time before abandoning the current discrete path, due to lack of progress, and requesting a new one. These problems become more pronounced in the case of the "random mazes" benchmarks, as many discrete plans, due to collision avoidance and differential constraints, cannot be easily followed.

Fig. 2 shows that the proposed approach yields significant computational speedups over RRT, fRRT, and Syclop. The improvements become more pronounced as the difficulty of the problem is increased.[1] By using cost heuristics, the approach is often able to expand the search rapidly towards the goal. In cases where sampling-based motion planning is not able to expand the search to a particular region in the frontier, its cost is increased to provide the approach with the flexibility to expand the search from new regions. Moreover, rather than following an entire discrete path as Syclop does, the approach simply expands the search from the selected region. This gives the approach greater flexibility to explore new regions and discover new ways to reach the goal.

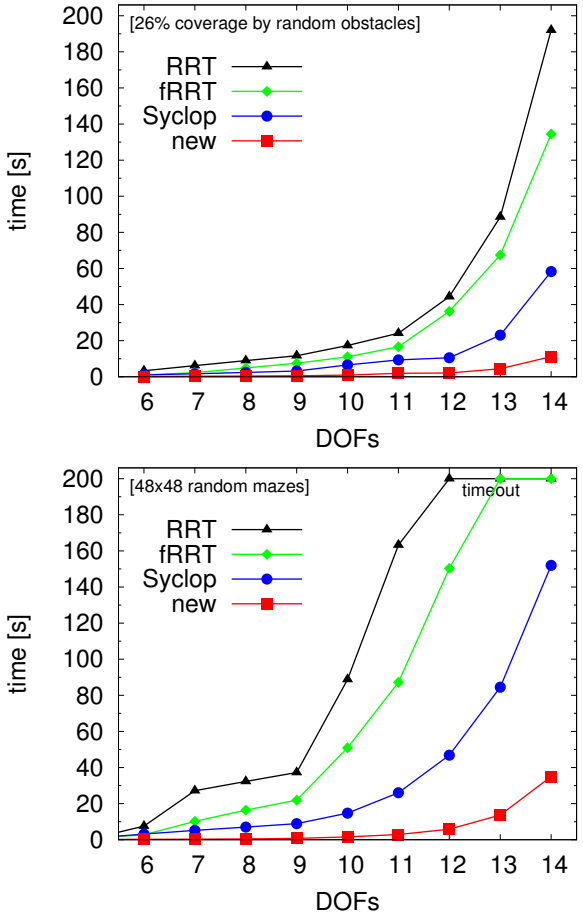


Figure 3: Impact of the problem dimensionality on computational efficiency. The label new indicates the results of the proposed approach.

Results as Function of Problem Dimensionality Fig. 3 provides a summary of the results when increasing DOFs. As more and more links are added to the robot, it becomes increasingly difficult to navigate through narrow passages and reach the goal. The

[1]The proposed approach is faster than RRT, fRRT, and Syclop even for the simpler problem instances, but, due to the scale of the graph, those results are harder to see. As an example, for the problem with 26% coverage and 6DOFs, the running times of the proposed approach, RRT, fRRT, and Syclop are 0.14, 3.30s, 1.15s, and 1.05s, respectively.

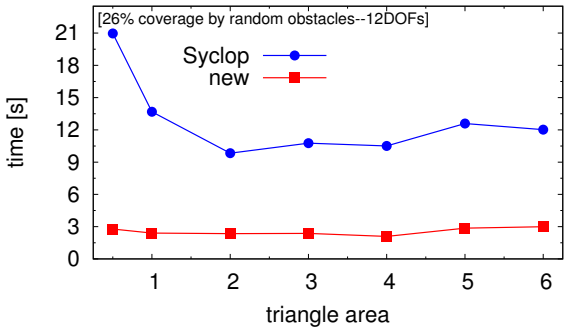


Figure 4: Impact of the workspace triangulation granularity on the computational efficiency.

results in Fig. 3 are presented for the "random obstacles" benchmark with 26% obstacle coverage and the 48×48 "random mazes" benchmark. Similar trends are observed for the other benchmark instances but are not shown here due to space constraints.

As shown in Fig. 3, the running time of RRT increases rapidly with the number of DOFs and times out on the high-dimensional problems. fRRT improves upon RRT, but still has difficulty solving challenging problems and times out in several instances. Syclop is faster than RRT and fRRT, but the running time still starts to slow down as more and more DOFs are added to the robot. The proposed approach yields significant speedups over RRT, fRRT, and Syclop, and efficiently solves even the high-dimensional problems.

Results as Function of Decomposition Granularity Fig. 4 summarizes the results when varying the granularity of the workspace triangulation. The results show that the proposed approach and Syclop work well for a wide range of triangulations. As the triangulation becomes more fine-grained, as also noted in (Plaku, Kavraki, and Vardi 2010), the running time of Syclop starts to increase since it becomes more difficult to follow the discrete plans. The proposed approach does not suffer from this problem as it uses the region frontier to determine where to expand the search.

Discussion

This paper presented a general framework for motion planning with dynamics as hybrid search over the continuous space of feasible motions and the discrete space of a low-dimensional decomposition. The hybrid search expanded a frontier of decomposition regions using sampling-based motion planning to reach regions in the frontier. Cost heuristics based on a low-dimensional decomposition were used to effectively guide the search toward the goal. Comparisons to related work showed significant improvements.

The proposed approach opens up several venues for further research. In particular, the approach could benefit from more advanced discrete search techniques and improved heuristics to more effectively couple discrete search and sampling-based motion planning. Pruning or branch-and-bound techniques could be used in connection with sampling-based motion planning to determine tree vertices and regions from which to expand the search.

Another direction is to investigate optimality. The focus of this work was on computational efficiency, as it is generally the case in sampling-based motion planning. Recent analysis has shown that rewiring RRT connections leads to an optimal algorithm, referred to as RRT* (Karaman and Frazzoli 2010). Rewiring, however, significantly increases the computational cost, rendering RRT* impractical in the case of motion planning with complex dynamics where even RRT has difficulty finding a solution. Moreover, rewiring causes gaps in the solution since due to constraints imposed by dynamics and collision avoidance it is not always possible to steer the system to a given state. Closing the gaps significantly increases the computational cost (**?**). The proposed approach tends to produce short solutions due to the cost heuristics it uses to guide the search. It may be possible to add a $gcost(r)$ component, which expresses the lowest cost path to r, which would then run the discrete search in an A* formulation, i.e., $gcost(r) + hcost(r)$. There are, however, some challenges. In particular, it is important to maintain the computational efficiency since $gcost(r) + hcost(r)$ could slow the search as it will penalize long solutions, which may be easier to obtain. We will tackle these challenges as part of our future work.

References

Berenson, D.; Srinivasa, S.; Ferguson, D.; Romea, A. C.; and Kuffner, J. 2009. Manipulation planning with workspace goal regions. In *IEEE International Conference on Robotics and Automation*, 618–624.

Beygelzimer, A.; Kakade, S.; and Langford, J. 2006. Cover trees for nearest neighbor. In *Interantional Conference on Machince Learning*, 97–104.

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

Şucan, I. A., and Kavraki, L. E. 2012. A sampling-based tree planner for systems with complex dynamics. *IEEE Transactions on Robotics* 28(1):116–131.

Dalibard, S., and Laumond, J.-P. 2008. Control of probabilistic diffusion in motion planning. In *International Workshop on Algorithmic Foundations of Robotics*, volume 57 of *Springer Tracts in Advanced Robotics*. 467–481.

de Berg, M.; Cheong, O.; van Kreveld, M.; and Overmars, M. H. 2008. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 3 edition.

Gochev, K.; Safonova, A.; and Likhachev, M. 2012. Planning with adaptive dimensionality for mobile manipulation. In *IEEE International Conference on Robotics and Automation*, 2944–2951.

Hsu, D.; Kindel, R.; Latombe, J. C.; and Rock, S. 2002. Randomized kinodynamic motion planning with moving obstacles. *International Journal of Robotics Research* 21(3):233–255.

Jaillet, L.; Cortes, J.; and Simeon, T. 2008. Transition-based RRT for path planning in continuous cost spaces. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2145–2150.

Karaman, S., and Frazzoli, E. 2010. Optimal kinodynamic motion planning using incremental sampling-based methods. In *IEEE Conference on Decision and Control*, 7681–7687.

Kiesel, S.; Burns, E.; and Ruml, W. 2012. Abstraction-guided sampling for motion planning. In *Symposium on Combinatorial Search*. also as UNH CS Technical Report 12-01.

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

LaValle, S. M., and Kuffner, J. J. 2001. Randomized kinodynamic planning. *International Journal of Robotics Research* 20(5):378–400.

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

LaValle, S. M. 2011. Motion planning: The essentials. *IEEE Robotics & Automation Magazine* 18(1):79–89.

Li, Y., and Bekris, K. E. 2011. Learning approximate cost-to-go metrics to improve sampling-based motion planning. In *IEEE International Conference on Robotics and Automation*, 4196–4201.

Likhachev, M., and Stentz, A. 2008. R* search. In *National Conference on Artificial Intelligence*, volume 1, 344–350.

Plaku, E.; Kavraki, L. E.; and Vardi, M. Y. 2010. Motion planning with dynamics by a synergistic combination of layers of planning. *IEEE Transactions on Robotics* 26(3):469–482.

Plaku, E. 2012. Guiding sampling-based motion planning by forward and backward discrete search. In *International Conference on Intelligent Robots and Applications*, volume 7508 of *Lecture Notes in Artificial Intelligence*. 289–300.

Shewchuk, J. R. 2002. Delaunay refinement algorithms for triangular mesh generation. *Computational Geometry: Theory and Applications* 22(1-3):21–74.

Shkolnik, A.; Walter, M.; and Tedrake, R. 2009. Reachability-guided sampling for planning under differential constraints. In *IEEE International Conference on Robotics and Automation*, 2859–2865.