2008 IEEE International Conference on
Robotics and Automation
Pasadena, CA, USA, May 19-23, 2008

Impact of Workspace Decompositions on Discrete Search Leading Continuous Exploration (DSLX) Motion Planning

Erion Plaku Lydia E. Kavraki Moshe Y. Vardi

Abstract— **We have recently proposed DSLX, a motion planner that significantly reduces the computational time for solving challenging kinodynamic problems by interleaving continuous state-space exploration with discrete search on a workspace decomposition. An important but inadequately understood aspect of DSLX is the role of the workspace decomposition on the computational efficiency of the planner. Understanding this role is important for successful applications of DSLX to increasingly complex robotic systems.**

This work shows that the granularity of the workspace decomposition directly impacts computational efficiency: DSLX is faster when the decomposition is neither too fine- nor too coarse-grained. Finding the right level of granularity can require extensive fine-tuning. This work demonstrates that significant computational efficiency can instead be obtained with no fine-tuning by using conforming Delaunay triangulations, which in the context of DSLX provide a natural workspace decomposition that allows an efficient interplay between continuous state-space exploration and discrete search. The results of this work are based on extensive experiments on DSLX using grid, trapezoidal, and triangular decompositions of various granularities to solve challenging first and second-order kinodynamic motion-planning problems.

I. INTRODUCTION

Motion planning is becoming increasingly relevant in transportation, exploration, and search and rescue missions [1]–[6]. While avoiding collisions when reaching a desired goal, an already challenging task in path planning, a robot deployed in realistic settings must additionally satisfy kinodynamic constraints that further restrict its possible motions. While many methods have made significant progress in taking into account kinodynamic constraints [1], [2], most notably Rapidly-exploring Random Tree (RRT) [4] and Expansive Space Tree (EST) [5], kinodynamic motion planning remains an active area of research.

Our recently proposed motion planner DSLX [7] has been shown to significantly reduce the computational time for solving challenging kinodynamic problems. The planner has been successfully applied to several second-order robotic models. DSLX represents a new class of planners that combines in novel ways continuous state-space exploration with discrete search on a workspace decomposition. The idea of using decompositions appears early in motion planning literature. Key theoretical results were obtained using configuration-space decompositions [8]. Recently workspace decompositions have been used in the Probabilistic RoadMap (PRM) method [9] to improve its sampling [10]–[14]. A survey of decomposition methods in motion planning can be found in [1].

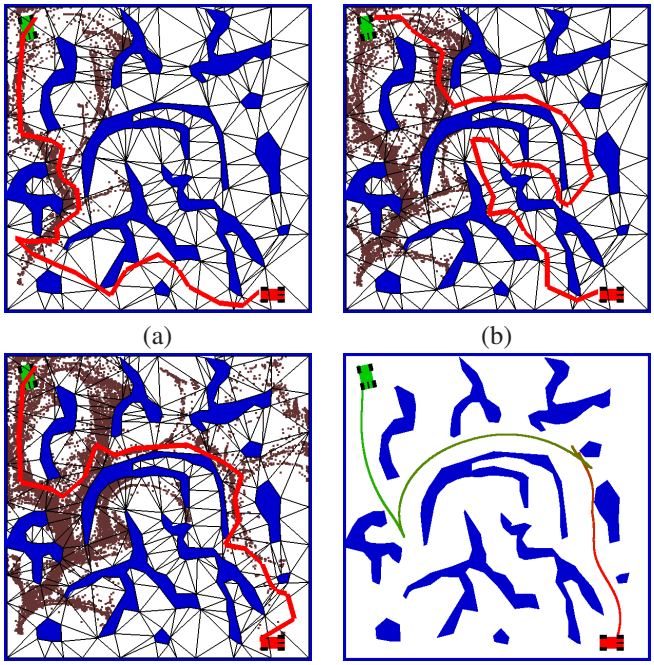


Fig. 1. DSLX solving a motion planning query for a second-order kinodynamic car. (a-c) Current lead is in red. States of the exploration tree projected onto the workspace are shown as brown dots. The lead guides the exploration and adapts to new information. This flexible interplay between discrete search and continuous state-space exploration takes place in a conforming Delaunay triangulation of the workspace. (d) Solution trajectory.

A critical difference of DSLX over related workspace decomposition methods is that these methods are primarily developed for geometric path planning while DSLX is designed to take advantage of workspace decompositions for kinodynamic motion planning [7] through a novel interplay between continuous state-space exploration and discrete search. At each iteration, the discrete search computes a lead, a sequence of decomposition regions that is estimated to be useful for advancing the exploration toward the goal. The continuous state-space exploration uses the current lead to extend a tree along the decomposition regions specified by the lead. Information such as coverage and exploration time is fed back from the continuous state-space exploration to the discrete search to improve the lead for the next iteration. This interaction provides DSLX with the flexibility to extend the tree along useful directions while able to radically change

Work on this paper has been supported in part by NSF CNS 0615328 (EP, LEK, MYV), NSF 0713623 (EP, LEK), a Sloan Fellowship (LEK), and NSF CCF 0613889, CCF 0728882 (MYV). Experiments were run 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

direction if information from the exploration suggests other useful leads, as shown in Fig. 1. It is this flexible interplay that takes place on a workspace decomposition between discrete search and continuous state-space exploration that enables DSLX to achieve significant speedups when compared to other motion planners such as RRT and EST [7].

An important but inadequately understood aspect of DSLX is the impact of the workspace decomposition. Understanding this impact is essential for successfully applying DSLX to increasingly challenging and realistic problems.

The main contribution of this work is a detailed study of the impact of different workspace decompositions on DSLX and the identification of a workspace decomposition that is particularly well-suited for DSLX. Our first implementation of DSLX in [7] relied on a fixed-size grid decomposition. This work shows that, on a grid decomposition, DSLX is faster when the grid is neither too fine- nor too coarse-grained. We also study the impact of trapezoidal and triangular decompositions, and similarly to our findings for grid decompositions, this work shows that DSLX is computationally more efficient when the decomposition granularity is balanced between fine- and coarse-grained.

Fine-tuning the decomposition to find the right granularity could increase the computational efficiency but can require extensive efforts. This work demonstrates that significant computational efficiency can instead be obtained with no fine-tuning by using conforming Delaunay triangulations [15]. Extensive experiments on several first and second order kinodynamic robot models operating in various workspaces show that conforming Delaunay triangulations in the context of DSLX provide a natural decomposition of the workspace that allows a remarkably efficient interplay between the discrete search and continuous state-space exploration. DSLX is generally one order of magnitude more efficient when using conforming Delaunay triangulations instead of the various grid, triangular, and trapezoidal workspace decompositions.

II. DSLX

A detailed description of DSLX appears in [7]. This section provides a summary of DSLX and describes several modifications to the DSLX algorithm to make the interplay between the discrete search and continuous state-space exploration even more efficient. Pseudocode is given in Algorithm 1. The lead computation occurs in line 6 and is described in section II-A. The continuous state-space exploration occurs in line 7 and is described in section II-B.

A. Lead

The objective of the discrete search is to compute at each iteration a lead (line 6), a sequence of decomposition regions that is estimated to be useful for advancing the continuous state-space exploration toward the goal. Let $R = \{R_1, R_2, \ldots, R_n\}$ denote the workspace decomposition into regions. DSLX associates a weight w_i with each decomposition region R_i. The weight w_i is a running estimate on the usefulness of including R_i in the current lead and is computed based on information gathered during each exploration

Algorithm 1 Pseudocode for DSLX

Input:
$\mathcal{W}$, geometric description of the workspace
s, g, initial and goal specifications
$t_{\max} \in \mathbb{R}^{>0}$, upper bound on computation time
$t_e \in \mathbb{R}^{>0}$, short time allocated to each exploration step

Output: A solution trajectory or NIL if no solution is found

1: StartClock
2: $G = (V, E) \leftarrow$ WorkspaceDecomposition($\mathcal{W}$)
3: InitExplorationEstimates(G)
4: $\mathcal{T} \leftarrow$ exploration tree rooted at s
5: **while** ElapsedTime $< t_{\max}$ **do**
6: $\quad[\mathrm{R}_{i_1}, \ldots, \mathrm{R}_{i_n}] \leftarrow$ Lead(G, s, g)
7: $\quad$Explore($\mathcal{T}, [\mathrm{R}_{i_1}, \ldots, \mathrm{R}_{i_n}], t_e$)
8: $\quad$**if** a solution is found **then return** solution trajectory
9: **return** NIL

of R_i as $w_i = t_i^{\alpha} / \left(\mathrm{cov}_i^{\beta} \mathrm{vol}_i^{\gamma} \right)$, where t_i is the total time spent exploring R_i; cov_i is an estimate on the exploration coverage of R_i computed as in [7]; vol_i is the volume of R_i; and α, β, γ are normalization constants. Usefulness is indicated by a lower weight, i.e., R_i is considered useful when R_i has a large volume and the exploration covers large parts of R_i in a short amount of time.

The physical adjacency of the decomposition regions is represented in a graph $G = (V, E)$. For each R_i there is a corresponding vertex $v_i \in V$. Similarly, for any two neighboring decomposition regions R_i and R_j there is an edge $(v_i, v_j) \in E$. Let $v(s), v(g) \in V$ be the two vertices whose corresponding decomposition regions are associated with the initial and goal states, s and g, respectively. A lead is computed by searching G for sequences of edges from $v(s)$ to $v(g)$. The usefulness of each lead for advancing the exploration toward the goal is estimated based on the weights $w_{ij} = w_i * w_j$ associated with each edge $(v_i, v_j) \in E$. The edge $(v_i, v_j) \in E$ is thus considered useful when the corresponding regions R_i and R_j are also considered useful. As in [7], the lead is computed more frequently as the shortest path according to the weights w_{ij} using A* or Dijkstra's shortest-path algorithm. As suggested in [7], random leads are also used, although infrequently, as a way to correct for errors inherent with the estimates and ensure that each possible lead is selected with non-zero probability.

B. Explore

The exploration starts by rooting a tree $\mathcal{T}$ at the initial state s (line 4). The objective during each exploration step (line 7) is to extend $\mathcal{T}$ along the decomposition regions specified by the current lead. Let R_{avail} include all the decomposition regions in the current lead as well as their neighbors. Let R_{use} include all the decomposition regions in R_{avail} that have been reached by states in $\mathcal{T}$. The exploration is an iterative process that lasts for at most t_e seconds. While there is time remaining, a decomposition region R_j is selected from R_{use} with probability $w_{\exp}(j) / \sum_{R_k \in R_{\mathrm{use}}} w_{\exp}(k)$, where $w_{\exp}(j) = \mathrm{vol}_j^{\gamma} / \left(t_j^{\alpha} \mathrm{cov}_j^{\beta} \right)$. This selection scheme gives priority to those decomposition regions in R_{use} that have large volume and low coverage and have not been frequently explored in the past. States associated with R_j are put into

different bins depending on their (x, y) position, similar to SBL [6]. A state s is selected from R_j for propagation by first selecting a bin uniformly at random and then picking a state uniformly at random from the bin. If the propagation is successful a new state s_{new} and the edge connecting s to s_{new} is added to $\mathcal{T}$. The state s_{new} is also added to the appropriate decomposition region R_k. If R_k is in R_{avail} but not in R_{use}, then R_k is added to R_{use}. Thus, when the exploration tree reaches decomposition regions specified by the current lead or their neighbors, they become available for selection during the next iteration of the exploration step.

The interplay between the discrete search and continuous state-space exploration, summarized in this section, is shown in [7] to allow DSLX to obtain significant computational speedups of up to two orders of magnitude when compared to other tree-based planners, e.g., RRT and EST, for solving challenging kinodynamic motion-planning problems.

III. Experiments and Results

Experiments on this work test the impact of different workspace decompositions on DSLX for solving kinodynamic motion-planning problems. Experiments involve first and second-order models of cars, unicycles, and differential drives moving in unstructured and large environments, since the design of DSLX [7] was motivated by such problems. Implementations are based on the OOPSMP framework [16]. Experiments are run on Rice Cray XD1 ADA and PBC clusters. Each processor runs at 2.2GHz and has 2GB RAM.

A. Kinodynamic Robot Models

Detailed descriptions of the robot models can be found in [1], [2]. Bounds on controls and states are empirically determined based on the workspaces used for the experiments. In all cases, (x, y, θ) denotes the configuration.

1) Kinematic Car (KCar): $\dot{x} = u_0 \cos(\theta); \dot{y} = u_0 \sin(\theta); \dot{\theta} = u_0 \tan(u_1)/L$, where L is the distance between the front and rear axles; $|u_0| \leq 3m/s$; and $|u_1| \leq 35°$.

2) Smooth Car (SCar): $\dot{x} = v\cos(\theta)$; $\dot{y} = v\sin(\theta)$; $\dot{\theta} = v\tan(\phi)/L$; $\dot{v} = u_0$; $\dot{\phi} = u_1$, where v is the velocity; ϕ is the steering angle; $|u_0| \leq 0.8m/s^2$; and $|u_1| \leq 20°/s$.

3) Kinematic Unicycle (KUni): $\dot{x} = u_0 \cos(\theta); \dot{y} = u_0 \sin(\theta); \dot{\theta} = u_1$, where $|u_0| \leq 3m/s$; and $|u_1| \leq 40°/s$.

4) Smooth Unicycle (SUni): $\dot{x} = v\cos(\theta)$; $\dot{y} = v\sin(\theta)$; $\dot{\theta} = \omega$; $\dot{v} = u_0$; $\dot{\omega} = u_1$, where v and ω are the translational and rotational velocities; $|u_0| \leq 0.3m/s^2$; and $|u_1| \leq 10°/s^2$.

5) Kinematic Differential Drive (KDDrive): $\dot{x} = ru_0\cos(\theta)$; $\dot{y} = ru_0\sin(\theta)$; $\dot{\theta} = ru_1/L$, where r is the wheel radius; and L is the length of the axis connecting the wheel centers. Controls u_0 and u_1 are obtained from transforming the controls to the left and right wheels as described in [2] and are restricted to $|u_0| \leq 3m/s$ and $|u_1| \leq 40°/s$.

6) Smooth Differential Drive (SDDrive): $\dot{x} = 0.5r(\omega_\ell + \omega_r)\cos(\theta)$; $\dot{y} = 0.5r(\omega_\ell + \omega_r)\sin(\theta)$; $\dot{\theta} = r(\omega_r - \omega_\ell)/L$; $\dot{\omega_\ell} = u_0$; $\dot{\omega_r} = u_1$, where ω_ℓ and ω_r are the rotational velocities of the left and right wheels; r is the wheel radius; L is the length of the axis connecting the wheel centers; $|u_0| \leq 10°/s^2$; and $|u_1| \leq 10°/s^2$.

B. Benchmarks

Workspaces used in the experiments are shown in Fig. 2(a) and are designed to vary in type and difficulty and provide representative problems for kinodynamic motion planning. Queries are generated at random in places similar to the queries illustrated in Fig. 2(a). Solutions to these queries require the robot to avoid obstacles, frequently change directions, make sharp turns, go through narrow passages and tunnels of varying width and length. Each workspace has unit dimensions (1m=0.05units). The body length and width of the car are set to 0.04 and 0.02; body length and wheel radius of the differential drive are set to 0.04 and 0.01; body length and width of the unicycle are set to 0.04 and 0.03.

C. Workspace Decompositions

We conducted experiments using grid, triangular, and trapezoidal decompositions of various granularities.

1) Grid Decompositions: We used grids with 1×1, 2×2, 4×4, 8×8, 16×16, 32×32, 64×64, and 128×128 cells. Fig. 2(b) provides an illustration.

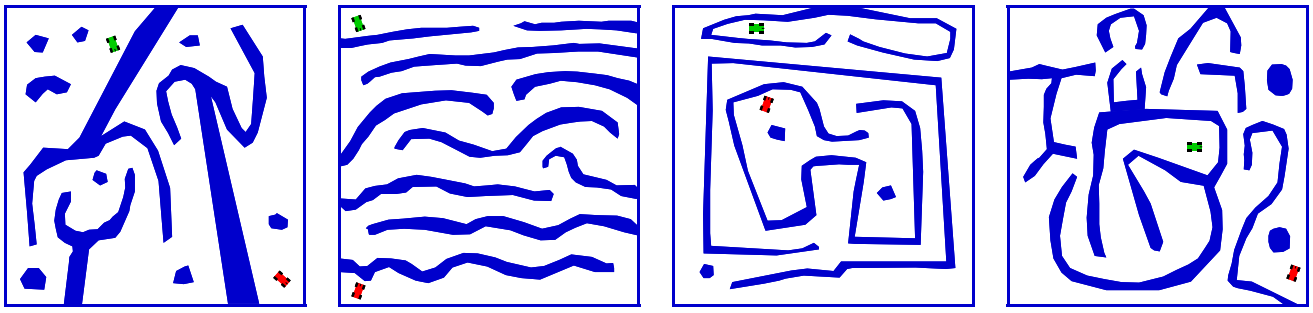
2) Triangular Decompositions: Different triangulations were obtained by varying the triangle area. Fig. 2(c) shows an illustration. Triangulation T1 is obtained by using Seidel's algorithm as implemented in [17]. It is a coarse triangulation and consists primarily of long and thin triangles. Triangulations T2, T3, T4 are computed using the industrial-strength package Triangle [15] and are obtained by requiring the minimum angle in each triangle to be at least $20°$ and the maximum area of each triangle in T2, T3, and T4 to be at most 0.01, 0.0005, and 0.0002, respectively. Such triangulations are commonly used in mesh generations.

3) Trapezoidal Decompositions: Trapezoidal decompositions are illustrated in Fig. 2(d) and are computed using Seidel's algorithm as implemented in [17].

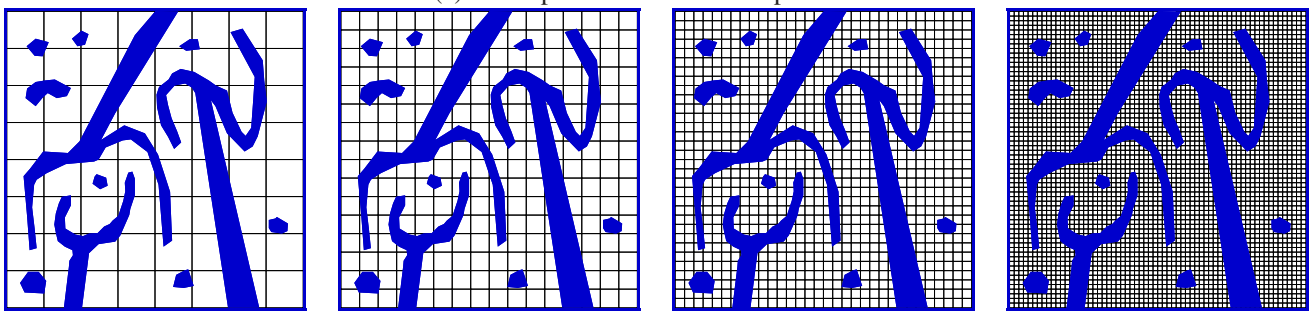
4) Conforming Delaunay Triangulations: Conforming Delaunay triangulations have been widely used in computational geometry and are similar to Delaunay triangulations for a set of points, which maximize the minimum angle among all possible triangulations, but could potentially differ in some places to take into account polygonal edge constraints by adding additional vertices [15]. Although in some theoretical pathological cases $O(n^3)$ new vertices are required, in practice the bound is linear [18]. Fig. 2(e) illustrates the conforming Delaunay triangulations as computed by the industrial-strength package Triangle [15].

D. Results

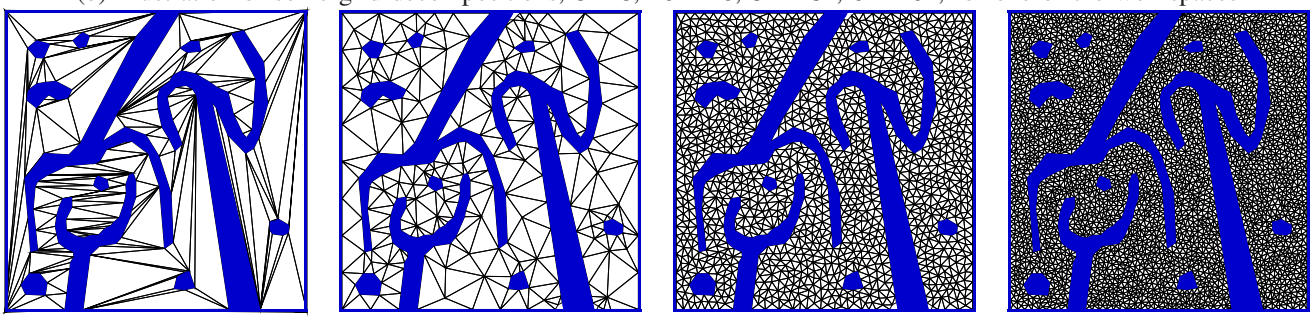
For each combination of workspace, workspace decomposition, and kinodynamic model, DSLX solves 30 queries generated as described in section III-B. The computational efficiency of DSLX for a given combination is measured as the average time to solve the queries after dropping the four lowest and highest times. A timeout of 800s is imposed for a given query. In each DSLX run, the exploration time for each step is set to $t_e = 0.01$s and the normalization constants are set to $\alpha = 4.0$, $\beta = 2.0$, and $\gamma = 2.0$ (see section II).



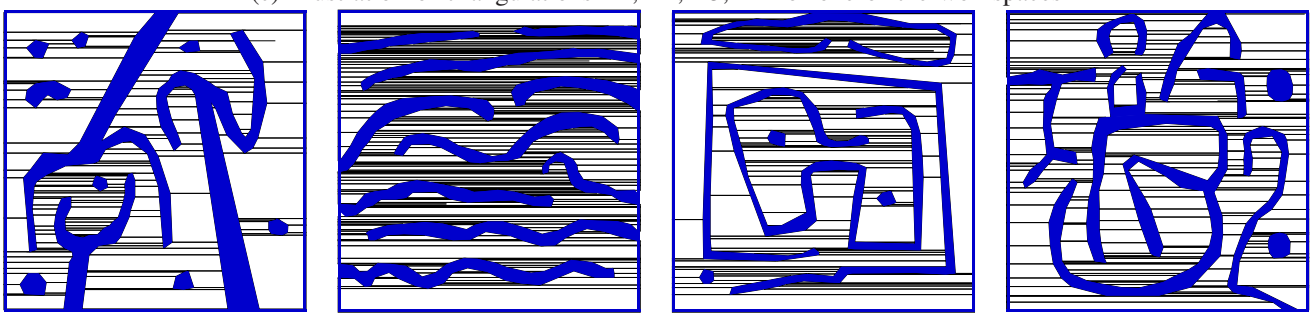
(a) Workspaces used for the experiments



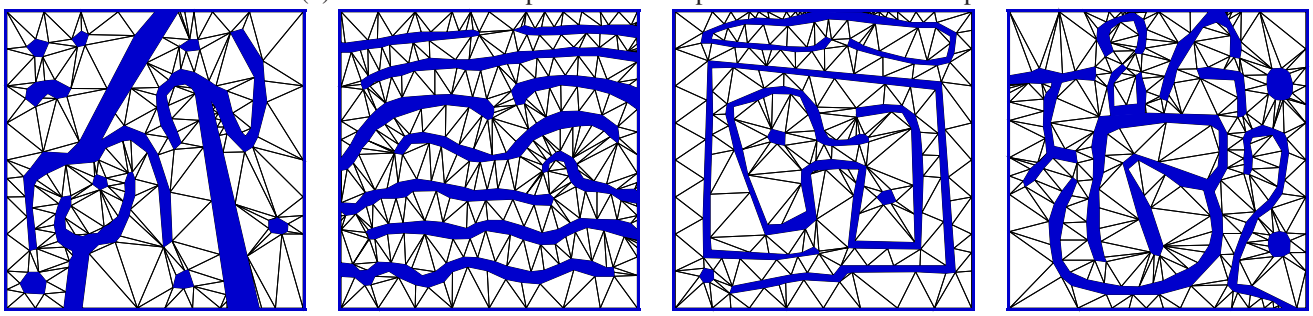
(b) Illustration of some grid decompositions, 8×8, 16×16, 32×32, 64×64, for one of the workspaces



(c) Illustration of triangulations T1, T2, T3, T4 for one of the workspaces



(d) Illustration of trapezoidal decompositions for each workspace



(e) Illustration of conforming Delaunay triangulations for each workspace

Fig. 2. (a) Workspaces used for the experiments. Obstacles are shown in blue. Each figure also illustrates a typical query for a second-order kinodynamic car model with the initial state shown in green and the goal state shown in red. All drawings are according to scale. (b-e) Illustrations of different workspace decompositions used for the experiments.

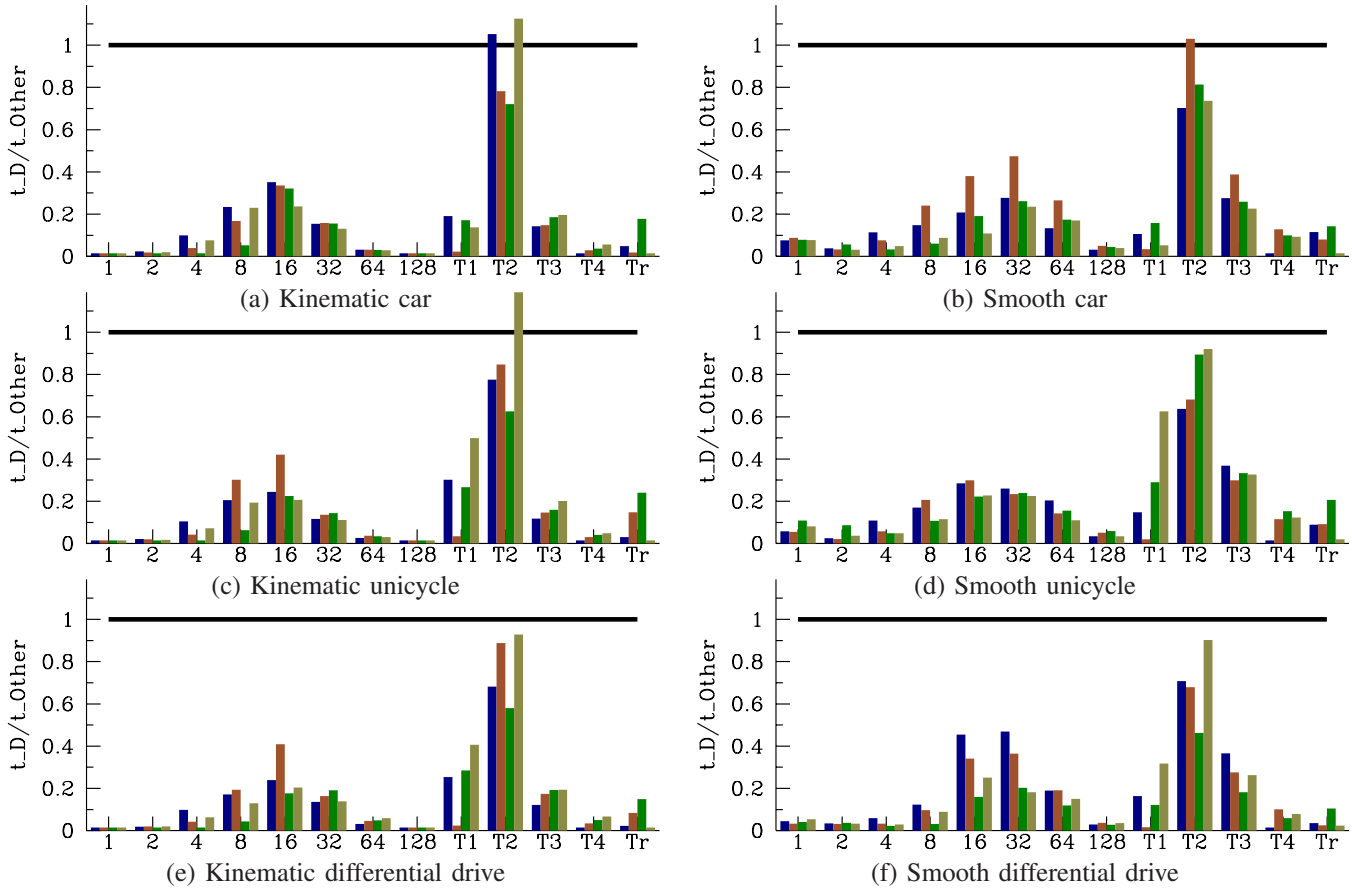


Fig. 3. Bars (from left to right) correspond to the results when using different decompositions (x-axis) on the workspaces in Fig. 2. t_D denotes the computational efficiency of DSLX, as defined in section III-D, when using a conforming Delaunay triangulation. t_{Other} denotes the computational efficiency of DSLX when using a different decomposition. Decompositions 1, 2, 4, 8, 16, 32, 64, and 128 denote grid decompositions. Decompositions T1, T2, T3, T4 denote triangular decompositions. Decomposition Tr denotes trapezoidal decomposition. Descriptions of these decompositions can be found in section III-C.

TABLE I

COMPUTATIONAL EFFICIENCY OF DSLX WHEN USING CONFORMING DELAUNAY TRIANGULATIONS OF THE FOUR WORKSPACES IN FIG. 2

	KCar	KUni	KDDrive	SCar	SUni	SDDrive
A	0.91s	0.61s	0.89s	13.29s	8.85s	11.89s
B	1.15s	1.07s	1.46s	22.48s	13.71s	13.25s
C	1.06s	0.98s	1.77s	19.41s	27.27s	17.47s
D	0.83s	0.78s	1.65s	11.24s	15.76s	18.66s

Table I shows the computational efficiency of DSLX when using conforming Delaunay triangulations. The graphs in Fig. 3 are obtained by plotting t_D/t_{Other}, where t_D denotes the computational time of DSLX when using a conforming Delaunay triangulation and t_{Other} denotes the computational time of DSLX when using one of the other grid, triangular, or trapezoidal decompositions described in section III-C.

To illustrate the difficulty of the problems solved and the computational efficiency of DSLX, we first note that RRT and EST require one to two orders of magnitude more time than DSLX. For example, when using the third workspace of Fig. 2(a) and the second-order unicycle model RRT and EST require more than 250s on average to solve a query, while DSLX requires only 27.27s. As another example, when using the fourth workspace of Fig. 2(a) and the second-order car model RRT and EST require more than 750s, while DSLX requires less than 12s. Similar speedups are obtained for the other workspaces and robotic models but due to scope and space limitations are not included here. We note that these results are in agreement with the work in [7] which provided an extensive comparison of DSLX to RRT and EST and showed that significant computational speedups of one to two orders of magnitude are offered by DSLX.

The results in Fig. 3 show that as the grid size decreases, the computational efficiency of DSLX generally increases. After a certain point the grid size becomes too small and the computational efficiency of DSLX starts decreasing. Similar to our findings for grid decompositions, the computational efficiency of DSLX depends on the granularity of the triangular decompositions. For example, DSLX is faster when using triangulation T_2 than when using the coarse-triangulation T_1 or the fine-triangulations T_3, T_4.

When the decomposition is too coarse-grained there is no significant advantage from the interplay of discrete search and continuous state-space exploration. DSLX behaves in such cases similar to traditional sampling-based tree planners that do not rely on a discrete component to lead the continuous state-space exploration.

On the other hand, when the decomposition is too fine-

grained, the overhead associated with the interplay between discrete search and continuous exploration outweighs the computational advantages it provides. Computational cost is incurred by lead computations and updates to exploration estimates. In fact logged data indicates that on too fine-grained decompositions, e.g., 128×128 grid or $T4$, only a small fraction of the total time is spent on calls to propagate, which is responsible for extending the exploration tree.

These findings thus show that to take full advantage of the interplay between discrete search and continuous state-space exploration and increase the computational efficiency of DSLX, the right level of decomposition granularity for the motion-planning problem under consideration must be found. Fine-tuning can however require extensive efforts.

An alternative solution is to use conforming Delaunay triangulations which allow DSLX to achieve significant computational efficiency, as shown in Table I and Fig. 3. We note that in some cases triangulation T2 offers slightly better results. The advantage of using conforming Delaunay triangulations however is that without fine-tuning DSLX can achieve significant computational speedups that are better or comparable to computational speedups obtained by fine-tuning the workspace decomposition.

IV. Discussion

This work conducted an extensive study of the impact of different workspace decompositions on DSLX in the context of kinodynamic motion planning. Experiments were carried out using several first and second-order kinodynamic robot models and various workspace decompositions that have commonly appeared in motion planning literature, such as grid, trapezoidal, and triangular decompositions.

This work showed that the granularity of the workspace decomposition directly impacts the computational efficiency. DSLX is faster when the decomposition is neither too fine- nor too-coarse grained. When the decomposition is too fine-grained, the computational advantages offered by the interplay between the continuous state-space exploration and discrete search are outweighed by the computational cost associated with lead computations and updates to exploration estimates. On the other end of the spectrum, when the decomposition is too coarse-grained there is no significant advantage by using the lead to guide the continuous state-space exploration. Finding the right level of granularity to take full advantage of the computational benefits offered by the interplay between continuous state-space exploration and discrete search could further increase the computational efficiency of DSLX but can require extensive fine-tuning.

This work demonstrated that significant computational efficiency can instead be obtained with no fine-tuning by using conforming Delaunay triangulations. In the context of DSLX, conforming Delaunay triangulations provide a natural workspace decomposition that achieves a balance between coarse- and fine-grained decomposition, and thus allows a remarkably efficient interplay between the continuous state-space exploration and discrete search. Although the focus of this work was on decompositions of 2D workspaces, Delaunay triangulations are also applicable to 3D workspaces. Algorithms and their complexity for computing conforming Delaunay triangulations in 2D and 3D or computing constrained Delaunay triangulations in any dimension, which are almost conforming Delaunay except in a vew places, are surveyed in [18]. In future work, we plan to use these findings to successfully apply DSLX to increasingly challenging motion-planning problems. As we move in this direction, it becomes important to also consider decompositions of low-dimensional projections of the state space, especially in the case of non-vehicle robotic systems, such as manipulators.

References

[1] 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.

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

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

[4] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," in *IEEE International Conference on Robotics and Automation*, 1999, pp. 473–479.

[5] 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.

[6] 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.

[7] E. Plaku, L. E. Kavraki, and M. Y. Vardi, "Discrete search leading continuous exploration for kinodynamic motion planning," in *Robotics: Science and Systems*, Atlanta, Georgia, 2007.

[8] J.-C. Latombe, *Robot Motion Planning*. Boston, MA: Kluwer, 1991.

[9] 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.

[10] C. Holleman and L. E. Kavraki, "A framework for using the workspace medial axis in PRM planners," in *IEEE International Conference on Robotics and Automation*, 2000, pp. 1408–1413.

[11] M. Foskey, M. Garber, M. C. Lin, and D. Manocha, "A voronoi-based hybrid motion planner," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Maui, HI, 2001, pp. 55–60.

[12] Y. Yang and O. Brock, "Efficient motion planning based on disassembly," in *Robotics: Science and Systems*. Cambridge, USA: MIT Press, 2005, pp. 97–104.

[13] R. Bohlin, "Path planning in practice: Lazy evaluation on a multi-resolution grid," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2001, pp. 49–54.

[14] H. Kurniawati and D. Hsu, "Workspace-based connectivity oracle: An adaptive sampling strategy for PRM planning," in *International Workshop on Algorithmic Foundations of Robotics*, New York, NY, 2006, in press.

[15] J. R. Shewchuk, "Delaunay refinement algorithms for triangular mesh generation," *Computational Geometry: Theory and Applications*, vol. 22, no. 1-3, pp. 21–74, 2002.

[16] E. Plaku, K. E. Bekris, and L. E. Kavraki, "OOPS for Motion Planning: An Online Open-source Programming System," in *IEEE International Conference on Robotics and Automation*, Rome, Italy, 2007, pp. 3711–3716.

[17] A. Narkhede and D. Manocha, "Fast polygon triangulation based on Seidel's algorithm," in *Graphics Gems 5*, A. Paeth, Ed. Academic Press, 1995.

[18] J. R. Shewchuk, "General-dimensional constrained delaunay and constrained regular triangulations, i: Combinatorial properties," *Discrete & Computational Geometry*, 2007.