Carnegie Mellon University
Research Showcase

Robotics Institute | School of Computer Science

1-1-2008

Adaptive Workspace Biasing for Sampling Based Planners

Matt Zucker
Carnegie Mellon University

James Kuffner
Carnegie Mellon University, kuffner@cs.cmu.edu

J. Andrew Bagnell
Carnegie Mellon University, dbagnell@ri.cmu.edu

Follow this and additional works at: http://repository.cmu.edu/robotics

Part of the Robotics Commons

Recommended Citation
Zucker, Matt; Kuffner, James; and Bagnell, J. Andrew, "Adaptive Workspace Biasing for Sampling Based Planners" (2008). *Robotics Institute.* Paper 47.
http://repository.cmu.edu/robotics/47

This Conference Proceeding is brought to you for free and open access by the School of Computer Science at Research Showcase. It has been accepted for inclusion in Robotics Institute by an authorized administrator of Research Showcase. For more information, please contact research-showcase@andrew.cmu.edu.

Adaptive workspace biasing for sampling-based planners

Matt Zucker James Kuffner J. Andrew Bagnell

The Robotics Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA, 15213, USA
{*mzucker,kuffner,dbagnell*}*@cs.cmu.edu*

***Abstract*— The widespread success of sampling-based planning algorithms stems from their ability to rapidly discover the connectivity of a configuration space. Past research has found that non-uniform sampling in the configuration space can significantly outperform uniform sampling; one important strategy is to bias the sampling distribution based on features present in the underlying workspace. In this paper, we unite several previous approaches to workspace biasing into a general framework for automatically discovering useful sampling distributions. We present a novel algorithm, based on the REINFORCE family of stochastic policy gradient algorithms, which automatically discovers a locally-optimal weighting of workspace features to produce a distribution which performs well for a given class of sampling-based motion planning queries. We present as well a novel set of workspace features that our adaptive algorithm can leverage for improved configuration space sampling. Experimental results show our algorithm to be effective across a variety of robotic platforms and high-dimensional configuration spaces.**

I. INTRODUCTION

In recent years, probabilistic sampling-based motion planning algorithms have gained popularity in a broad variety of problem domains. By building a graph structure that implicitly represents the connectivity of a configuration space, algorithms such as the rapidly-exploring random tree (RRT) and probabilistic roadmap (PRM) planners can often solve high-dimensional planning problems much more quickly than combinatoric or complete methods [1], [2].

Although probabilistically complete, in practice, sampling-based planners commonly suffer from the so-called "narrow passage problem": the relatively low probability of sampling states that extend the graph through small gaps in free configuration space [3]. Much effort has been dedicated to developing sampling strategies for improved planning, based on either configuration space or workspace features.

Configuration space ($\mathcal{C}$-space) samplers include the bridge test sampler to identify samples within narrow passages in $\mathcal{C}$ [4], approximate medial axis sampling [5], and rejection of samples within the Voronoi region of near-colliding states [6]. Characteristics of specific robotic platforms such as manipulability can be used as well [7]. For dynamic re-planning applications, the sampling distribution can be biased towards previously useful states [8]. Since many sampling strategies themselves take as input samples from an underlying distribution, their strengths can be combined by setting up chains of dependent samplers [9].

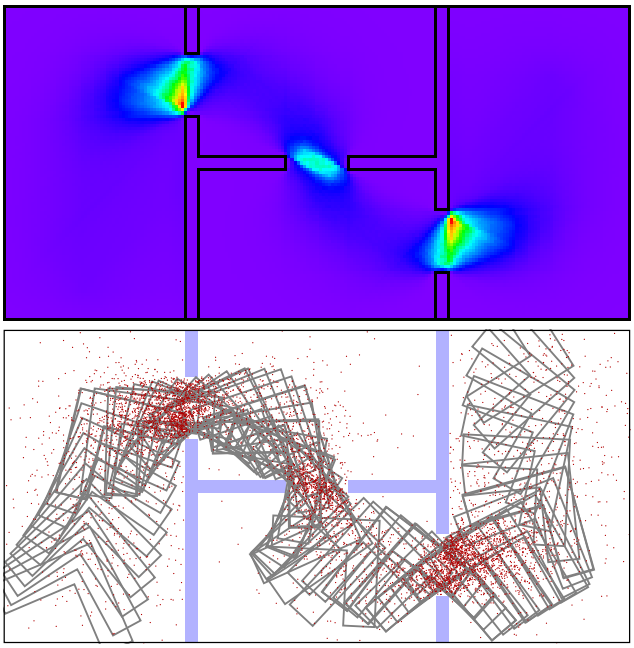

Fig. 1. Workspace-biased sampling distribution used to plan paths for a planar rotating and translating L-beam. *Top:* automatically learned probability distribution. *Bottom:* workspace samples drawn from distribution (red) shown along with robot path (gray).

Workspace biasing methods include the Gaussian sampler for sampling near workspace obstacles [10], as well as the workspace medial axis approach [11]. More recent methods have focused on geometric analysis to identify workspace narrow passages [12], [13].

In this paper, we propose a general framework for adapting the sampling distribution of probabilistic motion planners to a particular problem class. We implemented our approach to bias bidirectional RRT sampling based on several workspace features, including a novel set of features which we present here. Experimental results show our method to be significantly effective in two different problem classes.

II. Adaptive workspace biasing

Recent work in machine learning [14], [15] has shown that informed and effective heuristic functions can be learned for (approximately) optimal deterministic planners. Effective heuristics prove crucial for solving complex robot planning problems. Similarly, we expect that machine learning techniques can prove effective in adapting the performance of modern sample based planners.

Despite all the effort invested in developing sampling strategies, experiments have shown that there is no "one-size-fits-all" sampler that gives optimal performance across all classes of planning problems [16], [17]. The sampling strategy here plays a role reminiscent of the learned heuristic functions in [14], [15]: an informed one can be crucial for effective planning. A key idea in our work is that the sampling strategy adopted by a probabilistic sample-based planner can be understood as a stochastic policy in the sense common in the field of Reinforcement Learning [18].

The cost-adaptive strategy for hybrid PRM sampling attempts to address this problem by maintaining weights on a discrete set of samplers [19]. Our approach differs from the cost-adaptive hybrid PRM approach in that it can be applied to a broader class of randomized planners, whereas the hybrid sampling approach features a reward function tailored specifically to probabilistic roadmaps. By applying rewards equally to all samples from a given planning query, we avoid the task of assigning credit or blame to individual samples—a difficult proposition even after the planning query is finished. Furthermore, we base our approach on continuous features in the workspace as opposed to performance of a discrete set of sampling strategies.

We choose to focus on the workspace instead of the configuration space space in forming our sampling distributions because many workspace features, such as visibility and connectivity, can be computed algebraically or numerically. Computing the configuration-space analogs of such features, however, can be intractable—in fact, doing so is often at least as hard as solving the original planning problem. Fortunately, it is often possible to deduce information about configurations from workspace information alone. For instance, the workspace distance to the nearest obstacle at a given configuration can be mapped into a free volume in the configuration space [20], [21]. More theoretical underpinnings of workspace-to-$\mathcal{C}$ mappings can be found in [13].

Another reason to focus on the workspace is due to size considerations. In order to have a valid parametric distribution which is easily sampled, we use a finite-element decomposition of the robot workspace. Creating a partition of a high-dimensional configuration space with similar granularity is typically infeasible.

Our framework produces a workspace-biased distribution that yields good performance on a given class of planning problems based on a weighting of workspace features. Aside from a sampling-based planner, our approach requires: a discretization of the workspace into a set $\mathcal{X}$ of finite elements; a feature vector $f(x) : \mathcal{X} \mapsto \mathbb{R}^K$ evaluated at each $x \in \mathcal{X}$; a distribution $p(y|x)$ over robot configurations given a particular $x \in \mathcal{X}$; and finally a reward function $r(\xi)$ which assigns rewards based on planning performance given the samples $\xi = \{x^{(1)}, \ldots, x^{(T)}\}$. We note that the methods presented here should generalize across a variety of probabilistic planning algorithms as long as the ingredients presented above are provided.

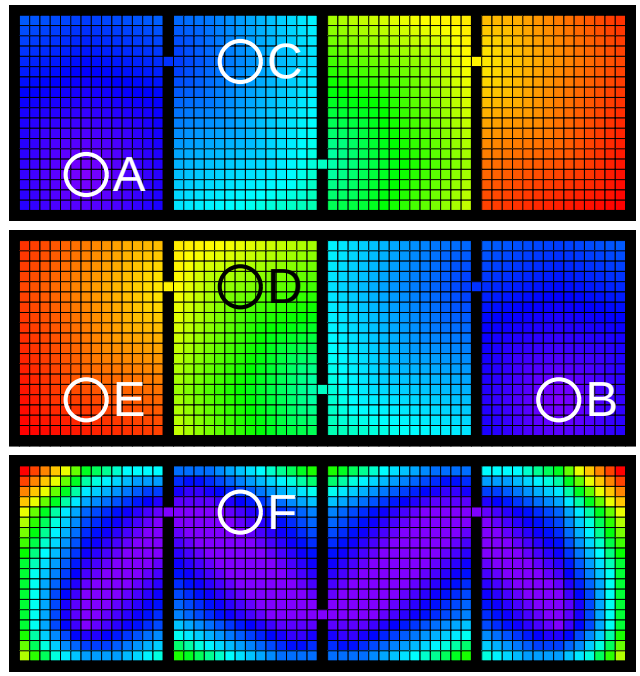

Fig. 2. Illustration of elliptical path distance in a simple 2D workspace. *Top:* optimal cost-to-go to get to initial workspace element A. *Middle:* optimal cost to get to goal element B. *Bottom:* Elliptical path distance. The value at F is equal to the sum of the values at C and D minus the value at E.

The overall course of the approach is as follows: Invoke sampling-based planner on an instance of the problem class. To generate samples, the planner should first sample an element $x \sim q(\theta, x)$ based on a weighting θ of the workspace features $f(x)$, and then sample a configuration y from $p(y|x)$. At the end of a planning episode, assign a reward $r(\xi)$. Finally, estimate the gradient of the expected reward with respect to the weights θ, take a gradient step, and start again.

A. Elliptical path distance

We found that the *elliptical path distance* feature proved to be critical in biasing sampling distributions. This feature, used in [22] as a descriptor of path topologies, helps to preferentially sample locations which fall close to the optimal workspace path between the initial and goal configurations of the search. To compute the feature, we run an optimal planner, such as Dijkstra's algorithm, twice over the set of workspace elements. The first run computes $d(x, x_{init})$, the optimal cost-to-go from every workspace element $x \in \mathcal{X}$ to get to the workspace element x_{init} associated with the initial configuration of the robot. The second run computes the optimal cost to get to x_{goal}, the workspace element

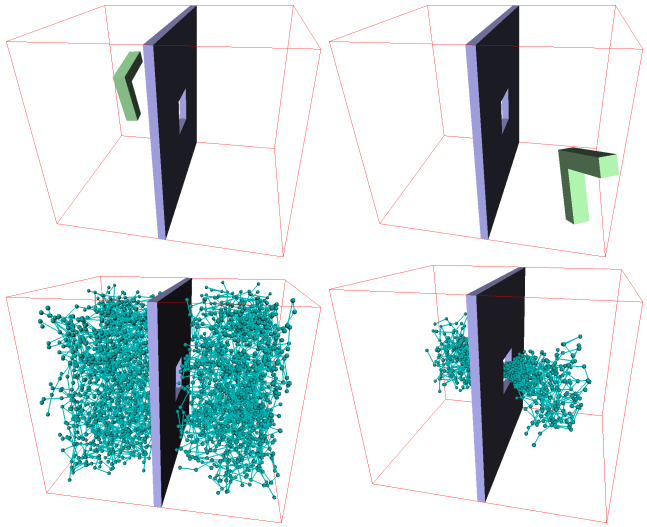

Fig. 3. The L-beam planning problem. *Top:* initial and goal configurations. *Bottom:* Comparison of two solution trees. The tree on the left (2924 nodes) was built with uniform sampling. The tree on the right (404 nodes) uses workspace feature biasing.

associated with the goal configuration. Then, the value of the elliptical path distance is given by

$$d(x) \quad = \quad d(x, x_{init}) + d(x, x_{goal}) - d(x_{init}, x_{goal}) \quad (1)$$

The feature is so named because all elements in a level set of $d(x)$ have the same net optimal cost to get to the initial and goal workspace elements, just as all points on an ellipse have the same total distance to the foci of the ellipse. The elliptical path distance has a value of zero anywhere along an optimal workspace path between the initial and goal elements. See Figure 2 for an illustration in a simple 2D workspace.

B. Workspace-biased distribution

We create a workspace-biased probability distribution $q(\theta, x)$ that samples in a discretization $\mathcal{X}$ of the workspace into finite elements. The distribution $q(\theta, x)$, a member of the exponential family of distributions, uses the weight vector θ to bias the salience of various features in the workspace. Our current implementation uses a uniform voxel discretization of the workspace, but non-uniform representations of the workspace (i.e. quadtrees, octrees, KD-trees) are in practice equally viable. For each workspace element $x \in \mathcal{X}$, we compute a feature vector $f(x) \in \mathbb{R}^K$.

Now we define the distribution $q(\theta, x)$ as

$$\begin{aligned} q(\theta, x) &= \frac{1}{Z(\theta)} \exp\left(\theta^T f(x)\right) \\ Z(\theta) &= \sum_{x_i \in \mathcal{X}} \exp\left(\theta^T f(x_i)\right) \end{aligned}$$

Note that $Z(\theta)$ serves as a normalizing term so that the probabilities sum to one. The distribution $q(\theta, x)$ is a Gibbs distribution, and has several useful properties: When $\theta = 0$, $q(\theta, x)$ becomes the uniform distribution. As θ_j approaches ∞, the distribution holds nonzero probabilities only where $f_j(x)$ reaches its maximum value. Given a uniform discretization over the workspace, as the cell size approaches zero, $q(\theta, x)$ approaches a continuous probability density function.

C. Sampling from the distribution

To sample $x \sim q(\theta, x)$, we store the cumulative probability

$$C_m \quad = \quad \sum_{i=1}^{m} q(\theta, x_i)$$

for $m = 1, \ldots, |\mathcal{X}|$, and then sample a real number β uniformly in the interval $(0, 1]$. There exists a unique element x_i for which $C_{i-1} < \beta \leq C_i$. For a given β value, the corresponding x_i can be found efficiently via binary search.

Recall that each $x \in \mathcal{X}$ corresponds to a volume in the workspace, but our sampling-based planner requires samples in the configuration space. Having sampled $x \sim q(\theta, x)$, we now need to sample $y \sim p(y|x)$. Since there is typically a one-to-many relationship between workspace locations and full configurations which is platform-specific, formulating the distribution $p(y|x)$ must depend upon the robot and configuration space being used. For example, with the 7-DOF planar arm illustrated in Figure 4, we define $p(y|x)$ to select among configurations which leave the end effector positioned within the sampled workspace volume x.

D. Estimating the reward gradient

The distribution for a sampling-based planner can be viewed as a stochastic policy. We are interested in the expected reward for a given planning episode:

$$\eta(\theta) \quad = \quad E_q[r(\xi)]$$

Given that the reward is assigned at the end of the episode, and given that all samples are drawn from q, we can approximate the policy gradient with respect to θ by multiplying the reward for the planning episode by the average score ratio over all samples:

$$\widehat{\nabla}\eta(\theta) \quad = \quad \frac{r(\xi)}{T} \sum_{t=1}^{T} \frac{\nabla q(\theta, x^{(t)})}{q(\theta, x^{(t)})}$$

In the case of the distribution $q(\theta, x)$ described above, the score ratio is the difference between the feature vector at a particular element and the expectation of the feature vector under the distribution:

$$\begin{aligned} \frac{\nabla q(\theta, x)}{q(\theta, x)} &= f(x) - E_q[f] \\ &= f(x) - \sum_{x_i \in \mathcal{X}} f(x_i)\, q(\theta, x_i) \end{aligned}$$

The update rule for θ is then

$$\theta_{new} \quad = \quad \theta_{old} + \alpha\, \widehat{\nabla}\eta(\theta)$$

where α is a small positive constant.

In our experiments, our goal was to find a sampling distribution to produce faster planning times. Hence, we used a very simple reward function equal to the number of queries

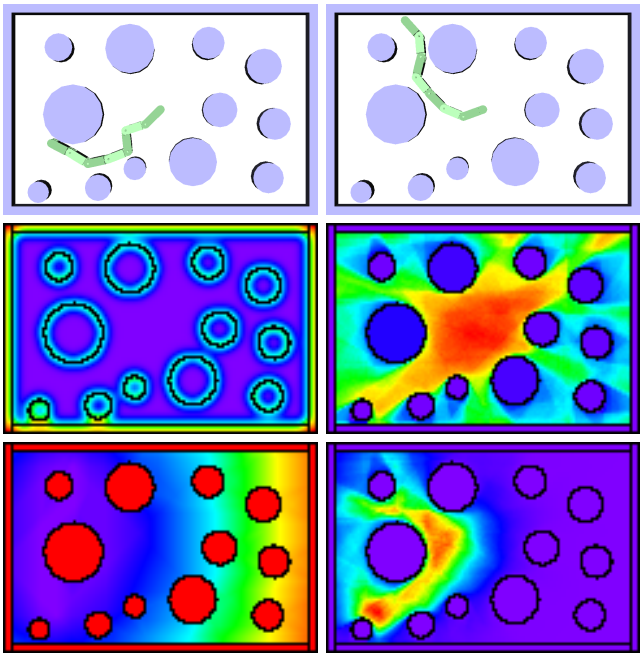

Fig. 4. *Top:* planar 7-DOF arm in sample initial and goal configurations. *Middle:* f_{g8} feature (*left*), and f_{rv} feature (*right*). *Bottom:* f_{epd} feature (*left*), and learned likelihood function (*right*).

solved per unit time. Instead of taking a gradient step after every planning query, we concatenated every five planning queries into a single sequence of samples ξ in order to get a smoother estimate of the reward gradient.

Other viable reward functions which are easy to implement include maximizing the probability that a query is solved given a probability of $(1-\gamma)$ at every timestep of resetting and beginning a new planning episode; or, suffering a uniform loss at every time step until a problem is solved, up to a maximum.

III. Experiments

Our experiments focused on finding good sampling distributions for the bidirectional RRT algorithm in two different problem classes. The first two problem classes consists of maneuvering a free-flying L-shaped beam through a narrow passageway (see Figure 3). The second class requires planning paths for an 7-DOF planar arm in a cluttered environment (see Figure 4). Experimental results are summarized in Table I.

A. Features used

We selected the same set of four features in each problem class. Features f_{g4} and f_{g8} are Gaussian convolutions of voxel occupancy with varying supports, directly corresponding to the strategy of [10]. Feature f_{rv} computes the relative visibility of each voxel cell in the workspace by casting rays to a number of reference locations and counting the number of visible locations. Finally, f_{epd} is computed as the elliptical path distance with respect to the initial and goal workspace elements of each planning query, as in Equation 1.

To avoid numerical instability, we rescale all the features so that $|f_i(x)| \leq 1 \ \forall x, i$. Representative illustrations of the features can be found in Figure 4.

Obviously, time spent computing features is an important consideration. Running a low-dimensional optimal planner twice as a preprocessing step for a high-dimensional planner may seem like a poor use of computer cycles; however, optimal planning on Euclidean grids is quite efficient [23], and does not add much running time in practice. For the 3D L-beam scene, the Dijkstra's step took 0.3s on average; for the planar scenes, runtime to compute f_{epd} averaged about a millisecond. The f_{rv} feature took somewhat longer to compute, especially on large scenes, owing to a naïve software implementation. In future work, we will investigate the use use of approximate visibility calculation as well as GPU-based line-of-sight algorithms [24]; we decided to focus the present experiments on vetting our framework for searching the space of workspace distributions.

B. Bidirectional RRT algorithm

The bidirectional RRT algorithm attempts to find a continuous path through $\mathcal{C}_{free}$ from the initial configuration of the system y_{init} to some goal configuration y_{goal} [1]. The algorithm begins by initializing the two search trees T_{init} and T_{goal} to contain their respective roots y_{init} and y_{goal}. The tree T_{init} is initially set to be the *active* tree. Then the following steps are repeated: A configuration $y_{new} \in \mathcal{C}$ is sampled. Then its nearest neighbor $y_{nearest}$ in the active tree T is selected according to a scalar metric $\rho(y_1, y_2) \mapsto \mathbb{R}^+$ defined on $\mathcal{C}$. Next, an edge e is generated which extends from $y_{nearest}$ towards y_{new}; if e lies in $\mathcal{C}_{free}$, then the terminal point of the edge (often y_{new} itself) is inserted into T as a child of $y_{nearest}$. The active tree is then swapped. If the previous extension was successful, an attempt to connect the trees by extending from the newly active tree to y_{new} is made. Otherwise, the process repeats until the trees are connected or another termination criterion has been reached (e.g., a specified time or memory limit is exceeded or the algorithm has reached a maximum number of iterations). Once the two search trees are connected the path between y_{init} and y_{goal} is extracted, and the problem is solved.

C. L-beam results

The L-beam scenario is a common starting point for investigating the narrow passage problem in motion planning. It consists of passing an L-shaped beam made of two perpendicular blocks through a square hole in a central wall. In [5], the experimenters construct two versions of the robot in order to scale problem difficulty: a small version made of $30 \times 4 \times 4$ blocks, and a large version made with $34 \times 8 \times 8$ blocks.

For the L-beam, we implemented the conditional distribution $p(y|x)$ by uniformly sampling a random position for the center of the elbow within the workspace element x, and then sampling uniformly among orientations. The $120 \times 100 \times 100$ workspace was decomposed into 82,369 voxels with sides of length 2.5.

We decided to bootstrap learning of the θ vector by performing the policy gradient descent on a restricted version of the problem where the robot is constrained to translation and rotation only along the medial plane of the workspace. To make sure the problem was hard enough to require a thorough search of the parameter space, we used an extra-large robot with $34 \times 10 \times 10$ blocks. Due to the relatively small size of the configuration space, we were able to conduct 1000 runs of the planner quite quickly, for a total of 200 gradient steps.

We then benchmarked the workspace-biased sampler against a uniform sampler in the planar restricted scene with the extra-large robot. For each run, the initial and goal configurations were placed randomly on opposite sides of the passage. In 100 runs of each sampling strategy, the mean planning times were 1.61s for the uniform sampler and 0.052s for the workspace-biased sampler—a speedup of over 22x.

Learned values for θ generalized well to the full 3D case. We ran 100 queries with each sampling strategy for both the small and the large robot. To keep experiment times reasonable, we halted the planner after 20,000 samples. Mean runtimes remained significantly shorter for the workspace-biased sampler on the small robot problem, with a 100% success rate. For the large robot, the difference in runtimes was secondary to the tremendous difference in success rate. While the workspace-biased sampler was able to solve 94 queries in under 20,000 samples, the uniform sampler was able to solve only four.

D. Planar arm results

For the planar arm scene, we anchor the base joint a 7-DOF arm to the center of a workspace containing a set of randomly distributed circular obstacles. For each problem query, we set the initial and goal conditions of the search to random poses within the workspace, with the additional requirement that the initial and goal positions of the end effector be separated by a minimum distance. The workspace for the scene was decomposed into 121×81 voxels.

We implemented the conditional distribution $p(y|x)$ by placing the end effector within the sampled workspace element x and sampling uniformly within the null space of the Jacobian to provide the extra degrees of freedom. To do so, we first sample a set of joint angles uniformly within the joint limts of the arm, and subsequently use a Jacobian transpose-based IK solver to bring the end effector into the workspace cell x.

Although the results of the planar arm experiment were not as dramatic as the L-beam results, we were able to observe nearly a 2x speedup when using the workspace-biased sampler. Presumably, the mapping between workspace features and configuration space features is weaker for a long kinematic chain than is that for a free-flying robot.

IV. Generalizations and Future Work

We have developed our approach using the earliest policy gradient method (Reinforce) to demonstrate its simplicity and effectiveness. In practice, recent work has demonstrated that more sophisticated algorithms can achieve better performance more quickly. In particular, we expect to improve learning speed through the gain adaptation and covariant gradient descent techniques [25], [26].

Our approach to learning effective samplers by treating them as stochastic policies was based on heuristics developed in the workspace of the planning queries. This is particularly convenient for the motion planning problems we discuss here, but it is important to note that the approach generalizes to any effective strategy for sampling. We believe in general that lower-dimensional problems will serve as effective features to guide sampling in the full configuration space, much as simplified or more abstracted planning problems can be used as heuristics to effectively guide deterministic planners [14], [27]. For instance, in coordinated multi-agent motion planning [28], we expect very effective sampling heuristics can be derived from deterministic motion planners for the agents planning independently. Equally, in this work we have treated the mapping $p(y|x)$ from the workspace sample to configuration space sampling as a fixed function. As part of the stochastic sampling policy, it too can naturally be adapted to improve planning performance using the algorithm as presented.

We have presented our work as using a discretization of the underlying workspace to guide the sampling. While computationally convenient, this can of course be generalized for arbitrary samplers in the exponential family over continuous space.

Acknowledgements

The authors gratefully acknowledge the partial support of this research by the DARPA Learning for Locomotion contract.

References

[1] J. Kuffner and S. LaValle, "RRT-Connect: An efficient approach to single-query path planning," in *Proc. IEEE Int'l Conf. on Robotics and Automation*, San Francisco, CA, Apr. 2000, pp. 995–1001.

[2] L. Kavraki, P. Švestka, J. C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration space," *IEEE Trans. on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[3] D. Hsu, L. Kavraki, J. Latombe, R. Motwani, and S. Sorkin, "On finding narrow passages with probabilistic roadmap planners," in *Proceedings of the Workshop on the Algorithmic Foundations of Robotics*. AK Peters, Ltd. Natick, MA, USA, 1998, pp. 141–153.

[4] D. Hsu, T. Jiang, J. Reif, and Z. Sun, "The bridge test for sampling narrow passages with probabilistic roadmap planners," in *Proc. IEEE Int'l Conf. on Robotics and Automation*, vol. 3, 2003.

[5] Y. Yang and O. Brock, "Adapting the sampling distribution in PRM planners based on an approximated medial axis," in *Proc. IEEE Int'l Conf. on Robotics and Automation*, vol. 5, 2004.

[6] A. Yershova, L. Jaillet, T. Simeon, and S. LaValle, "Dynamic-Domain RRTs: Efficient Exploration by Controlling the Sampling Domain," in *Proc. IEEE Int'l Conf. on Robotics and Automation*, 2005, pp. 3856–3861.

[7] P. Leven and S. Hutchinson, "Using manipulability to bias sampling during the construction of probabilistic roadmaps," *Robotics and Automation, IEEE Transactions on*, vol. 19, no. 6, pp. 1020–1026, 2003.

Robot	Sampler	Success rate	Fail time	Tree nodes	Samples	Edge CC	State CC	Plan time	Speedup
7-DOF Planar Arm	uniform	98%	6.131	362	4,488	4,839	15,158	1.278	(0.507)
	workspace	100%	-	306	1,224	1,517	7,624	0.648	1.974
Planar L-beam: 34×10	uniform	98%	12.695	1,584	4,490	6,055	37,692	1.161	(0.045)
	workspace	100%	-	125	636	754	3,894	0.052	22.456
Full 3D L-beam: 30×4	uniform	83%	17.872	3,171	7,021	10,079	111,529	3.945	(0.144)
	workspace	100%	-	657	4,126	4,773	23,357	0.569	6.937
Full 3D L-beam: 34×8	uniform	4%	16.495	3,159	7,471	10,539	111,668	4.573	(0.277)
	workspace	94%	3.365	806	9,821	10,619	39,474	1.265	3.614

TABLE I. Experimental results. Data in columns from "Tree nodes" to "Plan time" are averaged over successful runs of the planner. "Edge CC" and "State CC" are the number of high-level edge collision checks and low level state collision checks, respectively. "Speedup" is calculated as the ratio of planning times between the two sampling schemes. "Fail time" is calculated as the mean time it takes to exceed 20,000 samples (the upper limit for the experiments performed).

[8] J. Bruce and M. Veloso, "Real-time randomized path planning for robot navigation," in *Proceedings of IROS-2002*, Switzerland, October 2002, an earlier version of this paper appears in the Proceedings of the RoboCup-2002 Symposium.

[9] S. Thomas, M. Morales, X. Tang, and N. Amato, "Biasing Samplers to Improve Motion Planning Performance," in *Proc. IEEE Int'l Conf. on Robotics and Automation*, 2007, pp. 1625–1630.

[10] V. Boor, M. Overmars, and A. van der Stappen, "Gaussian sampling strategy for probabilistic roadmap planners," in *Proc. IEEE Int'l Conf. on Robotics and Automation*, vol. 2, 1999, pp. 1018–1023.

[11] C. Holleman and L. Kavraki, "A framework for using the workspace medial axis in PRM planners," in *Proc. IEEE Int'l Conf. on Robotics and Automation*, vol. 2, 2000.

[12] H. Kurniawati and D. Hsu, "Workspace importance sampling for probabilistic roadmap planning," in *Proc. IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, vol. 2, 2004.

[13] J. van den Berg and M. Overmars, "Using Workspace Information as a Guide to Non-uniform Sampling in Probabilistic Roadmap Planners," *International Journal of Robotics Research*, vol. 24, no. 12, p. 1055, 2005.

[14] N. Ratliff, D. Bradley, J. Bagnell, and J. Chestnutt, "Boosting structured prediction for imitation learning," in *Advances in Neural Information Processing Systems 19*. Cambridge, MA: MIT Press, 2007.

[15] Y. Xu, A. Fern, and S. Yoon, "Discriminative learning of beam-search heuristics for planning," 2007.

[16] R. Geraerts and M. Overmars, "A comparative study of probabilistic roadmap planners," in *Proceedings of the Workshop on the Algorithmic Foundations of Robotics*, 2004, pp. 43–57.

[17] ——, "Sampling Techniques for Probabilistic Roadmap Planners," in *Proc. IEEE Int'l Conf. on Robotics and Automation*, 2004, pp. 10–13.

[18] J. Baxter and P. Bartlett, "Direct gradient-based reinforcement learning," in *Circuits and Systems, 2000. Proceedings. ISCAS 2000 Geneva. The 2000 IEEE International Symposium on*, vol. 3, 2000.

[19] D. Hsu and G. Sun, "Hybrid PRM Sampling with a Cost-Sensitive Adaptive Strategy," in *Proc. IEEE Int'l Conf. on Robotics and Automation*, 2005, pp. 3874–3880.

[20] S. Quinlan, "Real-time modification of collision-free paths," Ph.D. dissertation, Stanford University, 1994.

[21] O. Brock and L. Kavraki, "Decomposition-based motion planning: a framework for real-time motion planning in high-dimensional configuration spaces," in *Proc. IEEE Int'l Conf. on Robotics and Automation*, vol. 2, 2001.

[22] Y. Fujita, Y. Nakamura, and Z. Shiller, "Dual Dijkstra Search for paths with different topologies," in *Proc. IEEE Int'l Conf. on Robotics and Automation*, vol. 3, 2003.

[23] J. Kuffner, "Efficient Optimal Search of Euclidean-Cost Grids and Lattices," in *Proc. IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, 2004.

[24] B. Salomon, N. Govindaraju, A. Sud, R. Gayle, M. Lin, D. Manocha, B. Butler, M. Bauer, A. Rodriguez, L. Eifert, *et al.*, *Accelerating Line of Sight Computation Using Graphics Processing Units*. Defense Technical Information Center, 2004.

[25] N. N. Schraudolph, J. Yu, and D. Aberdeen, "Fast online policy gradient learning with SMD gain vector adaptation," in *Advances in Neural Information Processing Systems*, Y. Weiss, B. Schölkopf, and J. Platt, Eds., vol. 18. The MIT Press, Cambridge, MA, 2006, pp. 1185–1192.

[26] J. Bagnell and J. Schneider, "Covariant policy search," in *Proceeding of the International Joint Conference on Artifical Intelligence*, August 2003.

[27] R. Korf, "Finding optimal solutions to rubik's cube using pattern databases," in *Proceedings of the Workshop on Computer Games (W31) at IJCAI-97*, Nagoya, Japan, 1997, pp. 21–26. [Online]. Available: citeseer.ist.psu.edu/korf97finding.html

[28] D. Ferguson, N. Kalra, and A. T. Stentz, "Replanning with RRTs," in *Proc. IEEE Int'l Conf. on Robotics and Automation*, May 2006.