Gaussian PRM Samplers for Dynamic Configuration Spaces

Yu-Te Lin and Shih-Chia Cheng
Computer Science Department
Stanford University
Stanford, CA 94305, USA
{yutelin, sccheng}@cs.stanford.edu
SUID: 05371954, 05362397

***Abstract* - Probabilistic roadmap planners (PRMs) are widely used in high dimensional motion planning problem. However, they are less effective in solving narrow passages because feasible configurations in a thin space are rarely sampled by random. Although some approaches have been proposed, they are either involved in complicated geometrical computations or requiring much information about the obstacles. Moreover, if the configuration space is dynamic instead of fixed, some solution may fail in some unexpected situations. In this work, we provide a novel approach which makes use of preprocessing to construct a Gaussian PRM sampler to replace the randomized sampling. For dynamic configuration spaces, we employ a machine technique to dynamically match samplers to different spaces, which is computed off-line as well.**

Index Terms - Probabilistic roadmap (PRM), sampling strategy, narrow passages, path planning and dynamic configuration spaces.

I. INTRODUCTION

PRM is a widely used path planning algorithm, which works efficiently in high dimensional environments; moreover, it could be involved in a variety of robots. However, there is a critical problem called "narrow passages". Since this problem comes from the randomized sampling in the roadmap construction; intuitively, if we can just sample in the narrow passages, this problem could be solved. In this work, we use some simple data structure, Gaussian distributions, to model the free space by preprocessing. We find the algorithm could improve the connectivity, roughly 25% in our experiment, of roadmap and success in sampling, roughly 50% in our experiment, in the learning phrase. For dynamic configuration spaces (C-spaces), to enumerate samplers for all C-spaces is computationally intractable; therefore, we just pick some representative samplers and generate training data by target application. By a machine technique, the support vector machine (SVM) in our experiment, we can produce a classifier off-line and uses it in each path planning.

The rest of this paper is organized as five sections. Section II describes the related work in PRM. Section III and Section IV describes the construction and reuse of Gaussian PRM samplers respectively. Section V shows the details of implementation and experimental results and Section VII discusses the future work and conclusion. VII and Section VIII discuss possible future work and summarize this research, respectively.

II. RELATED WORK

A. Probabilistic roadmap path planner

In robotics, motion planning usually confronts the problem of high dimensionality, which will cause the search space of configurations to become extremely large. Although the workspace may have only three degrees of freedom (DOF), the C-space may be of 6 or even higher DOF. A general approach to solve this high-dimensional problem is to use the sampling based methods. A very well-known technique is called PRM method [1]. Basically, there are two phrases in this algorithm, the learning phrase and the query phrase. In the learning phrase, it randomly samples free space and uses local planner to construct a roadmap. In the query phrase, it tries to connect the start and goal configurations to the roadmap respectively. If both of the end configurations are able to be connected to the roadmap, a planned path could be generated. Actually, there are different algorithms in this family, such as rapidly exploring random trees (RRTs) [2][3] which builds the roadmap from known configurations, Fuzzy PRM [4], which builds the roadmap without a strictly feasible configurations checking local planner and validates the roadmap in the last stage, and the predictive PRM [5], which uses active sampling and predictive PRM to reduce expansive configuration validations.

B. Sampling Strategies for PRM

This family of research is derived from the general problem narrow passages of PRM. In the learning phrase of roadmap building, this algorithm usually samples for free configurations in a randomized manner. However, by randomized sampling, we can build a roadmap covering the narrow parts only by probing almost everywhere in the C-space, which extremely reduces the benefit of PRM. Consequently, there are several sampling strategies proposed to overcome this problem. Most of these approaches assume that narrow passages appear near the obstacles in a workspace. Therefore, some of them use the heuristic methods based on obstacle surface properties [6] or the shrinking and growing obstacles [7]. There are also some approaches, such as the Gaussian sampling [8] and the bridge test [9] which directly replace the randomized sampler in PRM to make samples near the obstacles.

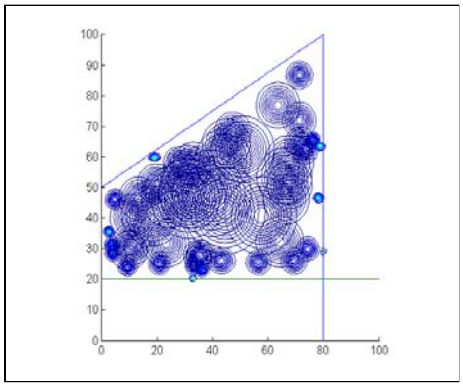

Figure 1. Gaussian PRM Sampler. This figure is drawn from a set of Gaussian Distributions. The circles model the free space of a C-space.

III. GAUSSIAN PRM SAMPLERS

In this section, a pre-computation method for mending this problem called Gaussian PRM Sampler is introduced. In a word, this method learns a set of Gaussian distributions to model the free space offline. That is, by exploring the geography of C-space in advance to construct a set of Gaussian distribution to approximate it. In this way, we can use the Gaussian PRM Sampler (again, a set of Gaussians) to generate free configurations more efficiently in the learning phrase of PRM. In addition, in addressing the narrow passage problem, some configurations sampled in the narrow passage are kept and subsequently fed into the bridge algorithm similar to [9] for further improvement of sampling efficiency.

A. Free Space Learning

Initially, we have no assumptions about the C-Space; there is nothing we can tell about it. Therefore, we use a random algorithm, Algorithm 1, as a start point to generate the set of Gaussians. As the algorithm shows, we adopt the method of expanding the Gaussians gradually to explore the C-space. First we randomly sample a free configuration by adopting the Monte Carlo method and use it as the mean of a Gaussian. Then we simultaneously increase the variances on the diagonal of covariance matrix and test this Gaussian's sampling error rate. If the rate is less than the tolerance, we continue to increase the variances and test its sampling error rate until the error rate is higher than the tolerance. We also have another tolerance for each variance in the matrix. The algorithm will then extend each variance individually to fit the C-space better. Other Gaussians are obtained with the same approach. We repeat this process until the number of Gaussians reaches the threshold and thus get a set of Gaussians for approximating the C-space. Note that we ignore the entries off the diagonal of the covariance matrix since it will enormously increase the computational complexity if we take them into consideration

Algorithm GAUSSIAN_SET_GENERATOR

1. RANDOM_FREE_CONFIGURATION(μ)
2. IDENTIY_COVARANCE_MATRIX($\sum$)
3. **if** GAUSSIAN_SAMPLE_ERROR($\sum$) < *tolerance_all*
   EXTEND_ALL_ELEMENT($\sum$, *step*)
   repeat 3
4. **foreach** GAUSSIAN_SAMPLE_ERROR($\sum$) < *tolerance_individual*
   EXTEND_ONE_ELEMENT($\sum$, *step*)
   repeat 4

Algorithm 1. Gaussian PRM sampler generator.

Algorithm MODIFIED_GAUSSIAN_SET_GENERATOR

1. RANDOM_FREE_CONFIGURATION(μ)
2. IDENTIY_COVARANCE_MATRIX($\sum$)
3. EXTEND_VARIANCE($\sum$, *bound*)
4. **if** GAUSSIAN_SAMPLE_ERROR($\sum$) > *tolerance_all*
   SHRINK_HALF_ALL_ELEMENT($\sum$)
   repeat 4
5. **foreach** GAUSSIAN_SAMPLE_ERROR($\sum$) < *tolerance_individual*
   EXTEND_ONE_ELEMENT($\sum$, *step*)
   repeat 5

Algorithm 2. Modified PRM Gaussian sampler generator.

Algorithm BRIDGE_TEST_SAMPLING

1. RANDOM_NOT_FREE_CONFIGURATION(*C1*)
2. RANDOM_NOT_FREE_CONFIGURATION(*C2*)
3. *C* := MIDDILE_CONFIGURATION(*C1*, *C2*)
4. **if** CEHCK_FREE_CONFIGURATION(*C*) is false
   repeat 1
5. **return** *C*

Algorithm 3. Bridge test sampling.

when expanding a single Gaussian. After all, what we want is to generate an adequate number of Gaussian distributions to model the free space instead of using a single one. From Figure 1, we thus believe that the C-space can be well-approximated by these Gaussians as long as the number of them is sufficient even though the ignoring of the covariance between variables. Moreover, if the distribution is too specific to a single C-space, the usefulness of it will be seriously limited.

B. Modified Gaussian Generator

In the above algorithm, we increase the variances by a small amount at a time and, after the expansion, use the new Gaussian to sample a large set of configurations to calculate the sampling error rate of it. As a result, this naive approach

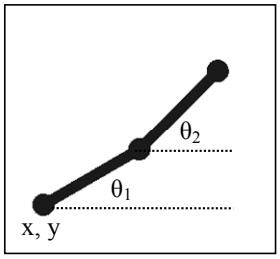

Figure 2. An articulated robot which is parameterized with four variables $(x, y, \theta_1, \theta_2)$

will take a lot of time to generate even a single appropriate Gaussian. Therefore we propose a modified Gaussian generator, Algorithm 2. During the expansion phrase, the algorithm directly jumps to a set of large variances; and the variances are cut in half iteratively. Then, the same approach described before applies.

C. *Bridge Test Gaussian Generator*

To solve the narrow passage problem efficiently, we invite a bridge test method to generator the Gaussians. Actually, it has the same extension process as Algorithm 1 or Algorithm 2, but it has a different mean sampling method. The algorithm of bridge test sampling is described in Algorithm 3; we first choose two infeasible configurations and calculate the middle configuration of them. If the middle configuration of is feasible, then we choose this configuration as the mean of a Gaussian. As a consequence, we will have Gaussians spread in the narrow passage.

D. *Composite Usage*

Theoretically, we can use the set of Gaussians in the C-space from which the set is generated. However, it is not a good idea if we want to use this set of Gaussians to do sampling in a similar C-space. As a matter of fact, we usually use a combination of Gaussian set and bridge test set to sample configurations. More specifically, we give each sampling method a weight; the higher the weight of the method, the more configurations will be generated from it. Therefore, we may have some configurations generated from general Gaussians, some from bridge test Gaussians and even some from random sampling.

IV. REUSE OF GAUSSIAN SAMPLERS

A. *Multi-step Single Query PRM*

Multi-step single query PRM needs to rebuild the roadmap in each single query. Take automatic climbing robot [10] as an example, the planner first plans a sequence of goals for climbing robot to graphs. In successive configurations, a single query PRM planner is responsible for generating a continuous path for the robot motion. Obviously, it is a multi-step single query PRM problem. In this kind of problem, we can not reuse every roadmap built in every query since this problem is composed of several different C-spaces. Therefore, for every query, the planner has to build a new roadmap for planning. If the C-space happens to contain narrow passages, the planner will either spend much time on sampling new configurations to complete the roadmap or simply fail to find out a path successfully. To solve this problem, the pre-built Gaussian PRM samplers described before can help. However, it is impractical to build all Gaussian PRM samplers for all C-spaces. Therefore, the machine learning approaches and pattern classification techniques can be adopted to match similar C-spaces to a common sampling strategy.

B. *C-space patterns*

Since it is impossible to enumerate all Gaussian PRM samplers, we can first generate several samplers for future. Intuitively, if we can generate the most representative samplers, our multi-step planner can have best candidate strategies to choose from. Nevertheless, in most planning problem, we should notice that it may be hard to find out the most representatives. So our method here is to generate more Gaussians than are actually needed and then trim them to the reasonable size later, i.e., we can use the target application to produce possible multiple goals for sampler generation.

Another significant and difficult problem is the acquisition of training data. As mentioned before, there is no way we can get training examples in advance. Therefore, we also use the target application to produce the training data. We can try many multi-step planning to test which sampler is the best for a specific C-space and then record it. As for testing a sampler, we can simply use it to sample several free configurations and compare the resulting error rate. In addition, we should also take into consideration area covering and roadmap connectivity of configurations generated by the sampler.

V. IMPLEMENTATION

A. *Problem Definition*

We construct a simplified test bed for dynamic C-space problem. In the setting, there is an articulated robot with one revolute joint as Figure 2. The robot could be parameterized by four variables (x, y, θ1, θ2). There will be several holds for a robot to grasp, and a robot has to grasp at least one hold at a time. Consequently, in a continuous path, a robot can only transit by changing the θ1 and θ2 to reach another hold. Therefore, for each one-step planning, there will be a different C-space. Since the C-space is not predefined, whenever there is a path to be planned, the planner has to build a new roadmap. We conduct two different experiments. In the first experiment, we build a Gaussian PRM sampler for the specific C-space. As for the second, we choose more than 100 C-spaces to generate Gaussian PRM samplers. In addition, we

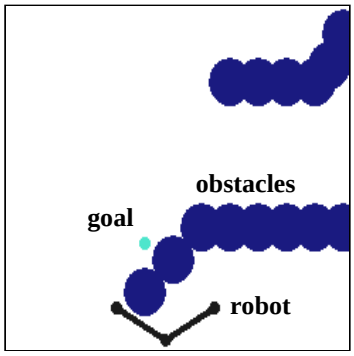

Figure 3. The first experiment. There is only one goal. A Gaussian PRM sample is made for the specific C-space.

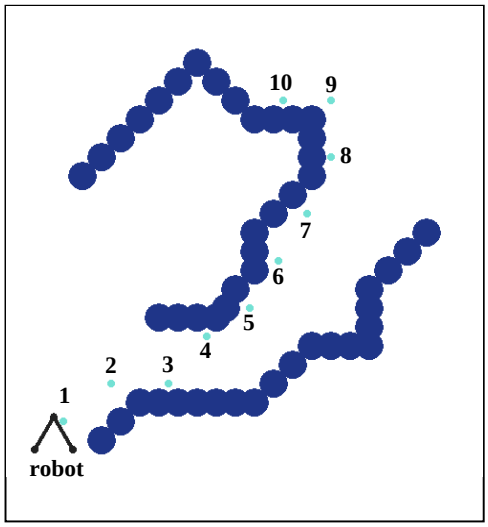

Figure 4. The second experiment. There are ten goals in this setting. The planner will conduct ten times of single-step path planning. In each planning, the planner uses a classifier to match the best Gaussian PRM sampler to replace the randomized sampler in PRM.

generate more than 10000 training examples from different C-spaces by comparing the error rate in each sampler. Then, we use support vector machine (SVM) to get a classifier for future PRM path planning. Figure 3 and Figure 4 visualize the first and second experiment respectively.

B. Results

Table 1 shows the one-step planning result where the "error rate" is the percentage of random sampler's erroneous sampling (not sample on free space) and "number of graphs" denotes the number of graphs in the roadmap. Meanwhile, the "variance" is the average of variances from all degrees of freedom. From the "error rate", we could estimate the accuracy of the sampler's ability to sample in free space while we can judge the connectivity the roadmap from the number of graphs. "Variance", on the other hand, represents how the area covered by a roadmap is. Generally speaking, lower "error rate," less "number of graphs" and bigger "variances" are

error rate	graphs number	variance
15%	208	9776

Table 1. The result of randomized sampling in the first experiment. While the error rate is the percentage of samples fall in the infeasible area, the graph number shows how many graphs in a roadmap and the variance shows the average variance from the nodes in roadmap.

Weight	sampler	error rate	graphs number	variance
20%	G	12.4%	198	8499
	G + B	12.5%	195	8275
40%	G	11.7%	173	8382
	G + B	11.8%	162	8106
60%	G	09.2%	162	7960
	G + B	10.0%	159	7774
80%	G	06.1%	166	6882
	G + B	06.2%	154	6619
100%	G	02.5%	135	5906
	G + B	04.0%	133	5841

G: general Gaussian PRM Sampler
G+B: both general and bridge test Gaussian PRM Samplers

Table 2. The result of Gaussian PRM sampling in the first experiment, where the weight is given for Gaussian PRM sampler(s).

error rate	graphs number	variance
26%	306	11600

Table 3. the result for randomized sampling in the second experiment, which is composed of ten dynamic C-spaces.

weight	sampler	error rate	graphs number	variance
20%	G	25.9%	328	10743
	G + B	27.2%	322	10536
40%	G	20.9%	298	8802
	G + B	22.0%	291	8658
60%	G	20.0%	304	7909
	G + B	20.5%	298	7586
80%	G	00.4%	109	7190
	G + B	00.6%	104	6981
100%	G	0.00%	61	5888
	G + B	0.00%	60	5673

Table 4. The result of Gaussian PRM sampling in the second experiment.

indicative of a better sampler. In Table 2, there are ten more results in which the "weight" denotes the percentage of sampling points generated by Gaussian PRM sampler and the "methods" indicate whether the Gaussian PRM sampler consists of only a general Gaussian PRM sampler (denoted by G) or of both general and bridge test samplers (denoted by B). As can be seen from Table 2, it is obvious that a prefect sampler could not be found, i.e. lower error rate and less number of graphs will inevitably lead to lower variances.

Table 3 and Table 4 demonstrate the results from the second experiment. Results of Table 3 come from randomized sampling while those of Table 4 come from Gaussian PRM sampling. We set a total of ten goals in this experiment. In each one-step planning, the planner will use the pre-computed

classifier to select the most appropriate Gaussian PRM sampler.

C. Discussion

As described above, there exists an inherent tradeoff between achieving lower error rate and higher connectivity and obtaining wider coverage. However, in a PRM planning problem, the connectivity plays a more important role. That the planner has a better connectivity means the roadmap may connect through narrow passages. In our experiments, the roadmap always contains more than 100 graphs since our goal is to reveal the samplers' ability to sample narrow passages. Theoretically, the single and multiple steps planning should lead to nearly identical results, i.e. results from Table 1 should be similar to those of Table 3 while those of Table 2 should be similar to those of Table 4. Nevertheless, the similarity is hard to achieve practically because we have not come up with an effective algorithm that could perfectly generate the most representative Gaussian PRM samplers for a workspace. Moreover, generation of training data is another crucial issue that should not be ignored. In our application, we produce the training examples by inspecting and comparing the error rate. If we can take into the consideration both connectivity and coverage, the criterion for selecting training examples will certainly be more complete and justified.

VI. Conclusion

In summary, Gaussian PRM sampler is a sampling strategy which makes use of pre-computation to reduce the erroneous sampling rate and solve narrow passage problems in PRM. Its ability to improve the connectivity and conduct successful sampling has been proved from our experiments. However, before we integrate it into other applications, a more robust sampler generator is required. Moreover, in order to make the sampler reusable in dynamic C-spaces, a machine learning technique is employed as well. The question which we must consider next is the generation of correct and good training examples as mentioned in the previous section. One final point should be made about our sampler; we design our work and experiment in a way suitable for future integration with the climbing robot [10]. Therefore, the very natural extension of it would be a more flexible and high-dimensional version of it.

Acknowledgment

Parts of the ideas are discussed with Jean-Claude Latombe and Timothy Wolfe Bretl. Along with this work, we produce a PRM toolkit [11] which is composed of several independent components, especially the sampler and feasible checker. Researchers interested in sampling strategies could extend the sampler and test it. Moreover, the toolkit has a separate feasible checker which can be easily worked with existing collision detection packages.

References

[1] L. E. Kavraki, P. Svestka, J. C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions Automatic Control*, vol. 12, no. 4, pp. 566-580, 1996.

[2] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," *Iowa State University,* 1998.

[3] J. J. Kuffner and S. M. LaValle, "RRT-Connect: An efficient approach to single-query path planning," *in Proceedings of the International Conference on Robotics and Automation,* vol. 2, pp. 995-1001, 2000.

[4] C. L. Nielsen and L. E. Kavraki, "A two level fuzzy PRM for manipulation planning," *in Proceeding of the International Conference on Intelligent Robots and Systems,* pp. 1716-1722, 2000.

[5] B. Burns, O. Brock, "Sampling-Based Motion Planning Using Predictive Models," *in Proceedings of International Conference on Robotics and Automation,* pp. 1274-1280, 2005.

[6] N. Amato, B. Bayazid, L. Dale, C. Jones, and D. Vallejo. "OBPRM: An obstacle-based PRM for 3D workspaces," *in Robotics: The Algorithmic Perspective. AK Peters*, 1998.

[7] M. Saha and J. C. Latombe, "Finding narrow passages with probabilistic roadmaps: The small step retraction method," *in Proceedings of International Conference of Robots and Systems,* 2005.

[8] V. Boor, M. Overmars and F. van der Stappen, "Gaussian Sampling for Probabilistic Roadmap Planners," *in Proceedings of the International Conference on Robotics and Automation,* 1999.

[9] D. Hsu, T. Jiang, J. Reif and Z. Sun, "The Bridge Test for Sampling Narrow Passages with Probabilistic Roadmap Planners." *in Proceedings of the International Conference on Robotics and Automation,* 2003.

[10]T. Bretl, S. Lall, J. C. Latombe and S. Rock, "Multi-Step Motion Planning for Free-Climbing Robots," *in Workshop on the Algorithmic Foundations of Robotics*, 2004.

[11]PRMKit http://www.stanford.edu/~yutelin/PRMKit.zip