Motion Planning using Hierarchical Aggregation of Workspace Obstacles*

Mukulika Ghosh[1], Shawna Thomas[1], Marco Morales[2], Sam Rodriguez[3] and Nancy M. Amato[1]

***Abstract*— Sampling-based motion planning is the state-of-the-art technique for solving challenging motion planning problems in a wide variety of domains. While generally successful, their performance suffers from increasing problem complexity. In many cases, the full problem complexity is not needed for the entire solution. We present a hierarchical aggregation framework that groups and models sets of obstacles based on the currently needed level of detail. The hierarchy enables sampling to be performed using the simplest and most conservative representation of the environment possible in that region. Our results show that this scheme improves planner performance irrespective of the underlying sampling method and input problem. In many cases, improvement is significant, with running times often less than 60% of the original planning time.**

I. Introduction

Motion planning is the problem of finding a valid path for a moveable object from a start to a goal placement. In addition to robotics, it has applications in computer graphics [1], virtual prototyping and mechanical design [2], and computational biology [3]. Sampling-based motion planning is the state-of-the-art solution for motion planning problems. They work by generating samples (robot placements), retaining valid ones, and connecting neighboring samples with feasible local plans. Typically, a sample or a path is considered to be valid if it avoids collision with the obstacles present in the environment. The obstacles are represented using geometric objects and their complexity (either in geometry or number) directly impacts the performance of the planners. Moreover, in many cases, the full problem complexity is not needed for the entire solution but only for a part, and planning at the finest level is highly inefficient.

In this paper, we propose a hierarchical aggregation framework that groups the obstacles in the workspace environment to manage complexity in sampling-based motion planning. The hierarchy creates levels of detail of the workspace environment that help in adjusting the environment complexity based on the current need. We begin planning at the coarsest level of the hierarchy. If the planner has not solved the problem, we iteratively increase the level of detail in the hierarchy and call the planner again. At every iteration, we retain the progress of the planner so that in the next iteration the planner can focus on portions of the environment that are newly revealed by the next level in the hierarchy. This reduces the sampling scope to parts of the environment with low obstacle density. By reducing problem complexity for most of the planning time and narrowing the sampling scope dynamically, we can significantly improve the performance of the planner.

The aggregation hierarchy is created by grouping sets of related obstacles as individual entities and computing a new geometric model that approximates the group and contains the original obstacle geometries in the group. Our hierarchical aggregation method identifies nearby obstacles to group and uses a clustering approach such that the lowest level is the original environment and the highest level contains a single aggregated obstacle. This way, we never completely mask the original input environment but restrict sampling to regions with low obstacle density where primitive operations are faster. This aggregation scheme is independent of the underlying planner details and thus applies to many different sampling-based motion planning schemes. Although we studied the performance on point and rigid body robots, our method can work with higher dimensional robots.

The main contribution of this paper are:

- An aggregation hierarchy of the workspace obstacles that can be used to improve the performance of sampling-based planners by restricting sampling to regions of low obstacle density.
- Results in 2D and 3D that show significant improvement in performance with the use of the aggregation hierarchy irrespective of the underlying planner.

II. Preliminaries and Related Work

While the geometry of obstacles and the robot (i.e., the moveable object) are represented in a 2D or 3D workspace, the specific placement or the *configuration* of the robot is characterized by a set of parameters, or *degrees of freedom*, such as position, orientation, and joint angles. The robot's configuration can be abstracted as a point in the space of all configurations (valid or not), known as *configuration space* or $\mathcal{C}_{space}$ [4]. The motion planning problem becomes that of finding a continuous path of valid configurations from the start configuration to the goal configuration in $\mathcal{C}_{space}$.

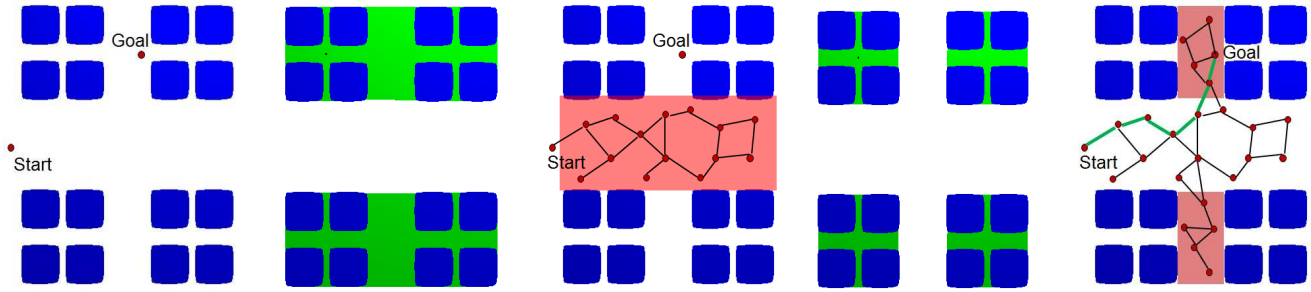
Sampling-based planners, such as Probabilistic Roadmaps (PRMs) [5] and Rapidly-exploring Random Trees (RRT)

*This research supported in part by NSF awards CNS-0551685, CCF-0833199, CCF-1423111, CCF-0830753, IIS-0916053, IIS-0917266, EFRI-1240483, RI-1217991, by NIH NCI R25 CA090301-11 and by Asociacion Mexicana de Cultura A.C.

[1]Parasol Laboratory, Department of Computer Science and Engineering Texas A&M University, College Station, Texas, 77843, USA {mghosh,sthomas,amato}@cse.tamu.edu

[2]Department of Digital Systems Instituto Tecnológico Autónomo de México, Progreso Tizápan, Mexico City, 01080, Mexico marco.morales@itam.mx

[3]Texas Wesleyan University, Texas, 76105, USA. Former affiliation: Parasol Laboratory, Department of Computer Science and Engineering Texas A&M University[1]. sorodriguez@txwes.edu



(a) Original Environment (b) Level N-Aggregation (c) Level N-Planning (d) Level N-1-Aggregation (e) Level N-1-Planning

Fig. 1. Given a motion planning problem in an environment (a), use the levels in the aggregation hierarchy (b) & (d) to apply the planning strategy by sampling in the regions freed at that level (c) & (e) until the problem is solved (e).

[6], are very popular since they are able to solve a wide variety of challenging problems. They work by sampling and connecting a set of valid configurations to produce a graph or a tree that represents the feasible motions. Since it is computationally infeasible to explicitly represent the obstacles in $\mathcal{C}_{space}$ [7], the validity of a configuration is determined through collision detection tests in the workspace. Variants of these basic planners have been developed to address issues such as the difficulty of sampling in narrow passages [8], [9], [10], or to have additional benefits such as optimal paths [11] or high clearance paths [12], [13].

Geometric approximation schemes are often used as supporting or pre-processing tools to handle the complexity of geometric objects. For example, in [14], simplification is used to render complex scenes virtually. A summary of geometric approximation methods can be found in [15]. The proposed aggregation hierarchy is similar to clustering methods as both are used to split a large number of items into a small number of groups. A detailed survey of clustering algorithms can be found in [16]. However, most hierarchical geometric clustering is often used to cluster a set of datapoints [17] or a set of non-disjoint objects [18]. Polygonal clustering of disjoint objects uses simple polygons such as orthogonal polygons [19] and cannot be easily extended to three dimensional objects [20].

Hierarchical geometric approximations are used for efficient handling of various supporting functions in motion planning algorithms such as collision detection [21]. Workspace decomposition, similarly to our proposed aggregation hierarchy, has shown benefit in various motion planning algorithms [22], [23], [24]. Some of them use triangulations of the free space in the environment [23], [24]. However, most of them use the decomposed regions to improve sampling distribution whereas our method aims at improving sampling efficiency. Also, the aggregation hierarchy characterizes the regions in the free space based on the distance between the obstacles.

III. Hierarchical Aggregation for Sampling-based Motion Planning

We apply hierarchical aggregation to workspace obstacles to improve the performance of sampling-based motion planners. In this section, we first briefly describe the hierarchical aggregation approach. We present the underlying aggregation method based on distance thresholds. Next, we describe how the hierarchy can be utilized in sampling-based planning algorithms.

A. Hierarchical Aggregation

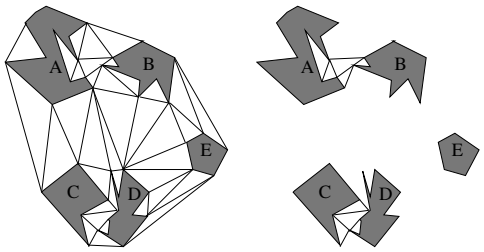
We develop a hierarchy of aggregation levels such that the highest level is the coarsest level where all obstacles are aggregated into a single model and the lowest level is the original input problem where all obstacles retain their original geometries. As one moves up the hierarchy, a greater number of obstacles are aggregated or clustered together.

A shape descriptor defines the approximate or aggregated model (cover) of a group of obstacles. A natural shape descriptor is the convex hull of the group. However, the convex hull may be overly conservative and block key portions of the free space causing over-utilization of the hierarchy. In this paper, we use a novel shape descriptor that incurs in less waste of free space compared to the convex hull or contemporary shape descriptors in the literature. It is described in more detail below.

1) Aggregation by distance thresholds: One simple way to determine which obstacles should be grouped together at level i of the hierarchy is with a distance threshold δ_i. Obstacles are then grouped together if the distance between them is less than δ_i.

We represent the workspace with a graph called a neighborhood graph in which the vertices represent obstacles and weighted edges represent the neighborhood relationship between the obstacles with weights indicating the distance between the obstacles. Given a δ_i, we generate the graph for the aggregated environment by collapsing the edges with weight less than δ_i. The (super) vertices in the graph of the aggregated environment represent the groups whose approximated cover needs to be computed.

The series of δ_is used to define the levels is created in a top-down approach by starting with the δ_i that aggregates all the obstacles together and refining it until the last level has none of the obstacles aggregated together. These δ_is may be further tuned based on various metrics such as the change in volume and the number of groups. The tuning of the hierarchy determines the number of levels.



(a) Initial CDT (b) Final Aggregated Model

Fig. 2. Iterative removal of triangles from (a) the Constrained Delaunay Triangulation (CDT) of the free space between the obstacles results in (b) the shapes of the aggregated obstacles.

2) Aggregated Obstacle Model: Recall that there are many ways to define the shape descriptor, or the cover of the aggregated obstacles. The convex hull of the aggregated obstacles may be too conservative in many cases, particularly as the convexity of the aggregated obstacles increases.

In this work, we use a shape descriptor based on the iterative removal of triangles/tetrahedra from a Constrained Delaunay Triangulation/Tetrahedralization (CDT) of the free space between the obstacles (See Fig. 2) . For level i of the aggregation, the removal of a triangle/tetrahedron depends on the distance threshold δ_i. We further adapt δ_i to ${\delta_i}^{sd}$ for shape descriptor use based on the variation of passage width and the distance between the objects as given by:

$$ {\delta_i}^{sd} = 0.5 * (d_{min} + \sigma_d + \delta_i) $$

where d_{min} is the minimum distance between the grouped obstacles and σ_d is the variation of the passage width between the objects. A triangle/tetrahedron in CDT is removed if it contains an edge whose length is greater than ${\delta_i}^{sd}$.

The iterative removal of triangles/tetrahedra to define the cover of the aggregated obstacles is specific to our implementation. However, other shape covers such as alpha shapes can be used. See Section III-C for more discussion on our implementation and design choices.

3) Identification of differences in aggregation levels: The space freed at a level in the hierarchy is the difference in the free space in the level and its parent's level. It is computed as a by-product in the aggregation hierarchy as the collection of triangles/tetrahedra removed specifically for the creation of the level in the hierarchy. In other words, these are the triangles/tetrahedra which have not been removed at the parent level but removed in the current level.

B. Application to Sampling-based Motion Planning

The aggregation hierarchy not only simplifies the number of obstacles to handle for collision checking, it also represents the evolution of the free space as one moves down the hierarchy. We take advantage of this free space evolution identification by focusing sampling in these regions.

The core idea behind applying the aggregation hierarchy is to iterate through the various levels of the hierarchy until the motion planning problem is solved. Given an input planner, we start at the highest (coarsest) level of the hierarchy and call the input motion planner operating at that aggregation level. If the planner has not solved the problem yet, we iteratively reduce the aggregation level (i.e., increase the level of detail) and call the planner again on the new aggregation level. At each reduction, we retain the collective progress of the planner, identify areas to focus planning based on the difference between the approximation models of the different levels, and continue planning. If the problem is not solved on reaching the lowest level of aggregation (i.e., the original input problem), then the number of samples created is insufficient to solve the problem. So to create more samples, we repeat the entire process starting back at the coarsest level while always retaining the progress made thus far. A sketch of the algorithm is provided in Algorithm 1.

Algorithm 1 Hierarchical Sampling for Motion Planning

Input: An environment E, a motion planning problem MP, and a sampling-based planner P.

1: Create the aggregation hierarchy, $H(E) = \{l_0, l_1, ..., l_n\}$ where l_i is a level in hierarchy.
2: $i = n$.
3: **while** done==FALSE **do**
4: Let R_i be the set of regions freed at level l_i.
5: Let B_i be the set of bounding volumes for R_i.
6: Apply P to MP, restricting new samples to B_i but allowing connections in all of E.
7: $i = i - 1$.
8: **if** $i < 0$ **then**
9: $i = n$
10: **end if**
11: **end while**

As stated earlier, the regions of space freed by the consecutive levels in the hierarchy can be easily computed. They are the difference in free space between the aggregated environments in the current level and its parent level. In Figs. 1(c), 1(e), these regions are shown in translucent red. Restricting sample creation to the freed regions of space not only focuses the sampling space, but it also increases the probability that the new samples are valid. We can approximate the freed regions using a variety of bounding volumes such as bounding boxes, bounding spheres or convex hulls to ease sampling.

Although the regions of space freed at a particular level might be disconnected (e.g., regions A and B in Fig. 1(e)), this is not an issue since we are only restricting the creation of new samples, not their connection. As shown in Fig. 1(e), regions marked A and B are disconnected.

1) Probabilistic Completeness: A key property of sampling-based motion planners is probabilistic completeness. A planner is probabilistically complete if the probability of reporting that no path exists when there is indeed a valid path between the start and the goal approaches 0 as the number of samples approaches ∞. We prove that our method is also probabilistically complete if the underlying planner P used in it is probabilistically complete.

Lemma 3.1: Algorithm 1 is probabilistically complete if the underlying sampling-based planner P used in it is probabilistically complete.

Proof: Recall that at each level i in the hierarchy, Algorithm 1 generates samples in a set of boundaries B_i defined by the aggregated obstacles at that level and then connects these samples using the original obstacles ($\mathcal{C}_{free}$). Because connection occurs in all of $\mathcal{C}_{free}$, it remains to show that sampling (collectively) also occurs in all of $\mathcal{C}_{free}$.

Assume that B_i is a superset of the portion of $\mathcal{C}_{space}$ occupied by the aggregated obstacles at level i. Many definitions of B_i exist for which this is true; in particular, this holds for the bounding volumes used in Algorithm 1. Thus, without loss of generality, we can consider instead the portion of $\mathcal{C}_{free}$ freed at level i and contained in B_i as the region for which sampling takes place. Let $\mathcal{C}_{free_i}$ denote the $\mathcal{C}_{free}$ freed at level i in the aggregation hierarchy. The freed areas at different levels are disjoint, i.e.,

$$\forall i, j | i \neq j : \mathcal{C}_{free_i} \cap \mathcal{C}_{free_j} = \emptyset.$$

Let $\mathcal{C}_{obst_i}$ be the obstacle space at level i in the aggregation hierarchy. Therefore, the $\mathcal{C}_{space}$ of the original environment can be written as:

$$\mathcal{C}_{space} = \mathcal{C}_{free_n} + \mathcal{C}_{obstn}. \tag{1}$$

By definition of the aggregation hierarchy, obstacles that are aggregated in the child level are also aggregated in the parent level. Thus, when level j is a child of level i, the aggregated obstacles, or $\mathcal{C}_{obst_i}$, is the union of $\mathcal{C}_{free_j}$ and $\mathcal{C}_{obst_j}$. Hence,

$$\mathcal{C}_{obst_i} = \mathcal{C}_{free_j} + \mathcal{C}_{obst_j}. \tag{2}$$

Expanding Equation 1 by recursive substitution of Equation 2, we have

$$\begin{aligned} \mathcal{C}_{space} &= \mathcal{C}_{free_n} + \mathcal{C}_{obstn} \\ &= \mathcal{C}_{free_n} + \mathcal{C}_{free_{n-1}} + \mathcal{C}_{obstn-1} \\ &= \mathcal{C}_{free_n} + \mathcal{C}_{free_{n-1}} + \ldots + \mathcal{C}_{free_0} + \mathcal{C}_{obst0}. \end{aligned} \tag{3}$$

As we know, $\mathcal{C}_{free}$ of the original environment can be written as:

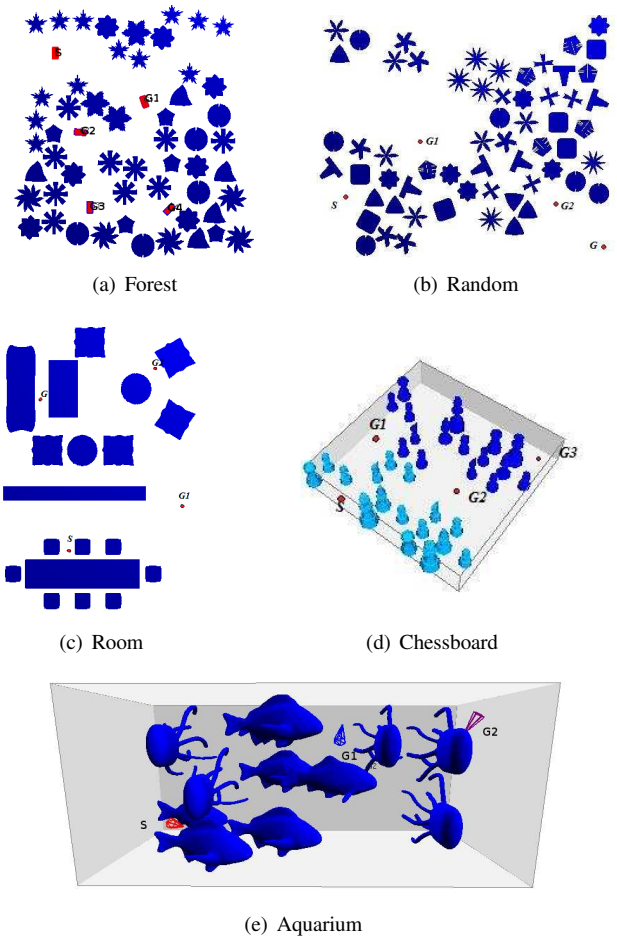
$$\mathcal{C}_{free} = \mathcal{C}_{space} - \mathcal{C}_{obst0}.$$

Substituting Equation 3, we get

$$\mathcal{C}_{free} = \mathcal{C}_{free_n} + \mathcal{C}_{free_{n-1}} + \ldots + \mathcal{C}_{free_0}$$

which is the union of all the spaces freed at each level in the hierarchy and thus all the spaces sampled at each level in the hierarchy.

When Algorithm 1 drills down to the lowest level of the hierarchy (which is the original environment), it samples and connects in all of $\mathcal{C}_{free}$. In this case, its probabilistic completeness is defined by the probabilistic completeness of the underlying planner P. When Algorithm 1 finds a solution before reaching the lowest level in the hierarchy, the problem is solved and probabilistic completeness still holds. ■



(a) Forest (b) Random

(c) Room (d) Chessboard

(e) Aquarium

Fig. 3. Environments used in the experiments. (a) -(c) 2D Environments and (d)-(e) 3D Environments.

C. Implementation Details

The implementation of the aggregation hierarchy is dependent on the CDT of the free space between the obstacles. The CDT is used to facilitate the implementation of various components in this work. The lengths of the edges in the CDT in decreasing order serve as the set of δ_is used for the creation of the hierarchy. We use the CDT edges to construct the neighborhood graph. Every CDT edge connecting two different obstacles is mapped to an edge connecting vertex representatives of the obstacles in the graph with weight equal to the minimum length CDT edge connecting the two obstacles. Lastly, the shape of the aggregated objects is constructed through the iterative removal of triangles/tetrahedra from CDT.

For our implementation, we generate samples in the set of axis aligned bounding boxes as an approximation of the regions freed at each level. One issue in using axis aligned bounding boxes is that the improvement in performance observed is dependent on the orientation of the obstacles with respect to the world coordinate frame. We could directly sample in the sets of triangles/tetrahedra removed at the current level but it might incur computation overhead as these sets might not be convex. Hence, in this initial implementation

we use axis aligned bounding boxes because they are fast to compute, easy to derive sampling bounds on, and completely cover the freed regions.

IV. Results

In this section, we study the performance improvement provided by our method using several different planners: uniform sampling (BasicPRM) [5], GaussianPRM [9] and Obstacle Based PRM (OBPRM) [25]. These were selected as they produce a variety of sampling distributions: uniform (BasicPRM) and near obstacle surfaces (GaussPRM and OBPRM). Although the sampling distributions are very different, our aggregation hierarchy improves the planning efficiency for all of them.

We study several different environments, see Fig. 3. The Random, Room, and Chessboard environments are for point robots in 2 and 3 dimensions. The robot for the Forest environment (Fig. 3(a)) is a rectangular rigid body constrained to the plane, and thus it has 3 degrees of freedom. The robot for the Aquarium environment (Fig. 3(e) is a free-flying pyramid with 6 degrees of freedom. Each of the environment shows the query or tour it needs to solve.

In our experiments, we tune the hierarchy such that the volume of the difference in free space between the levels is at least 10% of the total volume of the environment. This prevents creation of an unmanageable number of levels in the hierarchy.

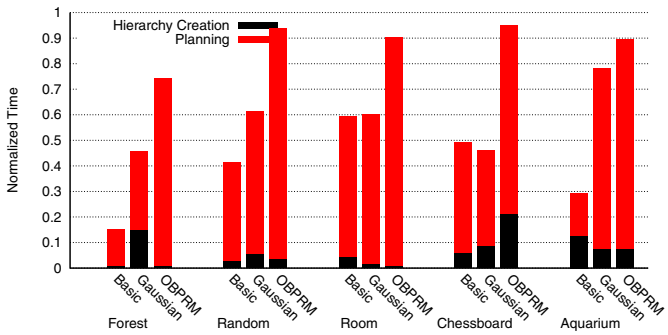
Each method is run until either the query is solved or 10,000 samples are generated. All methods use Euclidean distance, straight-line local planning, and a $k = 5$-closest neighbor connection strategy. We compare the total planning time and the sampling success rate (computed as percentage of valid samples) for each method. All results are averaged over 10 runs.

All methods are implemented in the Parasol Lab motion planning library developed at Texas A&M University. We also use a generic, scalable and parallel graph library called STAPL [26] developed by Parasol Lab at Texas A&M University. Experiments are conducted on Dell Optiplex 780 computers running Fedora 17 with Intel Core 2 Quad CPU 2.83 GHz processors using the GNU gcc compiler version 4.7.

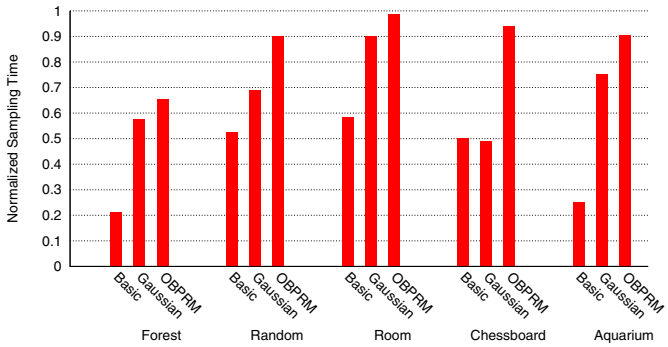
A. Performance

Fig. 4(a) shows the running time of our method with respect to the different underlying sampling-based motion planners. It also shows the part of the time used in the creation of the hierarchy. Times are normalized to the time taken by the planner without the aggregation framework.

Across all environments and all planners, the aggregation framework is faster than the original planner. In half the cases, the aggregation framework is less than 60% of the original planning time. Note that while aggregation occurs in the workspace, performance improvements extend to rigid body cases as well (i.e., the Forest and Aquarium environments) where $\mathcal{C}_{space}$ and workspace are not the same. The time taken to create the hierarchy is relatively small (less than 25% of the total planning time in most of the cases).



(a) Total Planning Time



(b) Sampling Time

Fig. 4. Normalized (a) total running time and (b) sampling time of different aggregation methods with various underlying planners for each environment. Data is normalized to the time spent by each underlying planner without the aggregation framework in each environment.

Fig. 4(b) shows just the portion of the running time spent generating samples. Again, times are normalized to the time taken by the underlying planner to sample without the aggregation framework. We see a similar pattern of improvement in the sampling time as in the overall running time. Thus, most of the performance improvement comes from the effect of aggregation on sampling.

B. Success Rate

Fig. 5 compares the sampling success rate with and without using aggregation hierarchy. The success rate is computed as the percentage of valid samples out of all sampling attempts. We normalize the success rate to that of BasicPRM without the aggregation framework in each environment.

We see that the success rate improves using the aggregation hierarchy irrespective of the original success rate of the

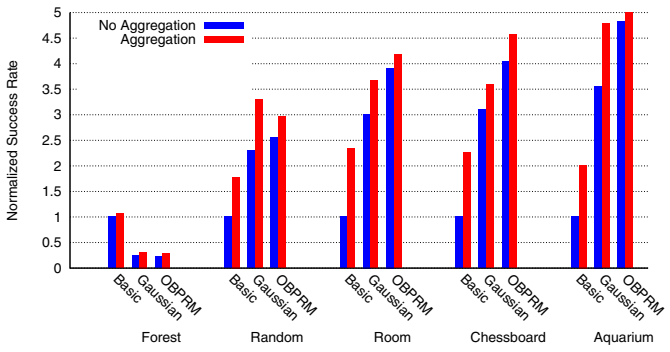


Fig. 5. Normalized sampling success rate of different aggregation methods with various underlying planners for each environment. Data is normalized to the sampling success rate of BasicPRM without the aggregation framework in each environment.

underlying sampling method. By identifying regions freed between aggregation levels and focusing sampling there, the framework boosts the success rate for all the planners.

V. Conclusions

In this paper, we propose a hierarchical aggregation framework that groups obstacles in the environment to improve the performance of sampling-based motion planners. We use the hierarchy to characterize the evolution of free space in the environment based on the distance between the obstacles. The improvement in performance is obtained through restricting the sampling to the newly revealed free regions between the consecutive levels in the hierarchy starting from the most aggregated level and going down until the planner solves the problem. Our results show that our approach shows significant performance improvements in time and in sampling success rate.

References

[1] J.-M. Lien, S. Rodriguez, J.-P. Malric, and N. M. Amato, "Shepherding behaviors with multiple shepherds," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, April 2005, pp. 3413–3418.

[2] O. B. Bayazit, G. Song, and N. M. Amato, "Enhancing randomized motion planners: Exploring with haptic hints," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2000, pp. 529–536.

[3] X. Tang, B. Kirkpatrick, S. Thomas, G. Song, and N. M. Amato, "Using motion planning to study RNA folding kinetics," in *Proc. Int. Conf. Comput. Molecular Biology (RECOMB)*, 2004, pp. 252–261.

[4] T. Lozano-Pérez and M. A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *Communications of the ACM*, vol. 22, no. 10, pp. 560–570, October 1979.

[5] L. E. Kavraki, P. Švestka, J. C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Automat.*, vol. 12, no. 4, pp. 566–580, August 1996.

[6] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *Int. J. Robot. Res.*, vol. 20, no. 5, pp. 378–400, May 2001.

[7] J. H. Reif, "Complexity of the mover's problem and generalizations," in *Proc. IEEE Symp. Foundations of Computer Science (FOCS)*, San Juan, Puerto Rico, October 1979, pp. 421–427.

[8] N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones, and D. Vallejo, "OBPRM: an obstacle-based PRM for 3d workspaces," in *Proceedings of the third Workshop on the Algorithmic Foundations of Robotics*. Natick, MA, USA: A. K. Peters, Ltd., 1998, pp. 155–168, (WAFR '98).

[9] V. Boor, M. H. Overmars, and A. F. van der Stappen, "The Gaussian sampling strategy for probabilistic roadmap planners," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, vol. 2, May 1999, pp. 1018–1023.

[10] D. Hsu, T. Jiang, J. Reif, and Z. Sun, "Bridge test for sampling narrow passages with probabilistic roadmap planners," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*. IEEE, 2003, pp. 4420–4426.

[11] S. Karaman and E. Frazzoli, "Incremental sampling-based algorithms for optimal motion planning," in *Proceedings of Robotics: Science and Systems*, Zaragoza, Spain, June 2010.

[12] S. A. Wilmarth, N. M. Amato, and P. F. Stiller, "MAPRM: A probabilistic roadmap planner with sampling on the medial axis of the free space," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, vol. 2, 1999, pp. 1024–1031.

[13] J.-M. Lien, S. Thomas, and N. Amato, "A general framework for sampling on the medial axis of the free space," in *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, vol. 3, sept. 2003, pp. 4439–4444.

[14] J. Rossignac and P. Borrel, *Multi-resolution 3D approximations for rendering complex scenes*. Springer, 1993.

[15] S. Har-Peled, *Geometric Approximation Algorithms*. Boston, MA, USA: American Mathematical Society, 2011.

[16] R. Xu, D. Wunsch *et al.*, "Survey of clustering algorithms," *Neural Networks, IEEE Transactions on*, vol. 16, no. 3, pp. 645–678, 2005.

[17] M. Parimala, D. Lopez, and N. Senthilkumar, "A survey on density based clustering algorithms for mining large spatial databases," *International Journal of Advanced Science and Technology*, vol. 31, no. 1, 2011.

[18] D. Joshi, L.-K. Soh, and A. Samal, "Redistricting using constrained polygonal clustering," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 24, no. 11, pp. 2065–2079, Nov 2012.

[19] J. Xie, L. Zhang, J. Li, H. Wang, and L. Yang, "Automatic simplification and visualization of 3d urban building models," *International Journal of Applied Earth Observation and Geoinformation*, vol. 18, pp. 222–231, 2012.

[20] S. Wang and C. F. Eick, "A polygon-based clustering and analysis framework for mining spatial datasets," *GeoInformatica*, vol. 18, no. 3, pp. 569–594, 2014.

[21] X. Li, T. W. Woon, T. S. Tan, and Z. Huang, "Decomposing polygon meshes for interactive applications," in *Proceedings of the 2001 Symposium on Interactive 3D Graphics*, ser. I3D '01. New York, NY, USA: ACM, 2001, pp. 35–42. [Online]. Available: http://doi.acm.org/10.1145/364338.364343

[22] J. P. Van den Berg and M. H. Overmars, "Using workspace information as a guide to non-uniform sampling in probabilistic roadmap planners," *The International Journal of Robotics Research*, vol. 24, no. 12, pp. 1055–1071, 2005.

[23] H. Kurniawati and D. Hsu, "Workspace-based connectivity oracle: An adaptive sampling strategy for prm planning," in *Algorithmic Foundation of Robotics VII*, ser. Springer Tracts in Advanced Robotics, S. Akella, N. Amato, W. Huang, and B. Mishra, Eds. Springer Berlin Heidelberg, 2008, vol. 47, pp. 35–51.

[24] E. Plaku, L. Kavraki, and M. Vardi, "Motion planning with dynamics by a synergistic combination of layers of planning," *Robotics, IEEE Transactions on*, vol. 26, no. 3, pp. 469–482, June 2010.

[25] N. M. Amato and Y. Wu, "A randomized roadmap method for path and manipulation planning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 1996, pp. 113–120.

[26] Harshvardhan, A. Fidel, N. M. Amato, and L. Rauchwerger, "The STAPL Parallel Graph Library," in *Languages and Compilers for Parallel Computing*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012, pp. 46–60.