Proceedings of the 1st International Workshop on Robot Learning and Planning (RLP 2016)
in conjunction with 2016 Robotics: Science and Systems
June 18, 2016 – Ann Arbor, Michigan, USA

The Impact of Approximate Methods on Local Learning in Motion Planning

Diane Uwacu, Chinwe Ekenna, Shawna Thomas, Nancy Amato

***Abstract*—Machine learning methods have been applied to many motion planning algorithms including probabilistic roadmap methods (PRM). There are many variants of these methods and choosing the best one every time is hard and depends on local properties of the environment. A successful learning approach has been developed to offset this issue. This learning approach was applied to PRMs to help decide intelligently what method to utilize in dynamically created local regions of the environment or task space. It used exact neighbor finding approaches and removed the need to partition environments to get improved results.**

In this work we make further advances by introducing approximate neighbor finder methods. It has been established that approximate neighbor finding methods are faster than exact methods, still work well in connecting nodes to edges in PRMs, and that connection is robust to noise. We study what happens when noise is introduced into learning by using approximate methods instead of already studied exact methods. We show that the impact of noise on learning depends on how much learning needs to take place given the topology of the environment. Our results demonstrate a correlation between heterogeneity and the need for learning over a local region.

I. Introduction

Motion planning finds a valid path for an object as it moves from one position to the next while avoiding obstacles in the environment. The motion planning problem has been shown to be P-SPACE hard [19], and so to mitigate this issue,

sampling based motion planning algorithms were introduced. These methods sample important points in the environment while still maintaining probabilistic completeness and in some cases asymptotic optimality [9]. This optimality depends on being able to use the right strategy for the right type of problem [3]. However, interesting problems are usually not homogeneous in nature and so there has been no single strategy that works well enough with all of them. In fact, there are many variants of motion planning strategies and choosing the best one is hard and problem dependent.

This issue motivated the introduction of machine learning methods in motion planning. Various methods have been introduced that utilize different machine learning methods. The Lightning framework in [5] introduces learning from experience where the robot's performance is stored in a hash table and comparisons made with current performance to see if there is an optimization. [1] used inverse reinforcement learning to observe how robots behave and improve or copy them in the next iteration termed "learning from demonstration". More recently machine learning was introduced to PRMs to help improve the connection of nodes in a timely manner [10].

Adaptive Neighbor Connection in local regions (ANC-Spatial) [11] makes use of reinforcement learning techniques to help decide what connection method is suitable in a particular region of the environment that gets partitioned on the fly. ANC-Spatial creates these regions by identifying neighbors to the nodes that will be added to the roadmap. It creates a dynamic region around them, after which it gets information from these neighbors on the performance of past connection methods. It uses this information to decide the next suitable method to use. These neighbor finding approaches that ANC-Spatial uses however are all exact. In this work we investigate the effect of using approximate methods instead.

We introduce approximate neighbor finding approaches which has shown to be faster and the state of the art choice for connection. We investigate what happens when approximate methods are used and how this introduction of noise into the ANC-Spatial framework affects learning. We discuss how good the neighbor finding information using this approximate approach has to be for ANC-Spatial to learn from. We find that the impact of noise is somewhat problem dependent. We show that the greater the heterogeneity, the more important it is to learn from a local set of neighbors, but this set need not be exact. On the other hand, when the critical portions of the environment lie in homogeneous regions, exact (or even approximate) learning is not needed but a global learning approach may be used.

II. Related Work

In this section we first explain the use of nearest neighbor finders in sampling-based strategies. We also review some of the approximation methods that have been studied.

A. *Sampling Based Motion Planning*

Sampling based algorithms, probabilistic roadmaps (PRMs) in particular, randomly sample the environment and then connect nodes to construct a map that is queried to find a path from a start to a goal [14]. There are a number of strategies that take advantage of the topology of the environment to optimize the roadmap construction phase of planning. PRM variants consider different topology which include uniformly generating samples in the environment [14], sampling near obstacles [2], [4], [6], [12], [20], sampling with constraints placed on the robots [17] and planning with uncertainty in the environment [13]. These methods have different characteristics and representation of the geometry of the robot in the planning space.

B. *Existing Connection Methods*

Connection methods are primitives used in PRM to connect nodes via edges together while building a roadmap. The connection method defined within our context comprises a combination of a distance metric to get calculations on distance between configurations, a neighbor finding approach to pair "close/similar" configurations and a local planner to make an edge if a feasible path exists between two configurations. In this section we discuss these three primitives used during the connection phase for PRMs, i.e. neighbor finding methods, distance metrics and local planners.

1) Exact Neighbor Finding Methods:

- K-Closest /R-Closest methods: returns the k closest neighbors to a node based on some distance metric, where k is normally some small constant, and can be done in logarithmic time. The advantage is that nodes are more likely to be connectable by the local planner because the volume of C-Space that the connection occupies is smaller. A similar approach is the r-closest method which returns all neighbors within a radius r of the node as determined by some distance metric. Here, the size of the neighbor set is not fixed but is dependent on the sampling density.
- Randomized K-Closest variants: Two randomized variants of these methods are proposed in [16]: K-Closest,K-Rand and R-Closest,K-Rand. K-Closest,K-Rand randomly selects k neighbors from the k_2 closest nodes, where typically $k_2 = 3k$. R-Closest,K-Rand selects k random neighbors from those within a distance r. In some cases, these methods outperform K-Closest as they introduce some useful randomness.

2) Approximate Neighbor Finding Methods:

- Spill-tree-based nearest neighbor search [15] uses metric tree data structures to sperate the plane into a left and right hand side. The algorithm then uses non-backtracking search to accurately return the nearest neighbor in $O(logn)$ by always searching one side of the tree. However, there is a chance of making the wrong decision when a node is at the separation point between the two sides. The algorithm fixes the issue by including a overlaping buffer of size τ that guarrantees that all the nodes that are not checked are at least τ away from the node.
- The Distance-based Projection onto Euclidean Space algorithm (DPES) [18] uses distance-based projection of high-dimensional metric spaces onto low-dimensional euclidean spaces. This projection improves efficiency because typically, not so many distance evaluations are needed to compute nearest neighbors. [18] make use of the approximate nearest neighbor algorithm (ANN) that uses kd-trees to find nearest neighbors in $O(dnlogn)$.
- Another technique MPNN [21] is an extension of the ANN algorithm. MPNN is used to find approximate nearest neighborhood by building multiple small kd-tree throughout the C-space.
- In this work we use CGAL (Computational Geometry Algorithms Library). This method has shown competence by saving time without losing correctness [7]. CGAL has different search data structures like k-d trees and range and segment trees that allow it to find the nearest neighbors in $O(logn)$.

3) Distance Metrics: A distance metric is a function δ that computes some "distance" between two configurations $a = \langle a_1, a_2, \ldots, a_d\rangle$ and $b = \langle b_1, b_2, \ldots, b_d\rangle$, i.e., $\delta(a, b) \rightarrow \mathbb{R}$, where d is the dimension of a configuration. A good distance metric for PRMs predicts how likely a pair of nodes can be connected. In this paper, we study two different distance metrics:

- The scaled Euclidean distance metric is a variant

  $$\delta(a,b) = \sqrt{s(\text{pos\_mag})^2 + (1-s)(\text{ori\_mag})^2}$$

  where pos_mag is the Euclidean distance of the positional dimensions, ori_mag is the Euclidean distance of orientational dimensions, and s is a weighting parameter. The Euclidean distance metric gives equal weighting for all dimensions. Scaled Euclidean is cheap and adequate in many situations. However, it is not a good predictor when the local planner differs from a simple straight line.
- Swept volume is the volume generated by the continuous motion (translation and/or rotation) of a geometric object through space. The swept volume distance is the volume swept by the robot while following the motion prescribed by the local planner. For an articulated linkage, this becomes the sum of the swept volumes of each of the links. This distance measure is expensive but accurate for any local planner.

4) Local Planners: A local planner (*LP*) connects two nodes with an edge based on defined closeness characteristics [3]. There are many local planners that can be used to connect two nodes a and b, and in this work we focused on two: *StraightLine and RotateAtS*. We used the StraightLine local planner because apart from being commonly used, it is the fastest to compute and thus less computation overhead. RotateAtS is useful as a comparison tool because it is an offshoot of the straightline local planner but does some rotation along the way. We also talk briefly about other local planning method called *TransformAtS and Toggle LP*.

- StraightLine [3]: interpolates between two points in C-Space checking intermediate points on a straight line in C-Space. Although this local planner is simple and fast, it usually fails in cluttered environments where nearest neighbors cannot be connected by a straight line due to the large swept volume.
- RotateAtS [3]: reduces the swept volume near endpoints by translating from a for some distance s toward b, changing all orientation degrees of freedom, and translating again to get to b. The rotation allows the local planner some chance to get around obstacles making it more successful with samples that are close to obstacles.
- TransformAtS is a modification of RotateAtS that changes all degrees of freedom one by one when it gets to s.
- Toggle LP [8]: a straight-line connection between the configurations a and b is attempted. If this fails then a

third configuration n is generated that defines a triangle between a, n and b and a path will be searched within this triangle. This method extends local planning to a 2 dimensional plane of C space. Toggle LP shows a proof of disconnection if no valid path exists but there is the added overhead of generating the third node which will prove expensive as the complexity of the problem increases.

C. Local Adaptive Neighborhood Connection (ANC-Spatial)

ANC-Spatial learns the best connection method to use on a given node by analyzing the performance of connection methods candidates in the vicinity of that node.

Figure 1 shows an example 2D point robot environment that is largely free with a small cluttered region in the bottom left corner. The roadmap is constructed with two candidate connection methods: CM_A (blue) and CM_B (red). Overall, the most successful connection method is CM_A (indicated by a greater number of blue edges). However, in the cluttered region in the bottom left, CM_B is much more successful. When connecting node q (in green) to the roadmap, it is important to take locality into account. The local learning method, such as ANC-Spatial, wisely chooses CM_B because CM_B is much more successful in this local area.

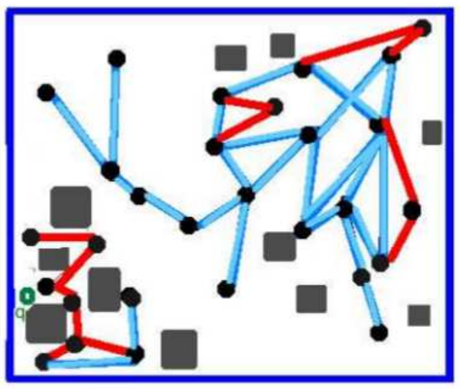

Fig. 1. To connect a new node q (in green) to the roadmap, method B (in red) is more likely to succeed than A is despite the global popularity of A.

To locate the vicinity of the node, ANC-Spatial uses a local neighbor finder that finds nearby nodes to the targetted node. ANC-Spatial analyzes the performance of the connection methods used in that region and based on that probability distribution, picks the connection method that is most likely to succeed. It then updates probabilities of the connection methods based on the performance of the chosen method. More details are presented in Algorithm 1.

Even if ANC-Spatial optimizes the choice of the connection method, its main overhead is the time it takes to find neighbors. In order to accurately learn, ANC-Spatial relies on a good local neighbor finder to return the nearest neighbors to the node. However, because of how frequent this operation is done, ANC-Spatial also relies on a fast neighbor finding algorithm.

Algorithm 1 ANC-Spatial

```
Input. A connecting vertex $q$, a set of connection methods $CM$, a locality neighbor finder $Nf_{local}$, a local planner $lp$ and a graph $G$
Output. A graph $G$ with additional edges to/from $q$.
1:  Initialize a set of connection method probabilities $P_q$ to the uniform distribution
2:  Initialize $N_{init}$ = The set of neighbors to $q$ using $Nf_{local}$
3:  for each $n \in N_{init}$ do
4:      Update $P_q$ using all $< cm, reward, cost >$ tuples stored in $n$, $\forall cm \in CM$
5:  end for
6:  Randomly select a $cm_q$ according to $P_q$
7:  $N = cm_q$.FIND_ NEIGHBORS($q, G$)
8:  for each $n \in N$ do
9:      if $lp$.IS_CONNECTABLE($q$, $n$) then
10:         $G$.ADD_EDGE($q$, $n$)
11:     end if
12:     Let $x_q$ be the reward and $c_q$ be the cost of the connection attempt
13:     Update $q$ and $n$ with $< cm_q, x_q, c_q >$
14: end for
```

III. Applying Approximation in Local Region Identification

The main goal of this work is to examine how approximate neighbor finding affects the ability to learn in ANC-Spatial. Recall that local learning examines method performance history over the nearest neighbors to determine which method to connect a given node with. We show the impact of using neighbor finding approximation during the learning process. We study different levels of approximation as well as using approximate information in different parts of the connection process (i.e., learning regions, connecting nodes). Steps 2 and 7 of Algorithm 1 find k nearest neighbors to a node q. We introduce approximation by replacing the exact neighbor finding call with and approximate version in one or both of these steps. Note that the neighbor finder used to define the learning region and the one used to determine connection canidates are likely not the same, thus computations are not reused. For example, the k nearest neighbors as determined by Euclidean distance may define the learning region but the learned connection method may use LPSwept distance for deciding which connections to attempt.

A. Experimental Setup

To investigate the performance of different approximation levels in the context of learning, we construct a roadmap until a representative query is solved. This query requires the robot to traverse the entire breadth of the environment. For performance, we look at time to build the roadmap. Running time varies based on the time required to find nearest neighbors and the affect of deteriorating neighbor quality as approximation level increases. For all experiments, we use OBPRM [2] sampling, set $k = 10$, and use straightline and

RotateAtS [3] local planning. All experiments are averaged over 10 runs.

1) Environments Studied: We examine the following environments of varying heterogeneity:

- **2D Maze** (see Figure 2): This is a 2D environment composed of several narrow passages and cluttered regions. A circular robot is expected to traverse from the bottom left (in red) to the top right (in blue) of the environment. This environment has 2 degrees of freedom.
- **Planar Rooms** (see Figure 3): This planar environment has a long rectangular rigid robot in a heterogeneous environment containing 8 different inter-connected rooms of different types including cluttered, free, and blocked regions. The robot must traverse each room to solve the query from a placement in the bottom left room to a placement in the top left room. A sample solution path is shown. This environment has 3 degrees of freedom.
- **3D Tunnel** (see Figure 4): This environment is composed of two types of regions: free regions at the top and bottom of the environment and a series of narrow passages (with dead-ends) connecting them. A spinning top-shaped robot must move from the top left to the bottom right. This environment has 6 degrees of freedom.

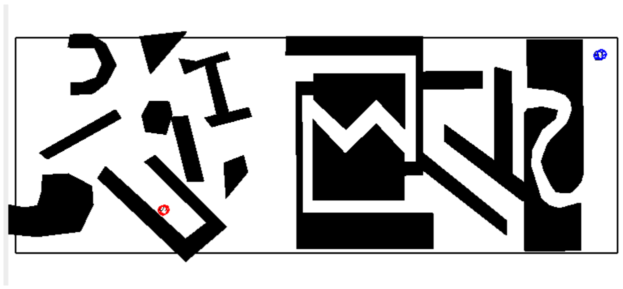
Fig. 2. 2D Maze: The circular robot must traverse from the lower left corner (in red) to the upper right corner (in blue) of the environment.

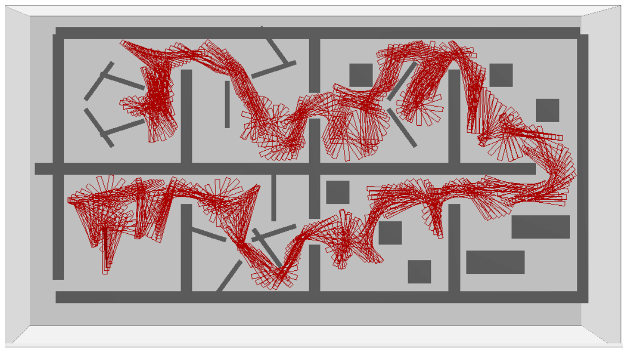
Fig. 3. Planar Rooms: A long rectangular rigid body robot must traverse each room of varying topologies to solve the query. A sample path is shown in red.

2) Methods Used: Recall that neighbor finders are used in two places: inside a connection method to determine which nodes to attempt connections (Connection NF) and inside the spatial learner to determine the local learning region (Local NF). We use the following neighbor finders:

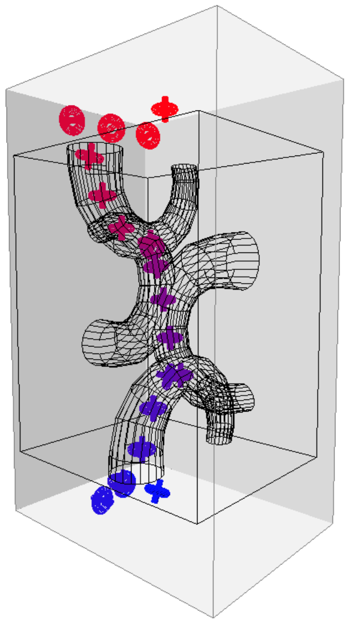
Fig. 4. 3D Tunnel: The robot must travel the length of the narrow passage to reach the free area at the bottom from the free area at the top.

- **Exact** — returns the exact k nearest neighbors as determined by some distance metric, implemented by CGAL with $\epsilon = 0$.
- **Approximate** — returns the approximate k nearest neighbors as determined by some distance metric, implemented by CGAL with $0 < \epsilon \leq 1$. Values for ϵ are explored with a stepsize of 0.25.
- **Random** — returns a random set of k neighbors irrespective of their proximity.

For a given neighbor finder, we must specify a distance metric. In these results we use Scaled Euclidean with $s = \{0.2, 0.5, 0.8\}$ and LPSwept. Note that Scaled Euclidean with $s = 0.5$ is the traditional Euclidean distance. Table I shows the combinations of methods used.

IV. Results and Discussion

We examine each environment in order of increasing complexity.

A. 2D Maze

We first establish the impact of approximate neighbor finding on the basic PRM algorithm. Here, no learning is performed. These correspond to the Basic * entries in Table I. We vary ϵ between 0 and 1 with a stepsize of 0.25. Recall that $\epsilon = 0$ is equivalent to exact nearest neighbors.

Figure 5 summarizes the performance for each distance metric for both local planners. The robot is circular, thus rotational degrees of freedom are not needed. For all approximate computations ($\epsilon > 0$), LPSwept is the slowest as the extra computations it incurs are unnecessary. LPSwept rankw

TABLE I

COMBINATIONS OF NEIGHBOR FINDER METHODS USED. ϵ_0 IS THE BEST PERFORMING VALUE > 0.

Name	Local NF	Connection NF
Basic Exact(dm)	n/a	CGAL($\epsilon = 0$, dm)
Basic Approx(dm)		CGAL($\epsilon = \epsilon_0$, dm)
Basic Random		Random
Exact-Exact	CGAL($\epsilon = 0$, Euclidean)	{CGAL($\epsilon = 0$, Euclidean), CGAL($\epsilon = 0$, Scaled Euclidean), CGAL($\epsilon = 0$, LPSwept)}
Approx-Exact	CGAL($\epsilon = \epsilon_0$, Euclidean)	
Random-Exact	Random	
Exact-Approx	CGAL($\epsilon = 0$, Euclidean)	{CGAL($\epsilon = \epsilon_0$, Euclidean), CGAL($\epsilon = \epsilon_0$, Scaled Euclidean), CGAL($\epsilon = \epsilon_0$, LPSwept)}
Approx-Approx	CGAL($\epsilon = \epsilon_0$, Euclidean)	
Random-Approx	Random	

neighboring nodes in the same way as Euclidean but will take longer to compute. Note that the extra time to compute swept volume distances is mitigated by the fact that CGAL's k-d tree greatly reduces the number of distance computations required. Thus, the increase in performance is lessened than with typical brute force approaches that compute all distances afresh with each nearest neighbor request.

In general, nearest neighbor computation is relatively quick here, so less is gained by approximation. In fact, for this environment, approximation runs slower as it requires more samples to solve the query. The best-performing, non zero approximation factor, ϵ_0, is 0.25. This value will be used as the fixed approximation factor for the remaining experiments on this environment.

We next look at the impact of approximation on learning. The first 3 bars in Figure 6 report the performance of different neighbor finders for learning given exact connection methods to choose from. Here, learning takes place over all possible combinations of exact distance metrics and local planners to provide a larger set of connection methods to learn over. We see that approximate computation is as fast as exact. This is an improvement from experiments with basic PRM, see Figure 5 This environment has some non-homogeneous regions. This causes learning over random neighbors to perform poorly as it cannot track the locality of good connection method decisions.

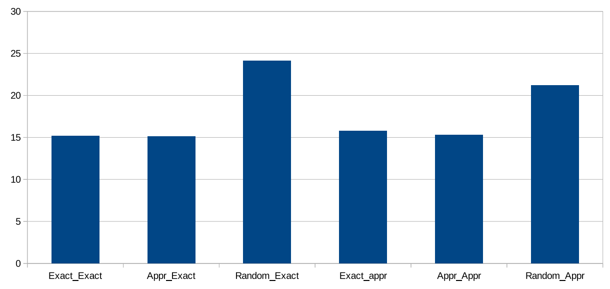

Fig. 6. Performance of different neighbor finders in learning and connection in the 2D maze. $\epsilon_0 = 0.25$.

The last 3 bars in Figure 6 report the time required when approximation is introduced into the connection methods. We observe the same trend as in the first set, namely that learning is useful (i.e., learning over random neighbors performs poorly) but that learning may be approximate. We also see a slight synergy between selecting exact or approximate methods for learning and for connection as the Approx-Approx combination performs better than Exact-Approx.

B. Planar Rooms

Results for the basic PRM without learning are reported in Figure 7. We see that using the straightline local planner results in far faster roadmap construction times. Since RotateAtS already has a hard time solving this problem, adding noise in the returned nearest neighbors worsens the situation. We see that increasing ϵ only increases running times. In this environment, ϵ = 0.25 will be used in the learning set.

Figure 8 shows results in the learning set. In this environment the random neighbor finder manages to compete with the other two methods. We see that when learning on exact neighbor finding methods (first 3 bars), the three local neighbor finders perform similarly. For the last 3 bars (approximate methods during connection), we see a slight increase for exact and approimate methods, but random stays consistent.

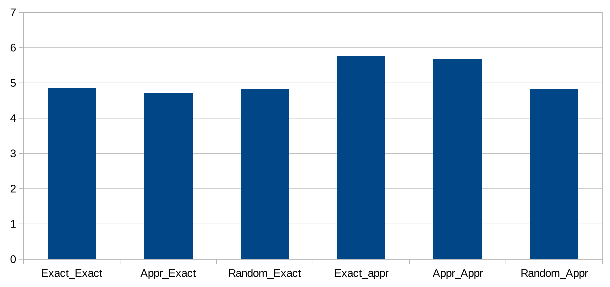

Fig. 8. Performance of different neighbor finders in learning and connection in the 2D heterogeneous maze. $\epsilon_0 = 0.25$.

C. 3D Tunnel

Again, we first establish the impact of approximate neighbor finding on the basic PRM algorithm without learning. Figure 9 shows the performance of various approximation levels using the different distance metrics for both local planners. As the tunnels are smooth in nature with thick walls surrounding them, it is easier to connect samples that are somewhat close together, but not necessarily needing to be the exact closest. Thus, this environment can not only tolerate more noise, it

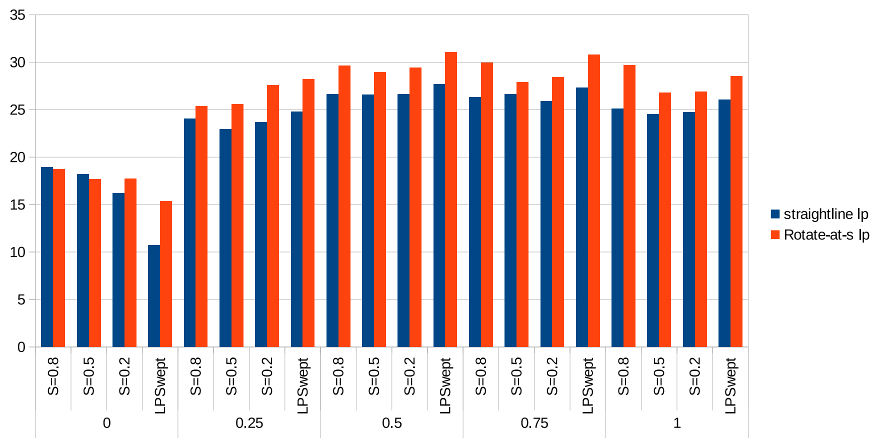

Fig. 5. ϵ study for various distance metrics in the 2D Maze for both local planners studied.

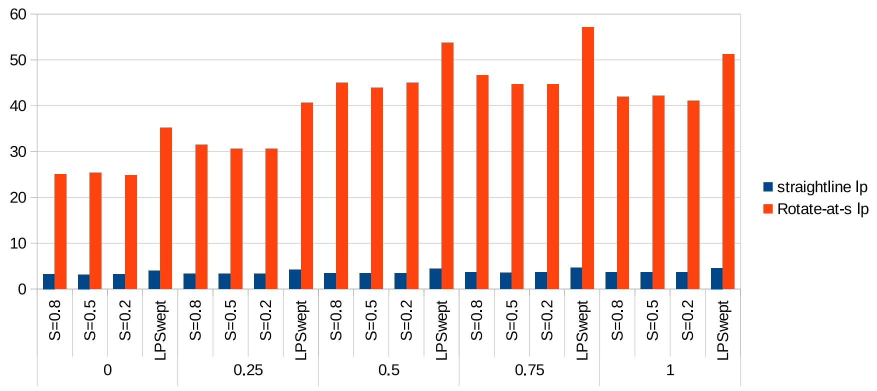

Fig. 7. ϵ study for various distance metrics in the Planar Rooms for both local planners studied.

greatly benefits from the faster neighbor finding times. Here, the best approximation level (ϵ_0) is 1.

The first 3 bars in Figure 10 display the running times of different neighbor finders for learning given exact connection methods to choose from. Recall that noise in identifying neighbors for connection is tolerated much more here. Thus, it is not surprising noise is tolerated in learning regions. In fact, for connecting most of the critical nodes (i.e., those in the tunnel), they lie in homogeneous regions (i.e., a narrow passage with thick surrounding obstacle space). Here there is not as much to learn, so selecting your learning region may even be random.

The last 3 bars in Figure 10 show the performance of learning when the underlying connection methods are approximate. A similar trend is seen for these as for learning over exact methods. When the critical portions of the environment are homogeneous, learning from local neighbors (instead of random neighbors) is not required.

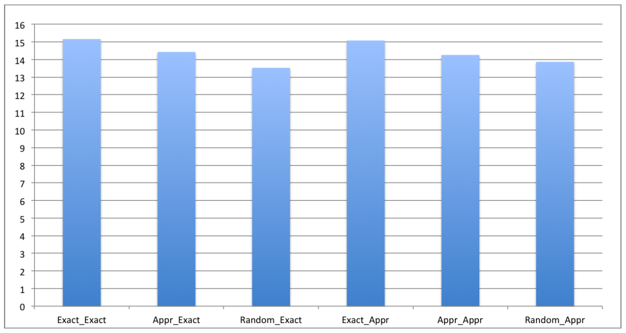

Fig. 10. Performance of different neighbor finders in learning and connection in the 3D Tunnel. $\epsilon_0 = 1$.

V. Conclusion

In this work we presented an analysis of the effect of using approximate neighbor finders in the local region learning framework ANC-Spatial. Given that the main bottleneck of ANC-Spatial is finding nearest neighbors, we investigated the benefits of allowing some level of noise to increase efficiency. Our study shows that different environments can tolerate different levels of noise (approximation) in either the neighbors identified for learning or in the neighbors identified for connection. We found that the greater the heterogeneity, the more important it is to learn from a local set of neighbors (i.e., not a random set) but that this set need not be exact. When the critical portions of the environment lie in homogeneous

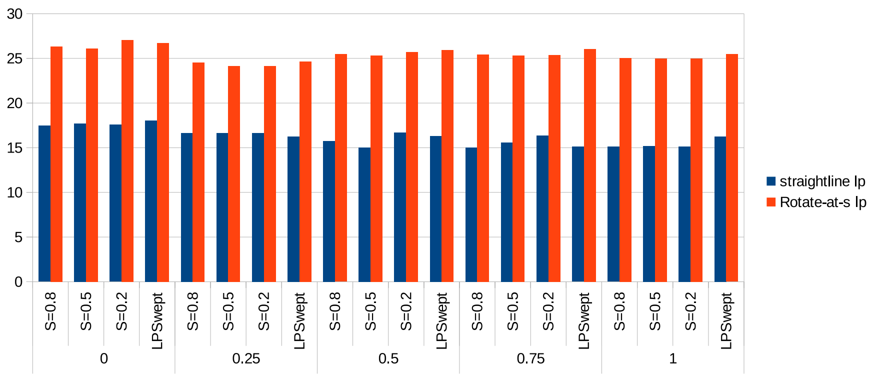

Fig. 9. ϵ study for various distance metrics in the 3D Tunnel for both local planners studied.

regions, exact (or even approximate) learning is not needed and learning over a random set (or the global set) is sufficient. In the future we plan to analyze other approximate methods that use different data structures and to examine additional types of environments.

References

[1] P. Abbeel, D. Dolgov, A. Y. Ng, and S. Thrun. Apprenticeship learning for motion planning with application to parking lot navigation. In *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, pages 1083–1090, Nice, France, September 2008.

[2] 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*, pages 155–168, Natick, MA, USA, 1998. A. K. Peters, Ltd. (WAFR '98).

[3] N. M. Amato, O. B. Bayazit, L. K. Dale, C. V. Jones, and D. Vallejo. Choosing good distance metrics and local planners for probabilistic roadmap methods. *IEEE Trans. Robot. Automat.*, 16(4):442–447, August 2000.

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

[5] D. Berenson, P. Abbeel, and K. Goldberg. A robot path planning framework that learns from experience. In *In the proceedings of the International Conference on Robotics and Automation (ICRA)*, 2012.

[6] 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)*, volume 2, pages 1018–1023, May 1999.

[7] S. Chaudhry, X. Xu, and N. Amato. Nearest neighbor search methods. Technical Report, Texas A&M University, College Station, TX, 2008.

[8] J. Denny and N. M. Amato. The toggle local planner for sampling-based motion planning. In *IEEE International Conference on Robotics and Automation, ICRA 2012, 14-18 May, 2012, St. Paul, Minnesota, USA*, pages 1779–1786, 2012.

[9] A. Dobson and K. E. Bekris. Sparse roadmap spanners for asymptotically near-optimal motion planning. *International Journal of Robotics Research*, 2013.

[10] C. Ekenna, S. A. Jacobs, S. Thomas, and N. M. Amato. Adaptive neighbor connection for PRMs: A natural fit for heterogeneous environments and parallelism. In *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, pages 1–8, Tokyo, Japan, November 2013.

[11] C. Ekenna, D. Uwacu, S. Thomas, and N. M. Amato. Improved roadmap connection via local learning for sampling based planners. In *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, pages 3227–3234, October 2015.

[12] 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)*, pages 4420–4426, 2003.

[13] L. Jaillet, J. Hoffman, J. van den Berg, P. Abbeel, J. M. Porta, and K. Goldberg. EG-RRT: Environment-guided random trees for kinodynamic motion planning with uncertainty and obstacles. In *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, 2011.

[14] 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.*, 12(4):566–580, August 1996.

[15] T. Liu, A. W. Moore, A. Gray, and K. Yang. An investigation of practical approximate nearest neighbor algorithms. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, pages 825–832, Cambridge, Massachusetts, 2005. MIT Press.

[16] T. McMahon, S. Jacobs, B. Boyd, L. Tapia, and N. M. Amato. Evaluation of the k-closest neighbor selection strategy for prm construction. Technical Report TR12-002, Texas A&M, College Station Tx., 2011.

[17] T. McMahon, S. Thomas, and N. M. Amato. Motion planning with reachable volumes. Technical Report 13-001, Parasol Lab, Department of Computer Science, Texas A&M University, Jan. 2013.

[18] E. Plaku and L. Kavraki. Quantitative analysis of nearest-neighbors search in high-dimensional sampling-based motion planning. In *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, July 2006.

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

[20] H.-Y. C. Yeh, J. Denny, A. Lindsey, S. Thomas, and N. M. Amato. UMAPRM: Uniformly sampling the medial axis. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 5798–5803, Hong Kong, P. R. China, June 2014.

[21] A. Yershova and S. M. LaValle. Improving motion-planning algorithms by efficient nearest-neighbor searching. *IEEE Trans. Robot. Automat.*, 23(1):151–157, 2007.