LLDM: Locally linear distance maps for robot motion planning

EXTENDED ABSTRACT

Josiah Putman, Lisa Oh, Luyang Zhao, Evan Honnold, Galen Brown, Weifu Wang, Devin Balkcom

Abstract— **This paper presents a data structure that summarizes distances between configurations across a robot configuration space, using a binary space partition whose cells contain parameters used for a locally linear approximation of the distance function. Querying the data structure is extremely fast, particularly when compared to graph search required for querying Probabilistic Roadmaps, and memory requirements are promising. The paper explores the use of the data structure constructed for a single robot to provide a heuristic for challenging multi-robot motion planning problems. Potential applications also include the use of remote computation to analyze the space of robot motions, which then might be transmitted on-demand to robots with fewer computational resources.**

As greater computational resources become available through large-scale clusters and cloud computing, the question arises of how to leverage those resources to allow fast and effective planning on a remote robot with far fewer resources. This paper takes one approach to the problem, finding approximate *compressed representations of optimal motion* that can then be stored, transmitted, and used within a tight computational budget. In particular, the paper discusses how to build and make use of a cell-based decomposition called a Locally Linear Distance Map (LLDM), where each cell contains a linear approximation of the value function.

We are particularly interested in motions that are optimal with respect to time cost, energy, precision, sensor coverage, or other objectives. In order to create a data structure that represents optimal motion, we must have: a) a method for discovering information about optimal motion, b) a method for storing that information, and c) a method for extracting optimal trajectories from the data structure.

Probabilistic Roadmap (PRM) frameworks provide all three methods: points are sampled and connected to analyze motion, and used to create a graph data structure from which paths may be extracted. The PRM* [11] algorithm converges to optimal paths in the limit, but convergence proofs require samples to be placed densely enough over the space to approximately cover any potential optimal path. This density can lead to a graph that is too large to store on disk or transmit over a network, and the computational cost of the A* search to find paths grows with the number of samples.

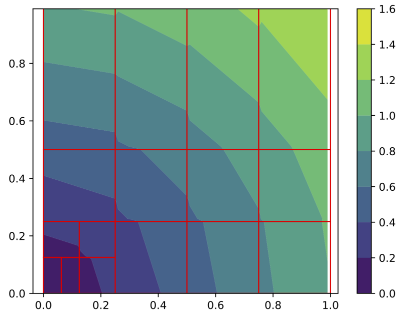

Fig. 1: An LLDM approximation of Euclidean distance to the origin.

In the present paper, the value function is constructed primarily to summarize an existing roadmap data structure. As the computation of the initial roadmap is quite expensive, a promising direction of future work is the incremental construction of the LLDM data structure without an existing graph.

The LLDM data structure may represent the cost to a single goal, or may summarize an all-pairs distance function over the space. As an initial demonstration of the potential usefulness of the very high speed queries, we make use of the LLDM as a heuristic for informed search using a traditional cell-based search method by Barraquand and Latombe [1]. While we do not claim that the informed search method we present is practically competitive with modern multi-robot planning methods, we believe that the high-speed distance function computation may serve as a useful component in future planning approaches.

The following table shows some comparison the memory required to store a PRM* data structure for a simple planar 1x1 world with point robots and polygonal obstacles, and the memory required to store an LLDM. One of the LLDM data structures was constructed from the true value function, computed using a visibility graph, and the second was constructed from the PRM* itself.

TABLE I: PRM-LLDM Comparison

Method	Memory (KB)	Max Error	Avg Error
LLDM (VG)	30.24	0.018	0.007
LLDM (PRM*)	30.24	0.134	0.016
PRM*	4412	1.472	0.024

Consider a toy problem that illustrates the challenges for PRM*, and the main insight that leads to approximation approaches. Let there be a point robot restricted to the box $[0, 1] \times [0, 1]$. For now, assume that the goal of the robot is always to reach the origin; we will relax this assumption shortly. Further assume that the true cost of reaching the origin is Euclidean: $d(x, y) = \sqrt{x^2 + y^2}$. To approximate this cost well, PRM* needs to place very many samples: for each possible starting configuration, there must be samples sufficiently close to an optimal trajectory such that the local planner can connect samples without deviating too far from the optimal.

For this toy problem, the analytical distance function may be computed quickly with high accuracy, and the formula requires little memory to store. But let us imagine that d is a black box that may only be queried at particular points, and will later be unavailable to us. Let $p_1 = (0, 0, 0)$, $p_2 = (1, 0, 1)$, and $p_3 = (1, 1, \sqrt{2})$, where the first two elements of each vector give the x and y location of each point, and the third element gives the value of the distance function computed at that point. These three points describe a plane in R^3. Given a new starting configuration (x, y), we may intersect the vertical line through (x, y) with that plane to approximate the distance function.

Of course, the further we get from the known points, the greater we expect the error to be. To mitigate this issue, we divide the space into regions, with a different linear approximation in each region. We separate the problem into *construction* and *query* phases. To *construct* the desired data structure, we will use a binary space partition (BSP) to segment the space into cells. Within each cell, we construct a linear approximation of the distance function, by sampling the true distance function at a few points within or near the cell and computing parameters that are stored in the portion of the data structure corresponding to the cell. To *query* the approximate distance function at a point, identify the cell containing the point using the BSP and compute a dot product with the parameters associated with the cell.

I. Related work

Perhaps the work closest to that proposed is on learning heuristics for robot motion planning in games by

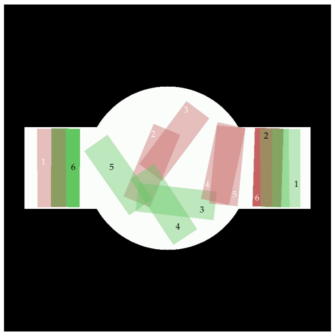

Fig. 2: Paths computed for the motion of two robots using a modified Barraquand and Latombe planner with LLDM used as heuristic.

Rayner, Bowling, and Sturtevant [20], which attempts to remap a motion problem with obstacles into a new map for which the Euclidean distance represents a provably consistent, admissible heuristic for the original problem. Network embedding problems similarly try to find a mapping that expresses distance between vertices in a network; [8] provides a recent survey.

Work on LQR trees [23] places controllers over the state space, effectively reducing the memory requirements while also achieving safety of motion, providing a promising data structure to compute and transmit to robots. Bialkowski *et al.* have reduced the time cost of collision detection with RRT*, by building balls in free (Euclidean) space in which collision detection needs to be performed only once [5]. Deits *et al.* showed a numerical optimization approach to computing large convex regions, also in a Euclidean space [9]. Early work on neural network approximations of value functions derived from optimal control includes [18]. Recent work on distance metric approximation for RRTs using supervised learning [4] is also quite close in spirit to the proposed work, and was shown to be quite effective for finding a policy for a pendulum swing-up problem.

Like the present work, Probabilistic Roadmap (PRM) algorithms ([12]) algorithms divide motion planning into learning and query phases. When optimal paths are sought, as with the PRM* algorithm [11], roadmaps can become very dense. Marble et al. [16] introduced *spanners* [19, 21, 7, 24] into the robotics community to reduce the density of PRM* roadmaps (at some cost in path optimality) [14, 17, 13, 15]. In our work [25], a modified version of more recent streaming spanner algorithms [10, 2, 3, 22, 6] achieved similar path quality to the work by Marble in seconds rather than hours. Like spanner approaches, the present work attempts to find a summary of a graph data structure, but the summary is continuous within each cell.

References

[1] Jérôme Barraquand and Jean-Claude Latombe. "Nonholonomic Multibody Mobile Robots: Controllability and Motion Planning in the Presence of Obstacles". In: *Algorithmica* 10 (1993), pp. 121–155.

[2] Surender Baswana. "Streaming algorithm for graph spanners – single pass and constant processing time per edge". In: *Inf. Process. Lett.* 106.3 (2008), pp. 110–114.

[3] Surender Baswana, Sumeet Khurana, and Soumojit Sarkar. "Fully dynamic randomized algorithms for graph spanners". In: *ACM Transactions on Algorithms* 8.4 (2012), p. 35.

[4] Mukunda Bharatheesha et al. "Distance metric approximation for state-space RRTs using supervised learning". In: *Proc. IROS*. 2014, pp. 252–257.

[5] Joshua Bialkowski et al. "Efficient Collision Checking in Sampling-Based Motion Planning". In: *Proc. WAFR*. 2012, pp. 365–380.

[6] Prosenjit Bose et al. "Robust Geometric Spanners". In: *Computing Research Repository, CoRR* abs/1204.4679 (2012).

[7] Edith Cohen. "Fast Algorithms for Constructing t-Spanners and Paths with Stretch t". In: *SIAM J. Comput.* 28.1 (1998), pp. 210–236.

[8] Peng Cui et al. "A Survey on Network Embedding". In: *CoRR* abs/1711.08752 (2017).

[9] Robin Deits and Russ Tedrake. "Computing Large Convex Regions of Obstacle-Free Space through Semidefinite Programming". In: *Workshop on the Algorithmic Foundations of Robotics (WAFR)*. Istanbul, Turkey, Aug. 2014.

[10] Michael Elkin. "Streaming and fully dynamic centralized algorithms for constructing and maintaining sparse spanners". In: *ACM Trans. Algorithms* 7.2 (Mar. 2011), 20:1–20:17.

[11] Sertac Karaman and Emilio Frazzoli. "Sampling-based Algorithms for Optimal Motion Planning". In: *The International Journal of Robotics Research* 30(7), 846-894 (2011).

[12] Lydia Kavraki et al. "Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces". In: *IEEE International Conference on Robotics and Automation*. 1996, pp. 566–580.

[13] Anthanasios Krontiis, Andrew Dobson, and Kostas Bekris. "Sparse roadmap spanners". In: *Proceedings of the workshop on the algorithmic foundations of robotics on Robotics, WAFR 2012* (2012).

[14] James D. Marble and Kostas E. Bekris. "Asymptotically Near-Optimal is Good Enough for Motion Planning". In: *Proc. of the 15th International Symposium on Robotics Research (ISRR-11)*. 28. Aug. - 1 Sep 2011.

[15] James D. Marble and Kostas E. Bekris. "Asymptotically Near-Optimal Planning With Probabilistic Roadmap Spanners". In: *IEEE Transactions on Robotics* 29.2 (2013), pp. 432–444.

[16] James D. Marble and Kostas E. Bekris. "Computing spanners of asymptotically optimal probabilistic roadmaps". In: *IROS*. 2011, pp. 4292–4298.

[17] James D. Marble and Kostas E. Bekris. "Towards small asymptotically near-optimal roadmaps". In: *ICRA*. 2012, pp. 2557–2562.

[18] Rémi Munos, Leemon C. Baird, and Andrew W. Moore. "Gradient descent approaches to neural-net-based solutions of the Hamilton-Jacobi-Bellman equation". In: *IJCNN*. 1999.

[19] David Peleg and Alejandro A. Schaeffer. "Graph spanners". In: *Journal of Graph Theory* 13.1 (1989), pp. 99–116.

[20] D. Chris Rayner, Michael H. Bowling, and Nathan R. Sturtevant. "Euclidean Heuristic Optimization". In: *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011, San Francisco, California, USA, August 7-11, 2011*. 2011.

[21] Iam Roditty, Mikkel Thorup, and Uri Zwick. "Roundtrip spanners and roundtrip routing in directed graphs". In: *ACM Trans. Algorithms* 4.3 (July 2008), 29:1–29:17.

[22] Liam Roditty. "Fully Dynamic Geometric Spanners". In: *Algorithmica* 62.3-4 (2012), pp. 1073–1087.

[23] Russ Tedrake et al. "LQR-trees: Feedback Motion Planning via Sums-of-Squares Verification". In: *I. J. Robotics Res.* 29.8 (2010), pp. 1038–1052.

[24] Mikkel Thorup and Uri Zwick. "Approximate distance oracles". In: *ACM Symposium on the Theory of Computing, STOC 2001*. Ed. by Jeffrey Scott Vitter, Paul G. Spirakis, and Mihalis Yannakakis. ACM, 2001, pp. 183–192.

[25] Weifu Wang, Devin J. Balkcom, and Amit Chakrabarti. "A fast online spanner for roadmap construction". In: *I. J. Robotics Res.* 34.11 (2015), pp. 1418–1432.