Learning Dimensional Descent planning for a highly-articulated robot arm

Paul Vernaza and Daniel D. Lee

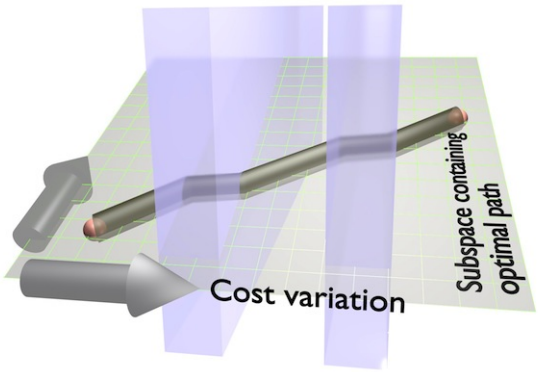


Fig. 1: Shown is an optimal path in three dimensions of a cost function that varies only in one dimension. The blue blocks represents high-cost regions, assumed to extend ad infinitum in directions orthogonal to the cost variation direction. The optimal path can be shown to lie in a two-dimensional subspace: the smallest subspace that contains both the direction of cost variation and both of the endpoints.

The challenge of motion planning for a robot with many degrees of freedom has inspired a plethora of interesting solutions, many of which have become staples of the robotics research community. In particular, sampling-based planners often succeed in producing motion plans for complex, high-dimensional problems while being very flexible and relatively simple to implement. Unfortunately, these desirable qualities come at a cost of generally low solution quality. This is particularly a concern in the problem of planning for a high-dimensional robot arm, which is the subject of the current work. Much recent work has attempted to address this issue via extensions of sampling-based planners ([1], [2], [3], [4]), while other work has focused on adapting classical deterministic search to high-dimensional settings ([5], [6]).

Our approach is based on a method we have recently developed, known as Learning Dimensional Descent (LDD [7]), that differs fundamentally from typical sampling-based and seach-based planners. The key result that motivates LDD is that optimal paths associated with low-dimensional cost functions, lie in low-dimensional subspaces, as illustrated in Fig. 1. Suppose we are given a *cost function*—i.e., a function mapping our configuration space to a scalar cost—and we wish to find a continuous path through configuration space that minimizes the cost integrated over the path. Fig. 1 illustrates a particular cost function, where high-cost regions are depicted by blue blocks. If the cost function varies only in one dimension, we can then show that any optimal path associated with this cost function, lies in the smallest subspace that contains both the direction of cost variation and the endpoints. This idea readily generalizes to vector spaces of arbitrary dimension. It is straightforward to see how knowing this subspace in advance leads to computational benefits—instead of searching over a high-dimensional space for the optimal path, we can restrict our computation to a low-dimensional subspace.

LDD implements this idea, but with adjustments that make it suitable for the typical case where low-dimensional structure is present only approximately. LDD first finds a basis W for the subspace that best captures the variation of the cost function. This is accomplished by sampling along with a simple spectral optimization, analogous to PCA. It then finds a sequence of paths $\bar{x}^k$ via the following iteration, where $J\{x\}$ is the cost of the path $x(\cdot)$ and W^k is a submatrix of W:

$$\bar{x}^{k+1}(t) = \underset{a^k(t),s(t)}{\arg\min}\, J\{W^k a^k(t) + \bar{x}^k(s(t))\}. \qquad (1)$$

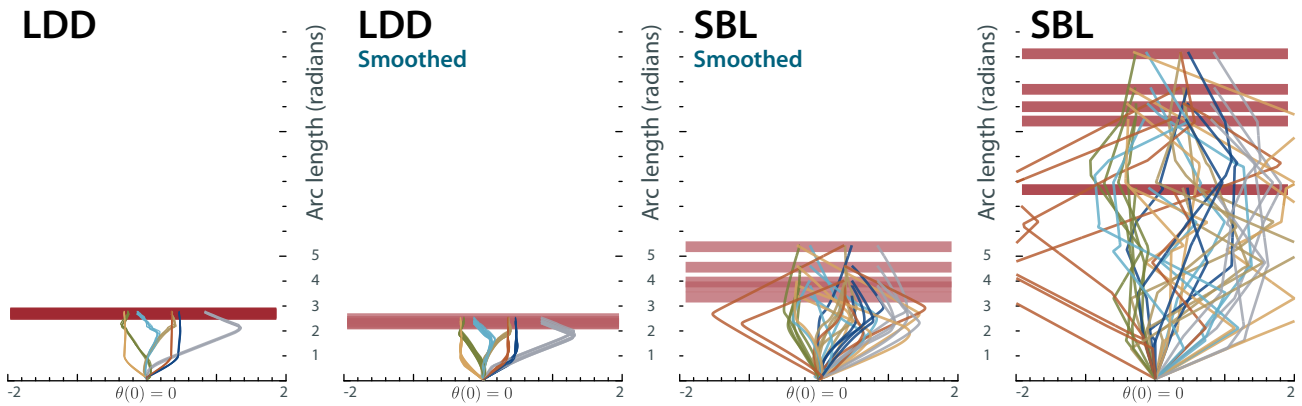
Each step in the iteration involves solving a low-dimensional dynamic programming problem. In the ideal case, this iteration will find the globally optimal solution in one step, as it will search over exactly the subspace guaranteed to contain the optimal solution (given a suitable W). Otherwise, it will cycle through the dimensions in order of decreasing importance, optimizing each in turn.

We implemented LDD in a mixture of OCaml and C++ with interfaces to ROS, the open-source Robot Operating System[1]. The resulting code is freely available online under an open-source license[2]. LDD was applied specifically to the problem of arm planning for the Willow Garage PR2 robot. LDD was compared to post-processed SBL [8] (a bidirectional sampling-based planner with lazy collision checking), using the open-source OMPL [9] implementation of the latter. Our objective was mainly to compare the consistency and quality of the solutions obtained with both methods. We used measured arc length of the seven-dimensional joint trajectories as a primary metric in order to compare the quality of the solutions.

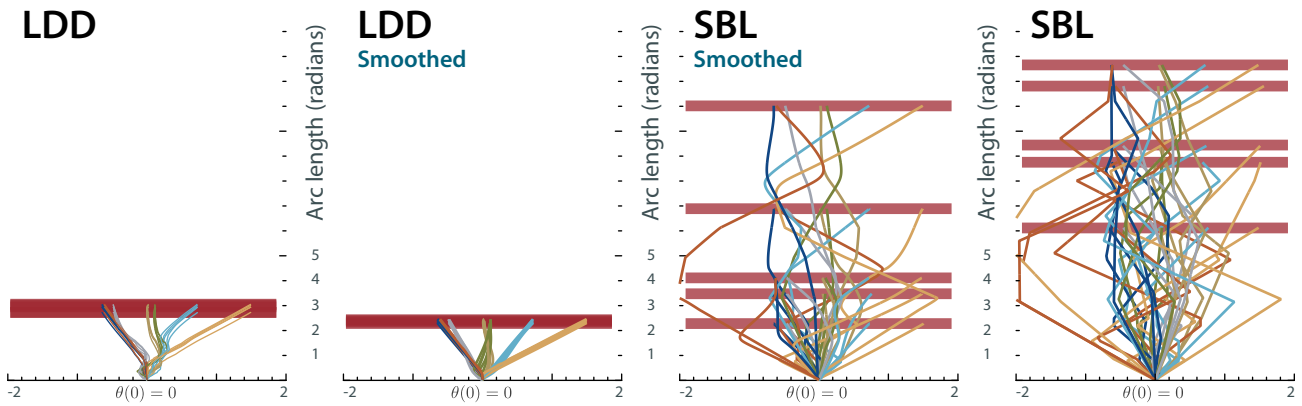
P. Vernaza <pvernaza@andrew.cmu.edu> is affiliated with the Robotics Institute, Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA 15213. D. D. Lee <ddlee@seas.upenn.edu> is affiliated with the GRASP Laboratory, University of Pennsylvania, Philadelphia, PA 19104

[1] http://www.ros.org

[2] Package penn-ros-pkgs/ldd_plan, available at https://mediabox.grasp.upenn.edu/svn



(a) *Windows* experiment



(b) *Poles* experiment

Fig. 2: Experimental comparison of repeatability and quality of trajectories obtained with LDD and SBL, with and without post-smoothing, for each of two scenarios. Each plot shows 35 joint angle trajectories: one for each matching of seven joints and five trials. Trajectories are plotted with (offset-normalized) joint angle on the horizontal axis and arc length (i.e., time) on the vertical axis. A thick horizontal line is plotted to mark the arc length at the end of each seven-dimensional trajectory.

Results are shown in Fig. 2 (see caption for detailed interpretation). Visualizing the joint trajectories bore out our observation that the LDD trajectories were usually nearly identical across trials. Smoothing the LDD trajectories also provided fairly little benefit in terms of decreasing arc length (vertical axis), indicating that these solutions were probably nearly optimal to begin with. As expected, the SBL trajectories were long and inconsistent across trials before smoothing. Smoothing these sometimes helped dramatically, but results were generally unpredictable. In both experiments, the unsmoothed LDD output almost always surpassed even the smoothed output of SBL. Finally, the consistency of SBL between trials was not obviously improved by smoothing.

We have demonstrated empirically that LDD finds high-quality plans for a robot arm operating in cluttered environments. The solutions obtained thus were of a significantly higher quality and much more consistent than those obtained using the common method of finding a feasible path with a sampling-based planner followed by post-smoothing. This provides important experimental validation of the concept that learning and exploiting low-dimensional structure is a promising avenue for further research as well as a practical method for solving difficult planning problems today.

References

[1] S. Karaman and E. Frazzoli, "Incremental sampling-based algorithms for optimal motion planning," in *RSS*, Zaragoza, Spain, June 2010.

[2] D. Ferguson and A. Stentz, "Anytime RRTs," in *IROS*. IEEE, 2006.

[3] R. Diankov and J. Kuffner, "Randomized statistical path planning," in *IROS*. IEEE, 2007, pp. 1–6.

[4] D. Berenson, T. Simeon, and S. Srinivasa, "Addressing cost-space chasms for manipulation planning," in *ICRA*, 2011.

[5] M. Likhachev, G. Gordon, and S. Thrun, "ARA*: Anytime A* with provable bounds on sub-optimality," in *NIPS*, 2004.

[6] M. Likhachev and A. Stentz, "R* search," in *AAAI*, 2008.

[7] P. Vernaza and D. D. Lee, "Learning dimensional descent for optimal motion planning in high-dimensional spaces," in *Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

[8] G. Sánchez and J.-C. Latombe, "A single-query bi-directional probabilistic roadmap planner with lazy collision checking," in *Proceedings International Symposium on Robotics Research*, 2001.

[9] "The Open Motion Planning Library (OMPL)," 2010. [Online]. Available: http://ompl.kavrakilab.org