Learning Explicit Infeasibility from Implicit Configuration Space Connectivity

Sihui Li
and Neil T. Dantam
Department of Computer Science
Colorado School of Mines
Golden, Colorado 80401, USA

***Abstract*—Sampling-based algorithms generate plans using samples from implicit representations of high dimensional configuration spaces. These samples contain unexploited information, which can be use to build a better understanding towards the configuration space's connectivity. We aim to use these samples as data to learn explicit infeasibility proofs. Previous work has shown success in up-to 4 DoF manipulation scenes. In this work, we focus on improving the learning step of the previous algorithm and show experimental results on a 5-DoF manipulator.**

I. Introduction

High dimensional motion planning problems often have an implicit representation of configuration space connectivity. For example, in sampling-based motion planners [1, 6, 11, 12, 15, 16, 19], we usually setup validity checkers to implicitly separate the free space and the obstacle region of a configuration space. For high degrees of freedom (DoF) manipulators, this representation efficiently links the 3D workspace scenes with the high dimensional configuration space.

Understanding the configuration space is important for motion planning. Earlier works have developed methods to explicitly define a configuration space [25]. However, such explicit representations are limited to lower dimensions and translational movements. Sampling-based motion planners generate samples while searching for path in the implicitly represented configuration space. These samples are data containing information of the original configuration space. The purpose of this research is to apply learning methods to the data points (configuration space samples) in order to gain a better understanding of the configuration space.

Given a motion planning problem configuration space, an important question we want to understand is whether there exists a collision free path connecting the start and the goal. A complete motion planner answers this question and produces a valid path in finite time [20]. In previous work [23], we approach this question by constructing an infeasibility proof in a configuration space using the sampled points from a sampling-based motion planner. The infeasibility proof is a closed manifold that exists in the obstacle region of the configuration space and separates the start and the goal. We first learn the manifold using the sampled points as data, then triangulate the manifold to prove its containment in the obstacle region. Existence of the infeasibility proof prevents any collision free path between the start and the goal, thus making the sampling-based motion planner complete in the limit. This algorithm scaled up-to 4-DoF manipulators.

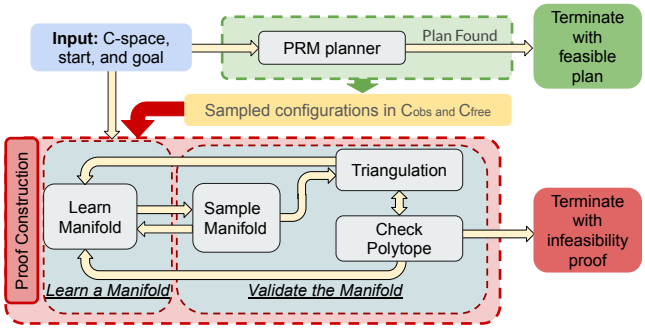


Fig. 1: Algorithm overview [24]. The red block shows the infeasibility proof construction, which runs in parallel with a PRM planner.

Our current goal is to scale our algorithm to higher dimensions. However, applying it to 5-DoF manipulators reveals issues in the learning process not present in lower-dimensional spaces. In this abstract, we generalize the previous algorithm to improve learning for higher DoF manipulators and propose future directions to scale further. Learning has previously been applied to feasibility of planning. Some works [29, 8] learn classifiers to predict motion planning feasibility from object features or visual inputs. However, these approaches produce estimations of motion feasibility, which is not a definitive answers. Our algorithm framework aims for definite infeasibility results.

II. Problem Definition

A motion planning problem consists of a configuration space $\mathcal{C}$ of dimension n, a start configuration $\boldsymbol{q}_{\text{start}}$, and a goal configuration $\boldsymbol{q}_{\text{goal}}$ [20]. The configuration space $\mathcal{C}$ is the union of the closed set obstacle region $\mathcal{C}_{\text{obs}}$ and the open set free space $\mathcal{C}_{\text{free}}$. A feasible plan is defined as a path σ such that $\sigma[0,1] \in \mathcal{C}_{\text{free}}$, and $\sigma[0] = \boldsymbol{q}_{\text{start}}$, $\sigma[1] = \boldsymbol{q}_{\text{goal}}$. The solution to the motion planning problem is a feasible plan if one exists, or an infeasibility proof if no motion plan exists.

Definition 1 (Infeasibility Proof)**.** *A manifold $\mathcal{M}$ in $\mathcal{C}$ defined by a continuous function $f(\boldsymbol{q}) = 0$ is an* infeasibility proof *if and only if,*

(I) $\mathcal{M}$ *is a closed manifold,*
(II) $\mathcal{M}$ *separates the start and the goal,* $f(\boldsymbol{q}_{\text{start}})f(\boldsymbol{q}_{\text{goal}}) < 0$,
(III) $\mathcal{M}$ *is contained entirely in* $\mathcal{C}_{\text{obs}}$, $\forall f(\boldsymbol{q}) = 0,\ \boldsymbol{q} \in \mathcal{C}_{\text{obs}}$.

III. Algorithm

A. Summary of Previous Algorithm

Figure 1 shows the overall algorithm structure, which runs in parallel a sampling-based planner and the infeasibility proof construction algorithm, terminating with either a plan or an infeasibility proof. The first step is to learn a manifold using available sampled configurations in $\mathcal{C}_{\text{free}}$. Compared to [23], which used RRT-connect [21] as the sampling-based planner, this abstract uses a Probabilistic Roadmap (PRM) planner [16]. We learn the manifold by setting up a binary classification problem with the sampled $\mathcal{C}_{\text{free}}$ configurations as the training data. Configurations in the graph component containing the goal configuration are one class, and all configurations in other components are under the other class. For training, we use a support vector machine (SVM) with Radial Basis Function (RBF) kernel. BRF kernel SVM has many advantages, including large margin, analytical form boundary function, and the ability to fit any curvature with only one hyper-parameter that controls over-fitting.

After training, we sample points on the manifold for triangulation in the next step. We sample points on the manifold by solving a nonlinear optimization problem for each point. The learning process iterates between training and sampling points on the manifold. If there are manifold points in $\mathcal{C}_{\text{free}}$, we add these points back to the training data set and retrain the manifold, until all manifold points are in $\mathcal{C}_{\text{obs}}$. The top part of Figure 2 shows this process.

After learning the manifold, we must verify that it is entirely in $\mathcal{C}_{\text{obs}}$. We first triangulate the manifold with tangential Delaunay complexes [2, 3] to construct a reassembling polytope, then check that the polytope is entirely in $\mathcal{C}_{\text{obs}}$ by checking each facet of the polytope with configuration space penetration depth. If this checking succeeds, we have an infeasibility proof.

B. New Algorithm

When applying the previous algorithm on a 5-DoF manipulator, the learning process cannot train successfully. In the previous algorithm, as stated before, we add $\mathcal{C}_{\text{free}}$ manifold points back to the training set and retrain the manifold. In this process, we must categorise the $\mathcal{C}_{\text{free}}$ manifold points into either of two classes, i.e., goal component or non-goal component. We attempt to determine this by linearly interpolating the manifold points with a set of its neighbors. If the manifold point is connectable to one of its neighbors, then its class is the same as its neighbor's class. If we cannot find any connectable neighbors, then we acquire more samples from the planning graph. This process works for 4-DoF manipulator and would theoretically work for higher DoF manipulators as well. In practice, however, this linear interpolation would require dense sampling of the configuration space in order to connect all difficult-to-determine manifold points to nearby samples. In higher dimensions, this requires an exponential sampling time.

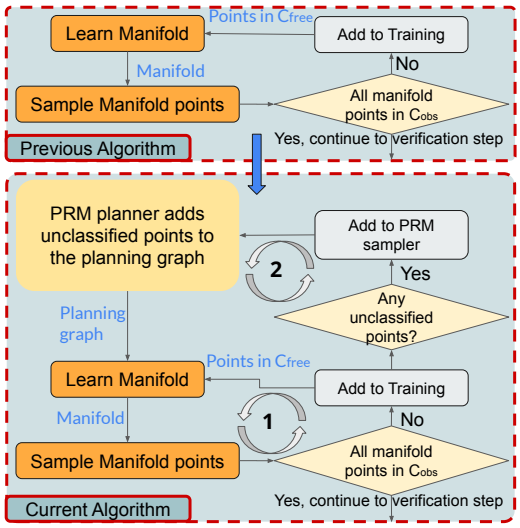


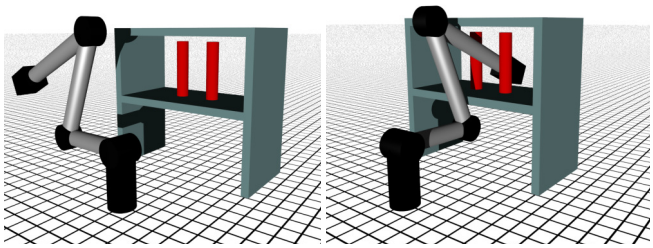
Fig. 2: Algorithm updates overview. Part of the manifold learning step uses the PRM planner to categorize points for training.

To accommodate fewer samples in higher dimension, we propose an updated algorithm for training the manifold that further couples learning and PRM sampling. Figure 2 shows this change. We define a new sampler for the PRM planner. When adding the manifold points to the training set, we save all uncategorized points to a set. During sampling, the new PRM sampler returns points from the uncategorized points set if the points are not used previously. Otherwise, the sampler performs normal uniform sampling. When the training and sampling iteration cannot improve anymore (loop 1 in Figure 2), that is, when we cannot categorize any of the $\mathcal{C}_{\text{free}}$ manifold points, the algorithm acquires more samples from the planning graph (loop 2 in Figure 2). Since the uncategorized points become samples of the PRM planner, they are marked as they are part of the planning graph. These samples may or may not be connectable to the existing PRM planning graph components. If a sample is not connectable, then it becomes a separated component of the planning graph.

The updated learning algorithm creates a stronger integration between the sampling-based motion planner and the infeasibility proof construction. On the infeasibility proof construction side, the learning step uses PRM's ability to categorize points (connect to a roadmap component, or become a separated component of the planning graph) for training. On the planner side, the new sampler employs the infeasibility proof by-product ($\mathcal{C}_{\text{free}}$ manifold points) as a sampling heuristic, which may aid in solving motion planning problems involving narrow passages. We discuss this further in future work.

IV. Preliminary Results

To test our updated algorithm, we setup a 5-DoF manipulator scene as shown in Figure 3. The goal in this scene is to reach



(a) Start scene. (b) Goal scene.

Fig. 3: Experiment scenes

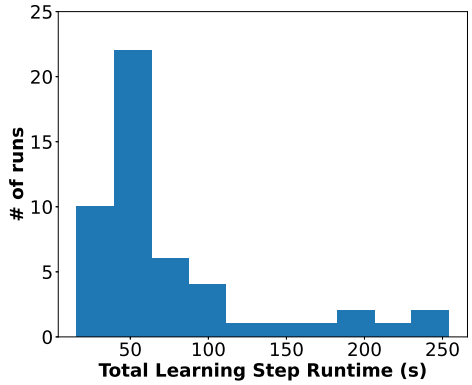


Fig. 4: Experimental result shows that the learning step for 5-DoF manipulator scenes take less than 100 seconds in most runs.

the back of the shelf from a start position outside of the shelf with some obstacles in the front, which simplifies a real-world scenario of reaching items in the back of a cluttered cabinet. The manipulator has the similar kinematics to the widely-fielded FLIR (formerly Endeavor Robotics and iRobot) 510 PackBot arm [27]. We run our experiments on a multi-core system, a dual CPU AMD EPYC 7402 with 24 cores per CPU. We adapt the PRM [16] in OMPL [5] to use the new sampler and run in parallel with our infeasibility proof construction. We solve the nonlinear optimization problems using sequential least-squares quadratic programming (SLSQP) [13, 17, 18]. We train the manifold using LIBSVM [4]. We model robot kinematics using Amino [7]. We determine ground truth for plan non-existence to high-confidence for a scene by running RRT-connect continuously for greater than 20 minutes.

As stated in Sec. III-A, when all manifold points are in $\mathcal{C}_{\text{obs}}$, we continue to the verification steps. For this preliminary experiment, we want to see how long the learning step takes for higher DoF manipulators, so we apply the updated learning algorithm to the scene in Figure 3 and train until all manifold points are in $\mathcal{C}_{\text{obs}}$. Figure 4 shows the runtime distribution among 50 runs. As we can see, most runs completed within 100 seconds. Table I shows more statistics on the runs. Training the manifold and sampling points on the manifold are the two major steps. On average, there are 61,702 points sampled on the manifold, and they are all in $\mathcal{C}_{\text{obs}}$ by the end, which is non-trivial amount of checks for the manifold's existence in $\mathcal{C}_{\text{obs}}$. This experiment only covers the learning step, we discuss improvements of the verification step as part of future work.

Runtime and Profiling Results					
	Total (s)	*Train (s)*	*Sample (s)*	*mani #*	*training #*
Mean	72.72	43.05	29.66	61,207	265,956
STD	59.24	51.94	8.94	3939	178,494

TABLE I: Runtime and profiling statistics.

V. Discussion and Future Work

On average, the training of the manifold takes about 60% of the overall runtime of the learning step. Comparing with experiments for lower-DoF manipulators, where the training time is negligible, training in 5-DoF scenes takes much longer due to a larger data set. To improve training time, one direction of future work is to explore machine learning methods that train faster and/or are more suitable for large data sets. Algorithms for training a faster SVM has been proposed in previous works [22]. Our current implementation uses single-core, CPU-based learning [4], so another possibility is to use multi-core CPUs and GPUs to accelerate training [30]. Also, training linear SVM is faster than kernel SVM and but cannot be applied to non-linear classification problems. Past work has explored methods that use a combination of linear SVMs to approximate non-linear boundaries [10, 26, 28, 31]. Neural-Network(NN) models are also possible. Recent work has been developed for training NN with large margins [9].

After the learning step, we also need to improve the verification steps. In previous experiments for lower DoF manipulators, the verification step takes the majority of overall algorithm runtime [23]. For 5-DoF manipulators, the verification step would take considerably longer using the previous algorithm. One important step of verification is the triangulation. Exploring faster triangulation methods would greatly improve our runtime. Recently, we have found that Coxeter triangulation [14] could be a good triangulation methods to use. Initial trials in 4-DoF scenes reduce the triangulation time from the scale of 10^2 seconds to around $1 \sim 5$ seconds.

Besides the above discussions for infeasibility proof construction, the proposed algorithm could also be beneficial for searching a valid path. Even though the learned manifold may not exist entirely in $\mathcal{C}_{\text{obs}}$ at the beginning, the manifold is very likely to be largely in $\mathcal{C}_{\text{obs}}$ because we train it to classify two groups of $\mathcal{C}_{\text{free}}$ points. If there exist narrow passages that connect the two classes of points, it is possible that the unclassified manifold points would quickly reveal them. Then, using the uncategorized manifold points as samples would guide the planning graph towards narrow passages to find a path. Effectiveness of this sampling heuristic in practice requires more experiments.

Acknowledgments

This work is supported in part by the National Science Foundation under Grant No. IIS-1849348. We thank Dr. Mehmet Belviranli for providing access to the multi-core server used for evaluating the presented algorithm.

REFERENCES

[1] Nancy M Amato and Yan Wu. A randomized roadmap method for path and manipulation planning. In *International Conference on Robotics and Automation1996*, volume 1, pages 113–120, 1996.

[2] Jean-Daniel Boissonnat and Arijit Ghosh. Manifold reconstruction using tangential delaunay complexes. *Discrete & Computational Geometry*, 51(1):221–267, 2014.

[3] Jean-Daniel Boissonnat, Frédéric Chazal, and Mariette Yvinec. *Geometric and Topological Inference*. Cambridge University Press, 2018. URL https://hal.inria.fr/hal-01615863. Cambridge Texts in Applied Mathematics.

[4] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[5] Ioan A. Şucan, Mark Moll, and Lydia E. Kavraki. The open motion planning library. *Robotics & Automation Magazine*, 19(4):72–82, 2012.

[6] Ioan Alexandru Şucan and Lydia E Kavraki. A sampling-based tree planner for systems with complex dynamics. *IEEE Transactions on Robotics*, 28(1):116–131, February 2012. ISSN 1552-3098. doi: 10.1109/tro.2011.2160466. URL http://dx.doi.org/10.1109/tro.2011.2160466.

[7] Neil T. Dantam. Robust and efficient forward, differential, and inverse kinematics using dual quaternions. *International Journal of Robotics Research*, 2020.

[8] Danny Driess, Ozgur Oguz, Jung-Su Ha, and Marc Toussaint. Deep visual heuristics: Learning feasibility of mixed-integer programs for manipulation planning. In *International Conference on Robotics and Automation2020*, pages 9563–9569, 2020. doi: 10.1109/ICRA40945.2020.9197291.

[9] Gamaleldin Elsayed, Dilip Krishnan, Hossein Mobahi, Kevin Regan, and Samy Bengio. Large margin deep networks for classification. *Advances in neural information processing systems*, 31, 2018.

[10] Zhouyu Fu, Antonio Robles-Kelly, and Jun Zhou. Mixing linear svms for nonlinear classification. *IEEE Transactions on Neural Networks*, 21(12):1963–1975, 2010.

[11] David Hsu, Robert Kindel, Jean-Claude Latombe, and Stephen Rock. Randomized kinodynamic motion planning with moving obstacles. *International Journal of Robotics Research*, 21(3):233–255, 2002.

[12] Lucas Janson, Edward Schmerling, Ashley Clark, and Marco Pavone. Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions. *International Journal of Robotics Research*, 34(7):883–921, 2015.

[13] Steven G. Johnson. The NLopt nonlinear-optimization package, 2022. http://github.com/stevengj/nlopt.

[14] Siargey Kachanovich. *Meshing submanifolds using Coxeter triangulations*. PhD thesis, COMUE Université Côte d'Azur (2015-2019), 2019.

[15] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *International Journal of Robotics Research*, 30(7):846–894, 2011.

[16] Lydia E Kavraki, Petr Svestka, J-C Latombe, and Mark H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics*, 12(4):566–580, 1996.

[17] Dieter Kraft. A software package for sequential quadratic programming. Technical Report DFVLR-FB 88-28, Institut für Dynamik der Flugsysteme, Oberpfaffenhofen, July 1988.

[18] Dieter Kraft. Algorithm 733: TOMP–fortran modules for optimal control calculations. *Transactions on Mathematical Software (TOMS)*, 20(3):262–281, 1994.

[19] James J. Kuffner and Steven M. LaValle. RRT-connect: An efficient approach to single-query path planning. In *International Conference on Robotics and Automation2000*, volume 2, pages 995–1001, 2000.

[20] Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006.

[21] Steven M. LaValle and James J. Kuffner. Randomized kinodynamic planning. *International Journal of Robotics Research*, 20(5):378–400, 2001.

[22] Boyang Li, Qiangwei Wang, and Jinglu Hu. A fast svm training method for very large datasets. In *2009 International Joint Conference on Neural Networks*, pages 1784–1789, 2009. doi: 10.1109/IJCNN.2009.5178618.

[23] Sihui Li and Neil T Dantam. Learning proofs of motion planning infeasibility. In *Robotics: Science and Systems*, 2021.

[24] Sihui Li and Neil T. Dantam. Exponential convergence of infeasibility proofs for kinematic motion planning. In *Workshop on the Algorithmic Foundations of Robotics*, 2022.

[25] Tomas Lozano-Perez. Spatial planning: A configuration space approach. In *Autonomous robot vehicles*, pages 259–271. Springer, 1990.

[26] Xue Mao, Ou Wu, Weiming Hu, and Peter O'Donovan. Nonlinear classification via linear svms and multi-task learning. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 1955–1958, 2014.

[27] Teledyne FLIR. Packbot 510. https://www.flir.com/products/packbot/?vertical=ugs&segment=uis, 2022. [Online; accessed 22-May-2022].

[28] Di Wang, Xiaoqin Zhang, Mingyu Fan, and Xiuzi Ye. Hierarchical mixing linear support vector machines for nonlinear classification. *Pattern Recognition*, 59:255–267, 2016.

[29] Andrew Wells, Neil T. Dantam, Anshumali Shrivastava, and Lydia E. Kavraki. Learning feasibility for task and motion planning in tabletop environments. *Robotics & Automation Magazine*, 4(2):1255–1262, 2019.

[30] Zeyi Wen, Jiashuai Shi, Qinbin Li, Bingsheng He, and Jian Chen. ThunderSVM: A fast SVM library on GPUs and CPUs. *Journal of Machine Learning Research*, 19:

797–801, 2018.

[31] Li Yujian, Liu Bo, Yang Xinwu, Fu Yaozong, and Li Houjun. Multiconlitron: A general piecewise linear classifier. *IEEE Transactions on Neural Networks*, 22(2): 276–289, 2010.