# Robot Motion Planning with Trajectory Wide Constraints

Rachel M. Holladay (Student) and Siddhartha S. Srinivasa (Mentor)
Robotics Institute, Carnegie Mellon University

## I. Introduction

Many of our everyday jobs are defined via a variety of task-specific constraints. In order for robots to perform these tasks, the robot's motion planners must respect these constraints. To allow robots to collaboratively work along side human partners, we also want these constraints to be easily specified by non-experts.

While a robotic manipulator moves and plans in its joint or configuration space, many human tasks are naturally defined in task space. For example, when carrying a glass full of water, we typically keep the glass upright to not spill the contents. For more complex tasks, we might require the robot's end effector follow a particular trajectory, tracing out a specific shape.

Therefore, in addition to joint limits and collision constraints, we further constrain our motion planner to follow a *specified task space path*. Our hope is that by defining constraints in task space, our constraints are easy to specify and generalize across scenes. For example, in Fig.1 a user demonstrates a path by tracing the robot's arm, which is visualized to the right.

Several prior approaches involve biasing the robot's planner towards the desired motion [1–3] or by formulating this as a controls problem [4].

We began by examining this problem via trajectory optimization. We explored recreating task space demonstrations by optimizing a joint space path with respect to a cost function defining the distance between the demonstration and the task space motion achieved by the path [5]. While our method produced promising and exciting results, the optimization process was often slow and susceptible, as trajectory optimizers are, to finding a local, not global, minimum.

To overcome this, we formulated our problem as an instance of graph search, interweaving task and configuration space. We propose a method that samples task space for poses and connects them via configuration space paths that maintain our constraint. By creating a graph of feasible sub-paths we can search over the graph to find our full path. This graph search method is still an active area of research for us, although we look forward to the coming results.

## II. Problem Statement

We work with a robot manipulator endowed with a configuration space $q \in \mathcal{C}$. We map configurations to the task space $x \in SE(3)$ via forward kinematics to $x = FK(q)$. We are given a *reference* path in task space $\bar{\xi}_x : [0, 1] \rightarrow SE(3)$.

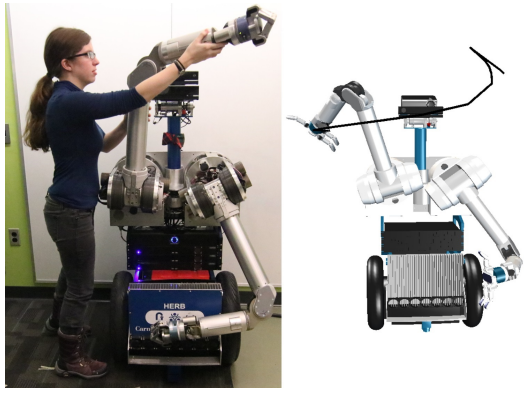


Fig. 1: On the left hand side, a user demonstrates a task space trajectory that is visualized on the right. Our goal is to enable the robot to be able to recreate the shape of the provided demonstration in the general setting.

Our goal is to produce a path that closely matches the reference path subject to constraints on the system:

$$\xi^* = \arg\min_{\xi \in \Xi} ||\xi - \bar{\xi}|| \quad \text{s.t. constraints} \tag{1}$$

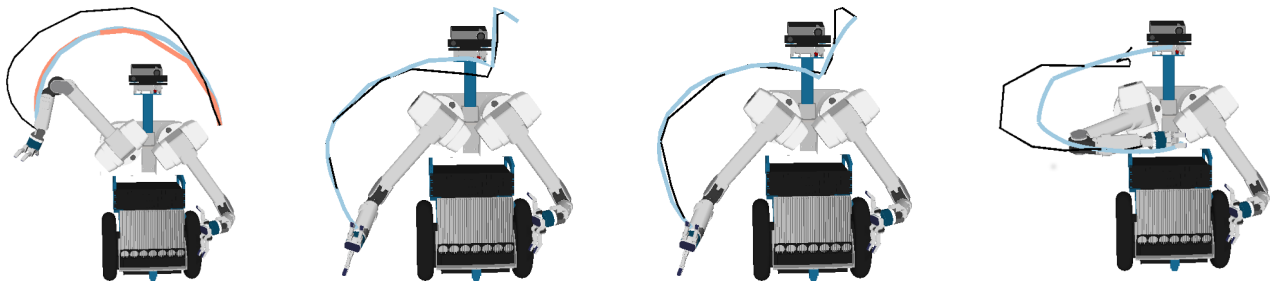
Critical to (1) is our distance metric for estimating the closeness of curves. We formated two path metrics for closeness based on the discrete one way Hausdorff and Frechet distances, described below.

**One-Way Hausdorff Distance.** The Hausdorff distance is a method for measuring how far apart two subsets of metric space are [6]. Although originally formulated as metric for shapes, the one-way Hausdorff distance can also be applied to curves, point sets and objects [7–9]. Hence for paths, if every point on the path was to find its closest neighbor on the reference path, the one way Hausdorff distance would be the longest neighbor to neighbor distance.

**Frechet Distance.** The Frechet distance captures the difference in flow between two curves [10]. The Frechet distance is commonly explained through an analogy, where a dog is walking along $\xi$ at speed parameterization $\alpha$ and its owner is walking along $\bar{\xi}$ at speed parameterization $\beta$ [11]. The two are connected via a leash. The Frechet distance is the shortest possible leash via some distance metric $d$ such that there exists a parameterization $\alpha$ and $\beta$ so that the two stay connected and move monotonically.

## III. Trajectory Optimization

Our goal is optimize the paths we produce to be close to our reference trajectories according to the distance



(a) Basic Optimization (b) Split Method (c) Staple Method (d) Translating Path

Fig. 2: While our initial optimization in (a) does not fully capture the refernce path, we can use splitting (b) and stapling (c) to improve performance. (d) By manipulating our original paths in task space (black), we can create new paths (blue).

metrics described in Sec. II. We use the power of existing trajectory optimization algorithms, specifically TrajOpt [12] which handles a broad range of constraints.

We recorded a set of 24 demonstrations and optimized using the Hausdorff or Frechet metric as our cost function. An example output is shown in Fig.2a for the Hausdorff (orange) and Frechet (blue) metrics. While the path in Fig.2a capture the shape to an extent, it fails to capture the shape entirely. Our optimization process does not drive our cost to zero in part because these demonstrations are difficult for an optimizer to achieve. Many times, the optimizer falls into a local minimum that is different from our demonstration due to self-collisions or joint limits.

To produce more accurate demonstrations we therefore assist TrajOpt via two methods: *split* and *staple*. These methods add more constraints to to our problem, thus moving our basin of attraction to new locations.

Using our splitting method, we split our path in $k$ segments and optimize on each segment. While this produces improved results (Fig.2b), it provides assistance evenly when we would prefer to provide assistance where it is most needed. For stapling, we begin with a path optimized with a distance metric. With that metric, we then find the point of of the path that errors furthest from our reference path. We then staple that point to the reference path, similarly to the splitting method, and recurse on the two pieces: the start to the staple point and the staple point to the end. We repeat this iteratively until the maximum violation is below some threshold. An example of stapling is shown in Fig.2c.

Hence, splitting segments the path in a predefined way, while stapling segments through a more intelligent method. We can now easily generate a path that follows a new task space path formed by warping the original reference path. For example, in Fig.2d, we can translate our original task space path by 10 cm and use our optimization to generate a new path.

Despite the success of this method, we are not entirely satisfied. The optimization process often produces longer than desirable planning times and suffers from local minima. Therefore, we have begun exploring a graph search method described below.

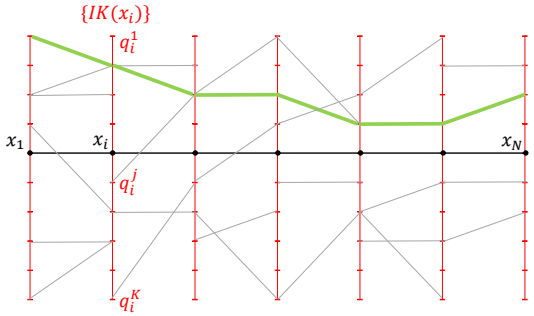



Fig. 3: A visualization of our graph search algorithm in chart space. We sample our reference trajectory in task space and plan between in configuration space, respecting the constraint to recreate our reference trajectory.

## IV. Graph Search

As an alternate approach to following task space paths, we draw upon the success of randomized sample based graph search motion planners [13–15]. Our goal is to sample on the task space path and plan along this constraint.

A visualization of the approach can be seen in Fig.3. The black line represents the reference task space path, with waypoints sampled at the black dots. For each point in task space, there are multiple possible inverse kinematic (IK) solutions in configuration space, visualized as tics in red. Connected points, or edges in the graph, represent a path from one configuration to another. To create an edge we require that the path is feasible and that the path is close to the reference path according to one of our distance metrics from Sec. II. Using graph search, such as Dikjsta or A* [16, 17], we attempt to find a path through the graph, like that shown in green in Fig.3.

## V. Discussion

Over the past year we have examined planning to follow a path in task space. We have explore two methods, trajectory optimization and graph search, and will continue to investigate these methods futher. Our goal is enable robots to easily and quickly plan in task space, thus increasing their skill set and usability as collaborative partners.

## VI. Deliverables

Through the project I maintained a main project webpage [18] as well as a engineering notebook where I tracked my progress [19].

In March we submitted our trajectory optimization work to IEEE/RSJ IROS (International Conference on Intelligent Robots and Systems) and it was recently accepted [5]. While we are continuing to develop our graph search method, we are hoping to submit our progress to IEEE ICRA (International Conference on Robotics and Automation) in September. I have proposed a larger scale version of the graph search as my SCS (School of Computer Science) Honor Undergraduate Senior Thesis.

## ACKNOWLEDGMENT

I greatly appreciate the gracious support CREU has provided by funding me to conduct this research. This has been an incredibly valuable and rewarding experience that has allowed me to grow as a researcher.

I would like to thank the members of the Personal Robotics Lab for their consistently helpful discussion, advice and support. Finally, I would like to thank my advisor Sidd for being my guiding 'sensei'.

## References

[1] D. Berenson, S. S. Srinivasa, D. Ferguson, and J. J. Kuffner, "Manipulation planning on constraint manifolds," in *IEEE ICRA*, pp. 625–632, IEEE, 2009.

[2] M. Stilman, "Global manipulation planning in robot joint space with task constraints," *Robotics, IEEE Transactions on*, vol. 26, no. 3, pp. 576–584, 2010.

[3] Z. Yao and K. Gupta, "Path planning with general end-effector constraints: Using task space to guide configuration space search," in *IEEE/RSJ IROS*, pp. 1875–1880, IEEE, 2005.

[4] S. Seereeram and J. T. Wen, "A global approach to path planning for redundant manipulators," *IEEE TRA*, vol. 11, no. 1, pp. 152–160, 1995.

[5] R. Holladay and S. Srinivasa, "Distance metrics and algorithms for task space path optimization," in *IEEE/RSJ IROS*, 2016.

[6] F. Hausdorff and E. Brieskorn, *Felix Hausdorff-Gesammelte Werke Band III: Mengenlehre (1927, 1935) Deskripte Mengenlehre und Topologie*, vol. 3. Springer Science & Business Media, 2008.

[7] M.-P. Dubuisson and A. K. Jain, "A modified hausdorff distance for object matching," in *ICPR*, vol. 1, pp. 566–568, IEEE, 1994.

[8] D. P. Huttenlocher and K. Kedem, "Computing the minimum hausdorff distance for point sets under translation," in *Annual symposium on Computational geometry*, pp. 340–349, ACM, 1990.

[9] É. Belogay, C. Cabrelli, U. Molter, and R. Shonkwiler, "Calculating the hausdorff distance between curves," *Information Processing Letters*, vol. 64, no. 1, pp. 17–22, 1997.

[10] M. M. Fréchet, "Sur quelques points du calcul fonctionnel," *Rendiconti del Circolo Matematico di Palermo (1884-1940)*, vol. 22, no. 1, pp. 1–72, 1906.

[11] E. W. Chambers, E. C. De Verdiere, J. Erickson, S. Lazard, F. Lazarus, and S. Thite, "Homotopic fréchet distance between curves or, walking your dog in the woods in polynomial time," *Computational Geometry*, vol. 43, no. 3, pp. 295–311, 2010.

[12] J. Schulman, J. Ho, A. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding locally optimal, collision-free trajectories with sequential convex optimization.," in *RSS*, vol. 9, pp. 1–10, Citeseer, 2013.

[13] S. M. LaValle, "Rapidly-exploring random trees a new tool for path planning," 1998.

[14] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[15] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Batch informed trees (bit*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs," in *IEEE ICRA*, pp. 3067–3074, IEEE, 2015.

[16] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[17] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *Systems Science and Cybernetics, IEEE Transactions on*, vol. 4, no. 2, pp. 100–107, 1968.

[18] R. Holladay, "Project homepage." http://www.andrew.cmu.edu/user/rmh/research/taskspace_planning.html, 2015-2016.

[19] R. Holladay, "Project engineering notebook." http://www.andrew.cmu.edu/user/rmh/constraints_blog.html, 2015-2016.