Minimizing Task-Space Fréchet Error via Efficient Incremental Graph Search

Rachel Holladay[1], Oren Salzman[2], and Siddhartha Srinivasa[3]

***Abstract*—We present an anytime algorithm that generates a collision-free configuration-space path that closely follows a desired path in task space, according to the discrete Fréchet distance. By leveraging tools from computational geometry, we approximate the search space using a cross-product graph. We use a variant of Dijkstra's graph-search algorithm to efficiently search for and iteratively improve the solution. We compare multiple proposed densification strategies and empirically show that our algorithm outperforms a set of state-of-the-art planners on a range of manipulation problems. Finally, we offer a proof sketch of the asymptotic optimality of our algorithm.**

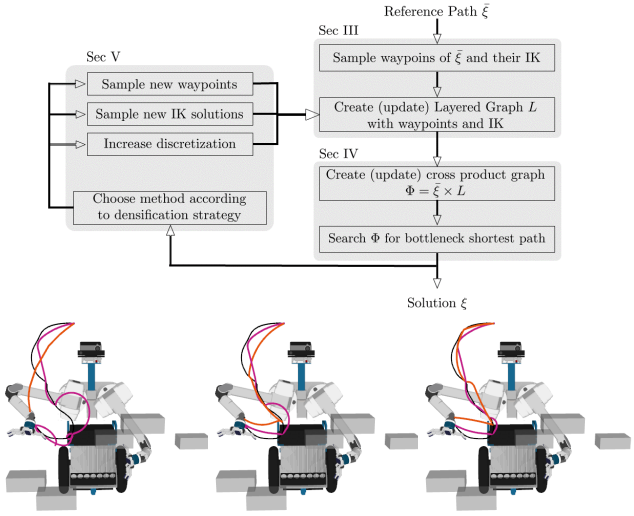
***Index Terms*—Motion and Path Planning, Computational Geometry, Kinematics**

I. Introduction

THE classical formulation of the motion-planning problem calls for planning a collision-free (possibly optimal) path between a given start and target configuration [1] in a robot's configuration space ($\mathcal{C}$-space). However for robot arms, the path of the end-effector is often of greater relevance. For example, the end-effector path might be subject to constraints such as keeping a coffee mug upright, or might even be restricted to a specific path such as pulling a door open, writing on a whiteboard, or welding a seam on a car.

We focus on the latter problem: enabling a redundant robotic manipulator to follow a reference path in task space. There are two state-of-the-art approaches to solving this problem:

1) **Projection-based approaches** exploit the kinematic redundancy of the manipulator [2], [3], [4], [5] to drive the robot along the desired path [6], [7], [8], [9], [10]. Although these are typically efficient and can follow the desired path accurately, they are *myopic* and can fail due to joint limits or collisions [11], [12], [13].
2) **Graph-based approaches** sample the task-space path, compute a set of inverse kinematics (IK) solutions in the $\mathcal{C}$-space for each sample along the path, create a graph by connecting nearby configurations via a simple planner (like a straight line), and solve for the shortest-feasible path on this graph [14], [15]. Although they can solve more intricate problems via organized search, they are typically much slower, when compared to projection-based approaches. More importantly, their optimization criteria (shortest path in $\mathcal{C}$-space) lacks any notion of "following" the task-space reference path.



(a) Initial Paths (b) Midway Progress (c) Final Paths

Fig. 1: On top is a flowchart of our algorithm. We create data structures that allow us to efficiently compute a path that minimizes the Fréchet distance to the reference path and then to incrementally reduce this distance. Each grey box outlines a major step: (1) generating candidate paths, (2) searching over paths and (3) densifying. On the bottom we see the progression of our planner (pink) and the planner from Holladay and Srinivasa [16] (orange) as they trace out the reference path (black) in the presence of obstacles (grey).

The goal of our paper is to make graph-based approaches more efficient while still being sufficiently accurate. Central to our approach is the simple, yet fundamental question: *What does it mean to approximately follow a path?* We can rephrase this more formally as

> What is the right distance metric for comparing two paths in task space?

Informally, let us say that we would like to stay within an ε-ball of any point on the reference path. The *one-way*

Manuscript received: September 10, 2018; Revised December 13, 2018; Accepted January 30, 2019.

This paper was recommended for publication by Editor Allison Okamura upon evaluation of the Associate Editor and Reviewers' comments. This material is based upon work supported by ONR BAA 13-0001 and grants from the CRA-W's CREU and CMU's SRC URO programs.

[1]Rachel Holladay is with the Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, USA. rhollada@mit.edu

[2]Oren Salzman is with the Robotics Institute, Carnegie Mellon University, USA. osalzman@andrew.cmu.edu

[3]Siddhartha Srinivasa is with the Paul G. Allen School of Computer Science & Engineering, University of Washington, USA.siddh@cs.uw.edu

Digital Object Identifier (DOI): see top of this page.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LRA.2019.2899668, IEEE Robotics and Automation Letters

Hausdorff distance satisfies this property [17]. However, we might end up shortcutting large sections of the path. Now, if we force such proximity for both the reference and the target path, via the *two-way Hausdorff distance*, we avoid shortcutting [17]. However, this does not preserve *monotonicity* of traversal. If we additionally enforce monotonicity, we end up with the *Fréchet distance* [18].

Holladay and Srinivasa [16] showed the practical superiority of the Fréchet distance over other metrics for trajectory optimization of manipulator motion. However, their approach, like other optimization-based approaches, suffers from local minima (Sec. II-D).

In this work we suggest to approximate the search space of candidate paths by a layered graph that organizes IK solutions by their task-space location along the reference path. By representing both the layered graph and our reference path as simplicial complexes [19], we can construct the cross product of these two complexes. This, in turn, allows us to efficiently compute the (discrete) Fréchet distance between the set of candidate paths in the layered graph and our reference path via a simple Bottleneck Shortest-Path algorithm.

We present an anytime algorithm for incrementally densifying these structures and improving our solution and prove that our approach is asymptotically optimal, given some assumptions. Empirically, we evaluate our approach on several seven degree-of-freedom manipulators and demonstrate its efficacy when compared to existing state-of-the-art algorithms on multiple paths and parameter settings. For a summary of our algorithmic approach, see Fig. 1 (top).

Our key insight is that marrying the correct metric (Fréchet distance) with the correct search algorithm (bottleneck shortest path) enables us to focus our computation on parts of the space that are most relevant for the problem, thereby gaining better efficiency.

II. Problem Definition and Algorithmic Background

In this section we provide the basic definitions (Sec II-A) which allow us to formally state our problem (Sec. II-B). We define the Fréchet distance (Sec. II-C) and briefly describe the approach proposed by Holladay and Srinivasa [16]. We explain a key shortcoming of their work, which motivates our approach (Sec. II-D).

A. Definitions

A configuration q is a d-dimensional point that completely describes the location of the robot and the $\mathcal{C}$-space $\mathcal{C}$ is the set of all configurations [20]. A task-space point $\tau \in SE(3)$ describes the position and orientation of the robot's end effector and the task space is the set of all such points. Paths in $\mathcal{C}$-space and task space are continuous mappings $\xi : [0,1] \to \mathcal{C}$ and $\xi : [0,1] \to SE(3)$, respectively[1].

[1]For simplicity, we use the same notation for paths in $\mathcal{C}$-space and in task space. The specific space will be clear from the context.

The robot induces a forward kinematics FK : $\mathcal{C} \to SE(3)$ and an inverse kinematics IK : $SE(3) \to 2^{\mathcal{C}}$ that map a configuration to a unique task-space pose and a task-space pose to a set of configurations, respectively. By a slight abuse of notation we will use FK$(\cdot)$ to map a $\mathcal{C}$-space path into a task-space path. Equipped with these definitions, we can define our problem.

B. Problem Statement

We are given a robot and a reference path in task space $\bar{\xi}$ that is a polyline given as a sequence of waypoints. Let $\Xi_{\bar{\xi}} \subset \mathcal{C}$ be the set of all collision-free paths in $\mathcal{C}$-space that have the same start and end task-space poses as $\bar{\xi}$. Our objective is to compute

$$\xi^* = \arg\min_{\xi \in \Xi_{\bar{\xi}}} ||\text{FK}(\xi), \bar{\xi}||. \quad (1)$$

Namely, we seek a collision-free path $\xi \in \mathcal{C}$ whose forward kinematics maps to a path in task space, FK(ξ), that follows $\bar{\xi}$ as close as possible, given some similarity metric $||\cdot,\cdot||$. Similarly to $\bar{\xi}$, our produced path ξ is a polyline represented by a sequence of waypoints. To validate that paths are collision-free, we assume that we are given access to a discriminative black-box collision detector that, given a configuration $q \in \mathcal{C}$, returns whether or not the robot, placed in q, would be in collision. The distance metric $||\cdot,\cdot||$ used to compare paths is the Fréchet distance, described below. For a discussion motivating the use of the Fréchet distance in this context, see Sec. I and Holladay and Srinivasa [16].

C. Distance Metrics

To describe the Fréchet distance, we borrow a common analogy where a dog is walking along a path ξ_0 at speed parameterization α and its owner is walking along another path ξ_1 at speed parameterization β. The two are connected via a leash. The Fréchet distance is the shortest possible leash via some distance metric d_{TS} such that there exists a parameterization α and β so that the two stay connected and move monotonically. More formally the continuous Fréchet distance between ξ_0 and ξ_1 is given by:

$$F(\xi_0, \xi_1) = \inf_{\alpha,\beta} \max_{t\in[0,1]} \left\{ d_{TS}\Big(\xi_0(\alpha(t)),\, \xi_1(\beta(t))\Big) \right\}. \quad (2)$$

As is common in motion planning [21], given two points $x, y \in SE(3) = \mathbb{R}^3 \times SO(3)$, we define their distance, $d_{TS}(x,y)$, as the weighted sum[2] of the Euclidean metric in $\mathbb{R}^3$ and the standard great circle solid angle metric in $SO(3)$ for the respective components.

Since computing the continuous Fréchet distance is notoriously difficult, especially in non-Euclidean spaces [22] and our path representation is given as a series of waypoints, we approximate $F(\cdot,\cdot)$ using the the discrete Fréchet distance $F_d(\cdot,\cdot)$, where the "leash"

[2]In our setting, we prioritize translational distance over rotational distance using a ratio of 0.17, which corresponds to 3mm mapping to 1 degree.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LRA.2019.2899668, IEEE Robotics and Automation Letters

is only considered between discrete waypoints along the two paths. This metric can be efficiently computed via dynamic programming [23], [24].

D. Trajectory-Optimization Approach

The key insight from Holladay and Srinivasa [16] is to optimize Eq. 1 by minimizing $F_d(\bar{\xi}, \text{FK}(\xi))$. Framed as a trajectory-optimization problem, the paper provides methods to heuristically assist the optimizer by constraining the computed path into a sequence of smaller problems.

We examine the algorithm's behavior on HERB, a bimanual mobile manipulator with seven degree-of-freedom arms [25], as it tries to follow a straight-line reference path $\bar{\xi}$, shown as the dotted line in Fig. 2. The algorithm picks start and end configurations and then plans a path from start to end, attempting to minimize $F_d(\bar{\xi}, \text{FK}(\xi))$.

With the starting configuration in Fig. 2a (left), the planner drives the cost to zero, producing the solid red line path shown in Fig. 2a (right). However, since this is a redundant manipulator, the algorithm could have also picked the starting configuration shown in Fig. 2b (left). Given this configuration, there is no path that exactly follows $\bar{\xi}$. Therefore the optimizer produces the red path in Fig. 2b (right), which deviates significantly from $\bar{\xi}$. The optimization-based algorithm of [16] will then split $\bar{\xi}$ at the point where the generated path deviates the most from $\bar{\xi}$, according to the Fréchet distance. In this case, it splits the path in the middle and samples an IK solution, shown in Fig. 2c. As shown in Fig. 2d, the first half of the path still suffers from the original problem.

This limitation stems from the fact that the algorithm samples one IK solution for each pose along $\bar{\xi}$. However, there is a space of multiple IK solutions which may admit different paths. This motivates our method, which searches over a space of IK solutions in an anytime fashion.

III. Generating a Set of Candidate Paths

Recall that our goal, defined in Eq. (1), is to find a collision-free path $\xi \in \mathcal{C}$ such that FK(ξ) minimizes the Fréchet distance to $\bar{\xi}$. This will be done by solving the following problem

$$\xi^* = \arg\min_{\xi \in \Xi_{\bar{\xi}}} F_d(\text{FK}(\xi), \bar{\xi}), \tag{3}$$

and iteratively refining the number of waypoints along ξ and $\bar{\xi}$ to refine our approximation of the continuous Fréchet distance. To do so, we build a layered graph L that approximates the set of candidate paths, $\Xi_{\bar{\xi}}$. As our algorithm progresses, we try to both improve the *quality* of our path by exploring more candidate paths and improve the *accuracy* of our Fréchet approximation by increasing our sampling resolution.

A. Layered Graph

Consider the set of inverse kinematic solutions to all points along our reference path $\bar{\xi}$:

$$\mathcal{M}_{\bar{\xi}} = \bigcup_{\alpha \in [0,1]} \text{IK}(\bar{\xi}(\alpha)). \tag{4}$$

Any collision-free path that connects IK($\bar{\xi}(0)$) with IK($\bar{\xi}(1)$) while completely lying on the manifold $\mathcal{M}_{\bar{\xi}}$ minimizes Eq. (1). To approximate such a path, we sample $\mathcal{M}_{\bar{\xi}}$ and connect samples by straight-line segments in $\mathcal{C}$-space (that may deviate from $\mathcal{M}_{\bar{\xi}}$).

The structure of Eq. (4) suggests two parameters that can be used to organize our sampling in a structured manner. The first is the location of a point in task space along $\bar{\xi}$, denoted by α. The second is the set of IK solutions at each point.[3] This further demonstrates the key limitation of the approach by Holladay and Srinivasa [16] which consider for each location α only one configuration.

Following the discussion above, we construct a layered graph $L = (V_L, E_L)$ embedded in $\mathcal{C}$-space where each layer is a set of IK solutions for one task-space point lying on the reference path (Fig. 3).

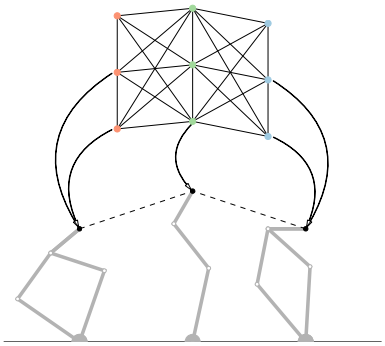


Fig. 3: Each layer in our layered graph (top) maps to a task-space pose (bottom) along our reference path, shown as the dotted line. For each pose, there are multiple IK solutions, which make up the elements of each layer.

To construct our graph, we begin by sampling n waypoints in task space along our reference path $\{w_1 \dots w_n\}$. At each waypoint w_j, we initially compute up to k IK solutions $\{q_j^1 \dots q_j^k\}$ by querying random solutions from an analytical IK solver (IKFast [26]). Each configuration q_j^i is a vertex in our graph L. Namely,

$$V_L = \{q_j^i | 1 \le j \le n \text{ and } 1 \le i \le k\}. \tag{5}$$

We next define our edge set, E_L. Each vertex in a layer of IK solutions connects to every vertex in the subsequent layer and to every vertex in its own layer. Intuitively the path passes through every waypoint, with the freedom to select any IK solution for that waypoint. More formally,

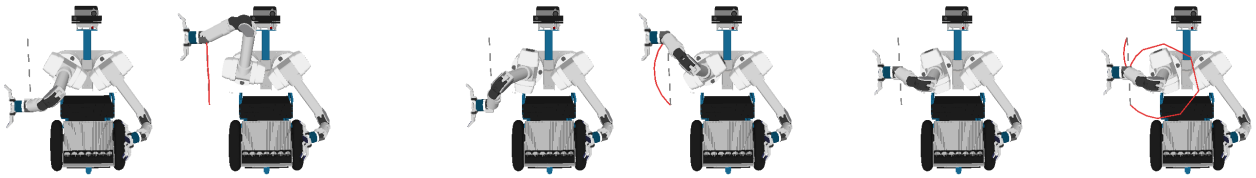
$$\begin{aligned} E_L = &\{(q_j^{i_1}, q_{j+1}^{i_2}) | 1 \le j < n-1, 1 \le i_1, i_2 \le k\} \\ &\cup \{(q_j^{i_1}, q_j^{i_2}) | 1 \le j \le n, 1 \le i_1, i_2 \le k\}. \end{aligned} \tag{6}$$

As mentioned, each edge is a straight-line segment in $\mathcal{C}$-space between the two configurations, which does not necessarily lie on $\mathcal{M}_{\bar{\xi}}$.[4] To account for deviation from $\mathcal{M}_{\bar{\xi}}$ in our discrete Fréchet calculation, we subsample each edge. As we increase the subsampling

[3] Assuming that we have a redundant manipulator, there is an infinite set of IK solutions for each task-space point.

[4] In our implementation, we delay collision checking of edges along these paths.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LRA.2019.2899668, IEEE Robotics and Automation Letters



(a) Successfully optimized path. (b) Unsuccessfully optimized path. (c) Splitting point. (d) New path.

Fig. 2: Visualization of the approach taken by Holladay and Srinivasa [16]. Reference path and computed path are shown in dotted black and solid red lines, respectively.

resolution during our densification process, our discrete Fréchet distance will better approximate the continuous Fréchet distance [27].

B. Naïve Search Method

Given our layered graph L, let Ξ_L denote the set of all paths in L that connect any vertex in the first layer of L to any vertex in the last layer of L. We can restate Eq. (3) as

$$\xi_L^* = \arg\min_{\xi \in \Xi_L} F_d(\text{FK}(\xi), \bar{\xi}). \tag{7}$$

Since Fréchet is a metric over entire paths, not path segments (i.e., individual edges), we cannot simply search L in a Dijkstra-like manner. One naïve option would be to enumerate all candidate paths in Ξ_L and compute $F_d(\bar{\xi}, \text{FK}(\xi_L))$ for all $\xi_L \in \Xi_L$. However, $|\Xi_L| = \Omega(n^k)$. Instead, we adapt a method that computes the cross product between our layered graph and the reference path, allowing us to efficiently search for the minimal-cost path in L in $O(n^2k^2\log(nk))$ time. If we used a metric that operated on individual edges, we would be able to search L directly in a Dijkstra-like manner. We discuss this further in Sec. VIII

IV. Computing the Minimal-Cost Path

To efficiently compute a solution to Eq. (7) we represent $\bar{\xi}$ as a (one-dimensional) graph $G_{\bar{\xi}} = (V_{\bar{\xi}}, E_{\bar{\xi}})$ where $V_{\bar{\xi}}$ are the sampled waypoints of $\bar{\xi}$ and an edge $e \in E_{\bar{\xi}}$ connects two subsequent waypoints. This allows us to view both L and $G_{\bar{\xi}}$ as abstract one-dimensional simplicial complexes[5]. Har-Peled and Raichel introduced an algorithm for computing the Fréchet distance between two such complexes by considering their cross product [19]. Therefore, our instance is a restricted case of their problem and we present our adaptation. Following Fig. 1, we first create a new graph $\Phi = L \times G_{\bar{\xi}}$ and then use it to solve Eq. (7).

A. Cross-Product Graph Φ

In this section, we define for Φ the set of vertices V_Φ, edges E_Φ and their costs. Set $V_\Phi = V_{\bar{\xi}} \times V_L$. Namely, each vertex in V_Φ is a pair (w, q) such that $w \in V_{\bar{\xi}}$ and $q \in V_L$. An edge connects two vertices in V_L if either or both elements of each vertex are adjacent to each other in their respective graph. Namely,

$$\begin{aligned} E_\Phi = \{((w_{m_1}, q_{j_1}^{i_1}), (w_{m_2}, q_{j_2}^{i_2}))| \text{ if} \\ ((w_{m_1} = w_{m_2}) \text{ and } (q_{j_1}^{i_1}, q_{j_2}^{i_2}) \in E_L) \text{ or} \\ ((w_{m_1}, w_{m_2}) \in E_{\bar{\xi}} \text{ and } (q_{j_1}^{i_1} = q_{j_2}^{i_2})) \text{ or} \\ ((w_{m_1}, w_{m_2}) \in E_{\bar{\xi}} \text{ and } (q_{j_1}^{i_1}, q_{j_2}^{i_2}) \in E_L)\}. \end{aligned} \tag{8}$$

Set $\text{cost}(w, q) = d_{TS}(w, \text{FK}(q))$ to be the cost[6] of a vertex $(w, q) \in V_\Phi$. The cost of an edge $e = (u, v)$ is the maximum of the cost of its endpoints. Namely, $\text{cost}(u, v) = \max(\text{cost}(u), \text{cost}(v))$.

The cost of a path in Φ is defined as the *maximal* edge cost along this path, also known as the "bottleneck cost". Har-Peled and Raichel show that the cost of such a path is equal to the Fréchet distance between the corresponding curves in the two simplicial complexes that compose the product space [19]. In other words, the cost of a path in the cross-product graph $(w_1, q_1) \dots (w_n, q_n)$ corresponds to the discrete Fréchet distance between the discretized reference path $\{w_1 \dots w_n\}$ and the FK of the polyline $\{q_1 \dots q_n\}$ in the layered graph. Thus, our goal can be restated as finding the bottleneck shortest path in Φ.

B. Computing the Bottleneck Shortest path

Computing the bottleneck shortest path in a graph $G = (V, E)$ is a well-studied problem and there are efficient algorithms that run in time linear in $|E|$ [19]. However, we chose to use a simple variant of Dijkstra's algorithm [29] (with complexity $\mathcal{O}(|E_\Phi| + |V_\Phi| \log |V_\Phi|)$) because we have found it to be empirically faster and it allows for more efficient updates to Φ, as described in Sec. V.

Standard implementations of Dijkstra's algorithm assume that the cost of a path to vertex v coming from vertex u is the cost to reach u plus the cost of the edge (u, v). To compute the bottleneck cost, we simply swap the sum of costs for a $\max()$ such that the $\text{cost}(v)$ = $\max(\text{cost}(v), \text{cost}(u, v))$.

Given the bottleneck shortest path in Φ, we can extract the corresponding path in the layered graph to generate ξ_L that optimizes Eq. (7). While searching, we lazily evaluate the edges in ξ_L for collision [30], [31].

[5]An abstract simplicial complex is a combinatorial description of a simplicial complex—a set composed of points, line segments, triangles, and their n-dimensional counterparts [28].

[6]Har-Peled and Raichel [19] use the term "elevation" to refer to our notion of cost.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LRA.2019.2899668, IEEE Robotics and Automation Letters

V. DENSIFICATION

Following the construction of the cross-product graph Φ, we want to iteratively improve (i) the quality of our solution and (ii) the accuracy of our approximation of the continuous Fréchet distance. This process creates an anytime algorithm.

To improve the quality of our solution, we *densify* our layered graph L to provide more candidate paths to search over. The two parameters of our layered graph (number of layers and number of IK solutions in each layer) suggest two approaches: we can either add another layer to L by choosing a new waypoint along $\bar{\xi}$ and sampling k IK solutions of this waypoint or we can increase the size of an existing layer in L by sampling more inverse kinematic solutions at an existing waypoint. To improve our approximation of the continuous Fréchet distance, we increase the subsampling resolution along edges of our two structures, $G_{\bar{\xi}}$ and L. Given updates to $G_{\bar{\xi}}$ or L, we then update our cross-product graph Φ accordingly.

To summarize, given these two objectives we have defined three densification methods: (i) adding a layer to L, (ii) adding IK samples to an existing layer of L and (iii) increasing subsampling resolution of an edge of L or $G_{\bar{\xi}}$. Given these three densification methods, we present several strategies on how to apply them followed by experimental comparisons.

A. Densification Strategies

Our strategies on where to apply our densification methods are inspired by the PRM literature, which balance global and local updates [32], [33], [34].

Global updates sample either L or $G_{\bar{\xi}}$ uniformly to determine where to add a layer, which layer to augment or which edge to increase the sampling resolution of. Local updates are applied in the neighborhood of the *bottleneck node* along the current best path in Φ, where the Fréchet distance is the largest. From the Fréchet distance analogy, this is where we have the longest leash between a point in the layered graph L and a point in the reference graph $G_{\bar{\xi}}$[7].

Within a single step of densification, we first determine whether to use local or global updates and then pick a densification method uniformly. Having densified either $G_{\bar{\xi}}$ or L and updated Φ accordingly, we search Φ for best current solution. Our Dijkstra-like search of Φ is thus an instance of a dynamic shortest-path problem which allows us to use efficient algorithms that reuse information from previous search episodes [35], [36], [37]. This loop is illustrated in Fig. 1.

We present two strategies for determining whether to use global or local updates and proceed to empirically evaluate their performance.

[7]This is similar to the stapling method described in [16] in that both leverage the Fréchet distance to heuristically focus effort to improve the quality of the current solution.

Hybrid Strategy trades off between local and global updates by choosing local updates with probability p. The values of $p = 0$ and $p = 1$ correspond to purely local updates and purely global updates, respectively.

Local-then-Global Strategy combines local and global methods by reasoning about the progress made across multiple densification steps. We use local updates as long as they continue to improve the current best solution. Once m successive iterations of local updates do not decrease the bottleneck cost, we switch to performing global updates. If global updates reduce our cost, this strategy returns to applying local updates.

B. Experimental Comparison of Densification Strategies

To compare densification strategies, we use the bimanual manipulator HERB to generate 100 instances of layered graphs for a given reference path $\bar{\xi}$, all with the same initial number of waypoints, IK solutions per waypoint, and level of subsampling resolution. For each problem we randomly place rectangular boxes in the vicinity of the robot. We then conduct many iterations of densification. We repeat this process for multiple parameter settings and reference paths. While a summary is given below, there are more detailed experimental results available in the extended version of this paper [38].

Our two strategies, hybrid and local-then-global, each have one parameter. We compare discrete choices of the parameters to select the best one. For the hybrid strategy we compare p-values in the set $\{0, 0.25, 0.5, 0.75, 1\}$ and observe that lower p-values (namely, biasing local updates), produce paths with a shorter Fréchet distance at each iteration. For the local-then-global strategy (referred to as, L-t-G) we compare m-values in the set $\{2, 3, 4, 5, 6\}$ and observe that mid-range m-values produce the best-quality results. Therefore, in comparing our two densification strategies, we used $p = 0.25$ for the hybrid strategy and $m = 5$ for the local-then-global strategy (Fig. 5a).

These results indicate that, while the Fréchet distance is a metric over entire paths and global updates are required, it is benficial to heuristically guide the densification process in the neighborhood of the local bottleneck. An improved densification process leads our anytime algorithm to produce better results faster.

For both strategies, most of the computation time is spent collision checking the path segments, with some smaller fraction spent computing the cost of nodes in the cross-product graph. Before empirically comparing our method to alternative algorithms, we first provide a proof sketch of its asymptotic optimality.

VI. ASYMPTOTIC OPTIMALITY

In this section we state our main theoretic result and provide a proof outline. We show that, under some technical assumptions, our algorithm is asymptotically

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LRA.2019.2899668, IEEE Robotics and Automation Letters

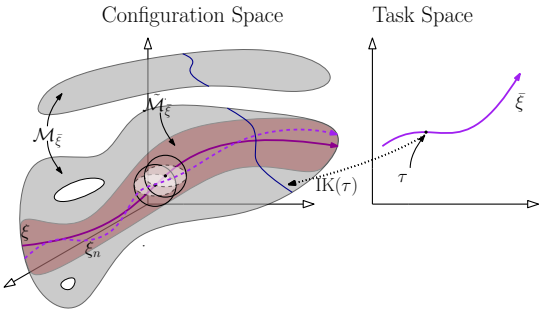


Fig. 4: Visualization of the notation used in proof sketch.

optimal. To do so, we assume that $\mathcal{M}_{\bar{\xi}}$, the set of all configurations that directly map to $\bar{\xi}$, contains a "well-behaved" portion. This notion, together with the proof are detailed in the extended version of this paper [38].

Theorem 1: If $\mathcal{M}_{\bar{\xi}}$ contains a well-behaved portion $\tilde{\mathcal{M}}_{\bar{\xi}}$ then our algorithm is asymptotically optimal. Namely, as $n \to \infty$ and $k \to \infty$ it will asymptotically find a collision-free $\mathcal{C}$-space path whose Fréchet distance from $\bar{\xi}$ tends to zero.

A. Proof sketch

Our proof relies on certain properties (detailed in the extended version of this paper [38]) which hold for any redundant manipulator. Roughly speaking, we require that there is some correlation between distances in $\mathcal{C}$-space and distances in task space. This is required to ensure both that (i) connecting close-by samples on $\mathcal{M}_{\bar{\xi}}$ will lead to minimizing the Fréchet distance and that (ii) sampling close-by points in task space can yield close-by configurations, given enough IK solutions.

Our proof sketch is similar in nature to [39, Thm. 34]. We assume that there exists some path ξ lying on $\tilde{\mathcal{M}}_{\bar{\xi}}$ that directly maps to $\bar{\xi}$. Namely, we have that $F_d\left(\text{FK}(\xi), \bar{\xi}\right) = 0$. We will show that there exists a sequence of a family of paths $\{\Xi_n\}_{n\in\mathbb{N}}$ such that any sequence of paths $\{\xi_n \in \Xi_n\}_{n\in\mathbb{N}}$ converge to ξ (recall that n is the number of sampled waypoints along the reference path $\bar{\xi}$). For each path $\xi_n \in \Xi_n$ we show that there exists some ε_n such that

$$F_d(\text{FK}(\xi_n), \bar{\xi}) \leq \varepsilon_n \text{ and that } \lim_{n\to\infty} \varepsilon_n = 0.$$

Furthermore, if P_n is the probability that our algorithm will produce a collision-free path in Ξ_n then we will show that

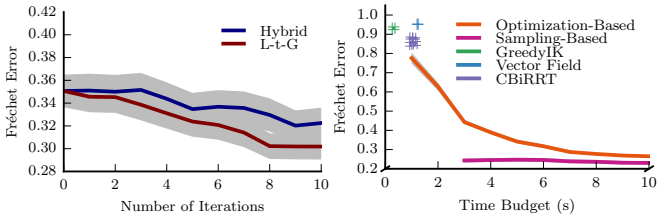
$$\lim_{n\to\infty} P_n = 1.$$

Combining the above will yield that our algorithm is asymptotically optimal. For a figure depicting the notation used throughout our proof, see Fig. 4.

B. Discussion

One assumption that we take implies that there is a path in $\mathcal{C}$-space that directly maps to the reference path. This is due to our sampling scheme which requires that we only sample *on* the reference path and not around it. If this assumption does not hold, an algorithm that minimizes the Fréchet distance cannot restrict itself to sampling only on the reference path.

An interesting difference between our proof and existing asymptotic-optimality proofs such as [39, Thm. 34] is that our algorithm connects any two vertices in subsequent layers. Thus, we did not have to argue about connection radius but about the rate at which we sample waypoints and IK solutions. It would be interesting to alter the algorithm to only connect vertices in subsequent layers if they are within some predefined distance. This would require adding this constraint to the proof of Thm .1.



(a) Densification Strategies (b) Random path

Fig. 5: Empirical evaluation. (a) A comparison of our densification strategies. (b) A comparison of our algorithm with state-of-the-art planners on one of the many random path we evaluated. The results from each algorithm are averaged from 100 iterations. While each figure only shows the results for one reference path and initial layered graph sizes, repeated experiments showed these results were consistent across multiple reference paths and graph sizes. The standard error is shown in grey.

VII. Experimental Results

We compare our sampling-based algorithm with four other planners: an optimization-based approach [16], a vector-field planner [40], a greedy IK planner [40] and CBiRRT (Constrained Bidirectional RRT) [41].

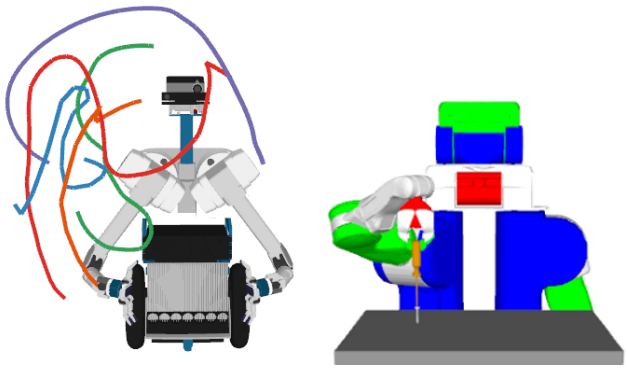
The optimization-based algorithm from [16], summarized in Sec. II-D, continues to split the path into subproblems until the Fréchet distance between the entire path and the reference path is below some threshold value[8]. We adapt this to an anytime algorithm where an entire path is produced and evaluated after each split. The vector-field planner integrates a Jacobian pseudo-inverse to follow a vector field defined by our path [40]. The greedy inverse kinematic planner (GreedyIK) samples IK solutions from $\bar{\xi}$ and attempts to interpolate between them in $\mathcal{C}$-space [40]. CBiRRT plans on constraint manifolds by projecting random samples to our manifold $M_{\bar{\xi}}$ [41]. The algorithm is set to only accept projected samples if they fall within some threshold distance κ of any point on the reference path.

We use the same experimental setup as described in Sec. V-B, averaging the results of each planner over 100 instances. The sampling-based and optimization-based planners both have anytime performance, so we query each planner after t seconds for their best solution so far. Vector Field, GreedyIK and CBiRRT[9] are treated as

[8]In [16] this is referred to as "stapling in task space".

[9]For CBiRRT, we plot the performance across several thresholds (we use κ-values in the set $\{0.2, 0.3, 0.4, 0.5\}$).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LRA.2019.2899668, IEEE Robotics and Automation Letters



(a) Reference Paths (b) Screwdriving Task

Fig. 6: On the left, each color shows one of the many reference paths we evaluated the algorithms on. For visual clarity, the obstacles are not shown. On the right is the setup for our screwdriving task.

single-query planners and therefore do not have anytime performance.

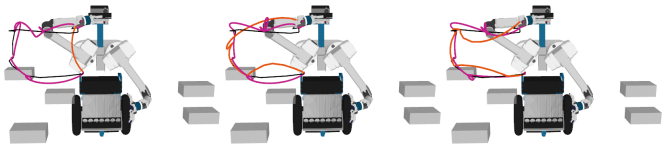
We then test on a variety of paths in the presence of randomly generated rectangular boxes that serves as obstacles. Some of the paths are shown in various colors in Fig. 6a. As a representative example, we compare performance in Fig. 5b and the progression of the anytime algorithms in Fig. 7 for one particular path. Further performance comparisons are available in the extended version of this paper [38].

For this path, the single-query planners quickly produce solutions of low quality. CBiRRT produces equally low-quality solution because the projection and threshold constraints do not enforce the monotonicity of traversal that Fréchet distance does for our sampling-based algorithm.

Turning to our anytime algorithms, the optimization-based approach finds an initial solution faster, but its solution is significantly worse than the one found by the sampling-based approach. While the optimization-based approach improves its solution at a faster rate, the sampling-based approach produces a higher-quality path for a fixed time budget. The difference in quality can be small, but our extended results in [38] show that it can be non-negligible for certain paths.

Our results show that our sample-based approach is able to converge to a path that more closely follows the reference path because it searches over sets of IK solutions and leverages the Fréchet distance to efficiently search. It is important to note that, as expected, the quality of the solution obtained by our planner increases over time. It is hard to observe this trend in Fig. 5b due to the scale needed to compare with the other planners but this can be observed in Fig. 5a as well as in the extended version of this paper [38].

Having verified our path-following approach in a cluttered scene, we also explored how our algorithm performed on a screwdriving task on a different bi-manual manipulator, the PR2 [42]. As seen in Fig. 6b, the goal is for the screwdriver to be rotated one full turn while keeping the screwdriver upright and in contact with the screw. By fixing a robot's grasp of the screwdriver, we



(a) Initial Paths (b) Midway Progress (c) Final Paths

Fig. 7: We show the progression of the optimization-based approach (orange) and the sample-based approach (pink) as they try to follow the reference path (black). Randomly-generated obstacles in the environment are shown in grey. These figures only capture the differences in position, not orientation.

can determine the necessary task-space path ξ of the robot's end effector. Only our sample-based approach was able to generate the turning path, producing initial solutions after 2-3 seconds[10].

VIII. Future Directions and Discussion

We presented an anytime algorithm that produces a collision-free configuration space path that "follows", according to the Fréchet distance, a reference path in task space. By leveraging the Fréchet distance, we were able to organize our space of candidate solutions into a structure that we could efficiently search and incrementally densify. We outlined several strategies for densifying the structure and provided a proof sketch of asymptotic optimality. We concluded with a comparison of our algorithm against several state-of-the-art planners across multiple paths, parameter settings and robotic platforms. Looking forward, we present several future research directions.

In this work we considered how to optimize paths to follow a reference path. We did not consider the length of the path in $\mathcal{C}$-space. In the future, we wish to focus on the bicriteria optimization problem of simultaneously decreasing both the distance (in task space) from the reference path and the path length (in $\mathcal{C}$-space).

In addition to task space positions, we may also want to specify end-effector velocities [2] or forces (especially for the screwdriving task). Another variant, originally suggested by [14] and explored with the Procrustes distance metric in [16], is to consider paths without a fixed starting point, thus allowing the shape to be translated and rotated in space freely.

Our algorithm only samples IK solutions on the reference path. Given many obstacles, we may want to encourage our path to deviate slightly by sampling solutions from an ε-ball around our reference path. This additional flexibility would require revisiting our theoretic guarantees on asymptotic optimality.

As discussed in Sec. III-B, the cumulative nature of the Fréchet motivated our use of the cross product graph. Given a metric that operated on individual edges, we could directly search on our layered graph. For example, if we wanted to plan a path that minimized the one-way Hausdorff distance, a metric mentioned in Sec. I, we

[10]Due to technical difficulties, in the screwdriving task, the optimization-based method was not tested.

would compute the bottleneck shortest-path algorithm on the layered graph, where the cost at every vertex is the minimal distance to any vertex in our reference path. While the decreased computational cost might be tempting, the one-way Hausdorff distance (and even the two-way Hausdorff distance) completely fail to capture distance between paths in the screwdriving task we explored in Sec. VII.

Finally, our work draws some parallels to Hauser's recent work on global redundancy resolution [43]. Both algorithms create PRM-like structures in configuration space, but our work has focused on pathwise redundancy resolution and we leverage a different search method and optimization criteria. We believe that our work, which is complementary to his, may benefit by using his method to pick good IK solutions and leave this as an area of future work.

ACKNOWLEDGMENT

We thank the members of the Personal Robotics Lab, the MCube Lab and Ananya Kumar for helpful discussion and advice.

REFERENCES

[1] D. Halperin, O. Salzman, and M. Sharir, "Algorithmic motion planning," in *Handbook of Discrete and Computational Geometry, Third Edition*, pp. 1311–1342, CRC Press, 2017.

[2] A. Maciejewski and C. Klein, "Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments," *IJRR*, vol. 4, no. 3, pp. 109–117, 1985.

[3] Y. Nakamura, H. Hanafusa, and T. Yoshikawa, "Task-priority based redundancy control of robot manipulators," *IJRR*, vol. 6, no. 2, pp. 3–15, 1987.

[4] S. Ahmad and S. Luo, "Coordinated motion control of multiple robotic devices for welding and redundancy coordination through constrained optimization in Cartesian space," *TRA*, vol. 5, no. 4, pp. 409–417, 1989.

[5] J. W. Burdick, "On the inverse kinematics of redundant manipulators: Characterization of the self-motion manifolds," in *ICRA*, pp. 264–270, IEEE, 1989.

[6] P. Chiacchio, S. Chiaverini, L. Sciavicco, and B. Siciliano, "Closed-loop inverse kinematics schemes for constrained redundant manipulators with task space augmentation and task priority strategy," *IJRR*, vol. 10, no. 4, pp. 410–425, 1991.

[7] Z. Guo and T. Hsia, "Joint trajectory generation for redundant robots in an environment with obstacles," *Journal of Field Robotics*, vol. 10, no. 2, pp. 199–215, 1993.

[8] R. Roberts and A. Maciejewski, "Repeatable generalized inverse control strategies for kinematically redundant manipulators," *Transactions on Automatic Control*, vol. 38, no. 5, pp. 689–699, 1993.

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

[10] B. Siciliano, "Kinematic control of redundant robot manipulators: A tutorial," *JIRS*, vol. 3, no. 3, pp. 201–212, 1990.

[11] D. P. Martin, J. Baillieul, and J. M. Hollerbach, "Resolution of kinematic redundancy using optimization techniques," *TRA*, vol. 5, no. 4, pp. 529–533, 1989.

[12] G. Oriolo, M. Cefalo, and M. Vendittelli, "Repeatable Motion Planning for Redundant Robots Over Cyclic Tasks," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1170–1183, 2017.

[13] D. Rakita, B. Mutlu, and M. Gleicher, "RelaxedIK: Real-time Synthesis of Accurate and Feasible Robot Arm Motion," in *RSS*, 2018.

[14] G. Oriolo, M. Ottavi, and M. Vendittelli, "Probabilistic motion planning for redundant robots along given end-effector paths," in *IROS*, vol. 2, pp. 1657–1662, IEEE, 2002.

[15] R. Industrial, "Descartes," 2015.

[16] R. Holladay and S. Srinivasa, "Distance metrics and algorithms for task space path optimization," in *IROS*, pp. 5533–5540, IEEE, 2016.

[17] Wikipedia contributors, "Hausdorff distance — Wikipedia, The Free Encyclopedia," 2010. [Online].

[18] 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.

[19] S. Har-Peled and B. Raichel, "The Fréchet distance revisited and extended," *TALG*, vol. 10, no. 1, p. 3, 2014.

[20] T. Lozano-Perez, "Spatial planning: A configuration space approach," in *Autonomous Robot Vehicles*, pp. 259–271, Springer, 1990.

[21] I. Şucan, M. Moll, and L. Kavraki, "The open motion planning library," *Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, 2012.

[22] K. Solovey and D. Halperin, "Sampling-Based Bottleneck Pathfinding with Applications to Fréchet Matching," in *ESA*, pp. 76:1–76:16, 2016.

[23] P. Agarwal, R. B. Avraham, H. Kaplan, and M. Sharir, "Computing the discrete Fréchet distance in subquadratic time," *Journal on Computing*, vol. 43, no. 2, pp. 429–449, 2014.

[24] T. Eiter and H. Mannila, "Computing discrete Fréchet distance," tech. rep., Citeseer, 1994.

[25] S. Srinivasa, D. Ferguson, C. Helfrich, D. Berenson, A. Collet, R. Diankov, G. Gallagher, G. Hollinger, J. Kuffner, and M. V. Weghe, "HERB: a home exploring robotic butler," *Autonomous Robots*, vol. 28, no. 1, p. 5, 2010.

[26] R. Diankov and J. Kuffner, "Openrave: A planning architecture for autonomous robotics," *Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-08-34*, vol. 79, 2008.

[27] T. R. Wylie *et al.*, *The discrete Fréchet distance with applications*. PhD thesis, Montana State University-Bozeman, College of Engineering, 2013.

[28] J. R. Munkres, *Elements of algebraic topology*, vol. 4586. Addison-Wesley Longman, 1984.

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

[30] C. Dellin and S. Srinivasa, "A Unifying Formalism for Shortest Path Problems with Expensive Edge Evaluations via Lazy Best-First Search over Paths with Edge Selectors," in *ICAPS*, pp. 459–467, 2016.

[31] N. Haghtalab, S. Mackenzie, A. D. Procaccia, O. Salzman, and S. S. Srinivasa, "The Provable Virtue of Laziness in Motion Planning," in *ICAPS*, pp. 106–113, 2018.

[32] L. Kavraki and J.-C. Latombe, "Randomized preprocessing of configuration for fast path planning," in *ICRA*, pp. 2138–2145, IEEE, 1994.

[33] L. E. Kavraki, P. Švestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *TRA*, vol. 12, no. 4, pp. 566–580, 1996.

[34] R. Bohlin and L. Kavraki, "Path planning using lazy PRM," in *ICRA*, vol. 1, pp. 521–528, IEEE, 2000.

[35] D. Frigioni, A. Marchetti-Spaccamela, and U. Nanni, "Fully dynamic algorithms for maintaining shortest paths trees," *Journal of Algorithms*, vol. 34, no. 2, pp. 251–281, 2000.

[36] G. Ramalingam and T. Reps, "On the computational complexity of dynamic graph problems," *Theoretical Computer Science*, vol. 158, no. 1, pp. 233–277, 1996.

[37] S. Koenig, M. Likhachev, and D. Furcy, "Lifelong planning A*," *Artificial Intelligence*, vol. 155, no. 1-2, pp. 93–146, 2004.

[38] R. Holladay, O. Salzman, and S. Srinivasa, "Minimizing task space Frechet error via efficient incremental graph search," *arXiv preprint arXiv:1710.06738*, 2018.

[39] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *IJRR*, vol. 30, no. 7, pp. 846–894, 2011.

[40] S. S. Srinivasa, A. M. Johnson, G. Lee, M. C. Koval, S. Choudhury, J. E. King, C. M. Dellin, *et al.*, "A system for multi-step mobile manipulation: Architecture, algorithms, and experiments," in *ISER*, pp. 254–265, Springer, 2016.

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

[42] W. G. Robotics, "PR2," *http://www.willowgarage.com/pages/pr2/overview*, 2010.

[43] K. Hauser and S. Emmons, "Global Redundancy Resolution via Continuous Pseudoinversion of the Forward Kinematic Map," *IEEE TASE*, vol. 15, no. 3, pp. 932–944, 2018.