Accepted for publication in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), October 2024

Expansion-GRR: Efficient Generation of Smooth Global Redundancy Resolution Roadmaps

Zhuoyun Zhong, Zhi Li, and Constantinos Chamzas

***Abstract*— Global redundancy resolution (GRR) roadmap is a novel concept in robotics that facilitates the mapping from task space paths to configuration space paths in a legible, predictable, and repeatable way. Such roadmaps could find widespread utility in applications such as safe teleoperation, consistent path planning, and motion primitives generation. However, previous methods to compute GRR roadmaps often necessitate a lengthy computation time and produce non-smooth paths, limiting their practical efficacy. To address this challenge, we introduce a novel method EXPANSION-GRR that leverages efficient configuration space projections and enables a rapid generation of smooth roadmaps that satisfy the task constraints. Additionally, we propose a simple multi-seed strategy that further enhances the final quality. We conducted experiments in simulation with a 5-link planar manipulator and a Kinova arm. We were able to generate the GRR roadmaps up to 2 orders of magnitude faster while achieving higher smoothness. We also demonstrate the utility of the GRR roadmaps in teleoperation tasks where our method outperformed prior methods and reactive IK solvers in terms of success rate and solution quality.**

I. Introduction

A robot is considered kinematically redundant for a specific task if it has more degrees of freedom (DOF) than those strictly required by the task [1]. As demonstrated in Fig. 1, the 7-DOF Kinova robotic arm is asked to reach a target object with a specific end-effector pose. Given its redundancy, the robot can accomplish this task through multiple or potentially infinite configurations. In general, the challenge of redundancy resolution is determining which among the many configurations to select as the most appropriate for the task at hand.

In this work, we focus on the problem of global redundancy resolution (GRR) for end-effector paths [2]. This resolution enables a useful consistency property such that after any cyclic paths, it will always lead the robot back to the same configuration. For example, as shown in Fig. 2, upon executing a closed ellipse path, the global method will always lead the robot back to the original configuration while the non-global method may take the robot to another one. This feature ensures the robot's motion is always repeatable and more predictable during execution.

GRR roadmaps can potentially serve various purposes in robotics, including streamlining repeatable tasks and motion primitives generation. Moreover, a primary application lies in teleoperation interfaces for humans. A common strategy of teleoperation is to allow operators to issue motion commands directly to the end effector space, while the teleoperation

All authors are affiliated with the Department of Robotics Engineering, Worcester Polytechnic Institute (WPI), Worcester, MA 01609, USA {zzhong3, zli11, cchamzas} @ wpi.edu.

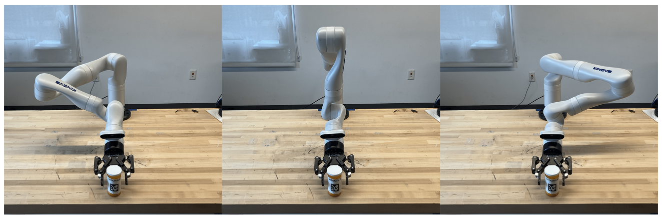
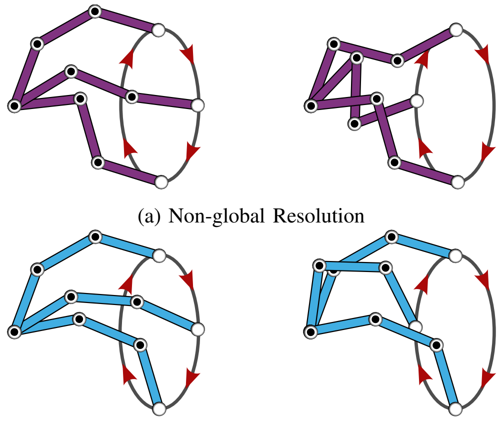


Fig. 1: The Kinova arm is tasked with reaching a 6-DOF pose to pick up an object. For this task, the 7-DOF arm is kinematically redundant and can reach the same object position with multiple configurations.



(a) Non-global Resolution

(b) Global Resolution

Fig. 2: (a) After a cyclic path, non-global resolution can lead to different ending configurations from the same starting point. (b) Global resolution results in consistent paths, where the robot always returns to the original configuration.

system translates these commands into actual joint motion. A GRR roadmap can aid such a system more effectively than reactive Inverse Kinematics (IK) solvers, as it can effectively recover from singularities and entrapment of local minima, while exhibiting repeatable and predictable motions.

However, the computation of such a roadmap poses significant computational challenges. Previously proposed methods often required several hours to days to complete the process [2] and often yield GRR roadmaps with lower smoothness. To tackle this issue, we propose a novel method EXPANSION-GRR that uses an expansion strategy to rapidly connect adjacent nodes through configuration space projections. To ensure continuity, we consider multiple nearest neighbors during expansion. Also, we propose a multi-seed strategy to further enhance roadmap quality.

Moreover, we introduce a straightforward yet effective approach to demonstrate the practical utility of this roadmap in teleoperation tasks. This method remains applicable even in scenarios where the GRR roadmap lacks full continuity or where users direct the robot to perform actions that might lead to illegal maneuvers, such as encountering a self-collision. Through our experiments in simulation, we illustrate that the proposed method not only generates higher-quality roadmaps more efficiently but also outperforms advanced IK solvers in teleoperation tasks, involving robots with up to 7 DOF.

Specifically, the contributions of this work include 1) the development of a new expansion algorithm for efficiently computing a smooth GRR roadmap, 2) the proposal of a multi-seed strategy resulting in higher quality GRR roadmaps, and 3) the establishment of a teleoperation pipeline utilizing the GRR roadmap that surpasses prior methods. The implementation of the proposed algorithm[1] is provided as open source.

II. Related Work

Redundancy resolution has historically been a core concept in robotics [3]. Its applications span a wide spectrum, including avoidance of singularities [4] and obstacles [5] as well as facilitating teleoperation tasks [6]. Traditionally, redundancy is resolved at the velocity level [7] or the torque level [3] through the utilization of projection operators. However, our work focuses on redundancy resolution at the position level [2], [8] where we compute a direct mapping from task-space point to robot configurations.

A commonly used strategy for addressing redundancy at the local position level is IK solvers. Examples include simple Jacobian-based IK solvers such as the Newton-Rapson method or more recent IK-solvers, such as Relaxed-IK [9] and Ranged IK [10], which incorporate more complicated constraints including self-collisions and singularity avoidance. These IK solvers exhibit high-speed performance, often processing at several iterations per second. However, they may get trapped in singularity or local minima during teleoperation and generate suboptimal solutions due to their lack of foresight and local problem-solving approaches.

Another category of methods that resolve redundancy at the position level while often avoiding the local minima is single-query planning methods [11]. Several planning methods that also incorporate task-space constraints [12] have been proposed, with a commonly used constrained sampling-based motion planning framework [13]. Nonetheless, the produced motions are often not repeatable as shown in Fig. 2a. To address this challenge, [14], [15] focus on producing cyclic repeatable paths that start and end at the same configuration. However, these methods are tailored towards producing a single repeatable path, while in this work, we aim to resolve all possible task-satisfying paths simultaneously.

Roadmap-based methods, e.g., probabilistic roadmaps (PRM) [16], compute an approximation of the connectivity of the configuration space through random sampling. Given a start and goal configuration, the roadmap can be queried with graph search, and retrieve a collision-free path. Our work also computes a roadmap but is tailored towards producing paths that additionally satisfy task constraints and the paths are smooth both in task space and configuration space. The work that is most similar to ours is [2], which formulated the global redundancy resolution problem, and proposed an algorithm to compute a GRR roadmap. In this work, we propose a new algorithm that generates the GRR roadmaps more efficiently and produces smoother paths. We also demonstrate how it can be used for real-time motion planning as in teleoperation.

Teleoperation is still a critical control mode of robotics, especially in safety-critical applications like surgery. There are many different teleoperating modes and interfaces depending on the application [17]. A common teleoperation interface is for humans to give commands in their natural 6-DOF task-space, which then has to be retargeted, to a kinematically different (and often redundant) robot [18]. To compute the retargeting, a common strategy is to use reactive IK-solvers [9], [10] and convert cartesian commands directly to the robot's configuration space. In this work, we showcase the utility of the computed GRR roadmap in teleoperation tasks, outperforming reactive-IK solvers, in terms of success rate and solution quality.

III. Notations and Problem Statement

A. Definitions And Notations

The general redundancy resolution problem aims to find a map $\mathcal{M}$ from the robot's Task Space ($\mathcal{T}$-space) to its Configuration Space ($\mathcal{C}$-space). $\mathcal{T}$-space represents the domain in which the robot performs its tasks, typically R^3 or $SE(3)$, while $\mathcal{C}$-space contains all potential configurations available to the robot. We will denote the elements of $\mathcal{T}$-space as $p \in \mathcal{T}$ and the elements of $\mathcal{C}$-space as $q \in \mathcal{C}$. Redundancy indicates that the dimension of $\mathcal{C}$-space is higher than that of $\mathcal{T}$-space, i.e., $dim(\mathcal{C}) > dim(\mathcal{T})$.

1) Local Redundancy Resolution: A map $\mathcal{M}$ is a local redundancy resolution if it is a transformation from $p \in \mathcal{T}$ to $q \in \mathcal{C}$ such that $\texttt{FK}(q) = p$, where FK is the forward kinematics function. For example, IK solvers are local resolutions at the point-wise level, and single-query planning methods are local resolutions at the path level.

2) Global Redundancy Resolution: A map $\mathcal{M}$ is defined as global redundancy resolution if it satisfies the $\texttt{FK}(q) = p$ constraint and additionally is an *injective*, *continuous* and *smooth*[2] function $\mathcal{F} : \mathcal{T} \rightarrow \mathcal{C}$. The injectivity property ensures that each point in $\mathcal{T}$-space is uniquely associated with a single configuration in $\mathcal{C}$-space, maintaining the consistency of the result. Continuity and smoothness guarantee that a small change in $\mathcal{T}$-space does not lead to discontinuous or large changes in $\mathcal{C}$-space, resulting in continuous and smooth robotic motions.

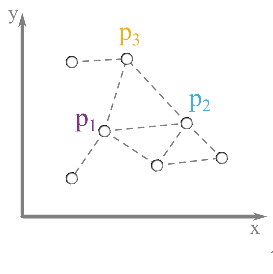
B. Approximate Global Redundancy Resolution

Given that finding an analytical form for $\mathcal{F}$ is intractable, the proposed method focuses instead on computing an approximation [2]. An illustrative example is shown in Fig. 3 for the same 3-link planar manipulator of Fig. 2 that operates in $\mathbb{R}^2$.

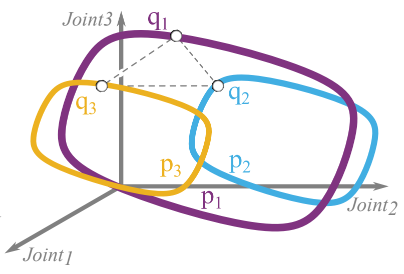
We first approximate $\mathcal{T}$-space with a discrete roadmap graph $G_p = (V_p, E_p)$. For the 3-link manipulator example, G_p

[1]https://github.com/elpis-lab/Expansion-GRR

[2]Technically, smoothness implies continuity, but since we measure them separately, we mention them both here.

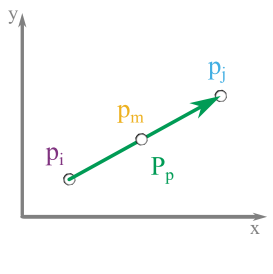


(a) $\mathcal{T}$-space Roadmap G_p

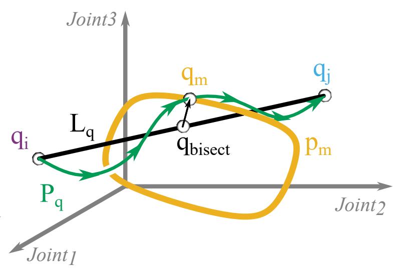


(b) $\mathcal{C}$-space Roadmap G_q

Fig. 3: An illustration of the roadmaps G_p and G_q. (a) The points p_1, p_2, and p_3 are neighboring samples in $\mathcal{T}$-space, in this case, the end-effector position in $\mathbb{R}^2$. (b) The solid lines indicate SMM of $\mathcal{T}$-space points. A roadmap G_q must select a single configuration q_i from each manifold, and additionally, ensure that the configurations, q_1, q_2 and q_3 of adjacent points, are "close enough" to pertain continuity and smooth transition in $\mathcal{C}$-space.



(a) Path in $\mathcal{T}$-space



(b) Path in $\mathcal{C}$-space

Fig. 4: An illustration of the continuity constraint. (a) The robot's end-effector follows a straight path P_p from p_i to its adjacent point p_j in $\mathcal{T}$-space (green). (b) The robot's corresponding configuration path P_q from q_i to q_j in $\mathcal{C}$-space (green) can be computed by projecting the intermediate configurations to their corresponding SMM in a bisection manner. Ideally, P_q should stay "close" to the straight line L_q delineated by q_i and q_j (black).

is shown in Fig. 3a, with the $\mathcal{T}$-space points $p_1, p_2, p_3 \in V_p$ as nodes and corresponding edges shown as dashed lines. In Fig. 3b, the solid lines indicate the set of configurations q that satisfy the constraints $\texttt{FK}(q) = p$ induced by p_1, p_2, and p_3, also known as the self-motion manifolds (SMM) [19]. To compute an approximation for $\mathcal{F}$, we need to build the corresponding $\mathcal{C}$-space roadmap G_q that satisfies the GRR properties:

1) **Injectivity:** For each $\mathcal{T}$-space vertex $p_i \in V_p$, there is only a single corresponding $\mathcal{C}$-space vertex $q_i \in V_q$, i.e, choosing a single configuration q from each self-motion manifold (as shown in Fig. 3b).
2) **Continuity:** For each $\mathcal{T}$-space edge $(p_i, p_j) \in E_p$, the configurations of the matching $\mathcal{C}$-space edge $(q_i, q_j) \in E_q$ satisfy the continuity constraint, formally defined in Sec. IV-A and shown in Fig. 4.
3) **Smoothness:** Minimize the length ratio of $\mathcal{C}$-space edges $(p_i, p_j) \in E_p$ to $\mathcal{T}$-space edges $(q_i, q_j) \in E_q$.

Concretely, the problem we are concerned with is: Given a graph G_p, compute a graph G_q that maximally satisfies 1), 2), and 3) while minimizing total computational time.

IV. Methodology

The main idea of the proposed method is to efficiently build the roadmap G_q by expanding from the selected initial configuration seeds. Again, for illustrative purposes, we will employ a 3-link planar manipulator operating in $\mathbb{R}^2$ task space (similar to that depicted in Fig. 2)

A. Continuity Constraint

To find a configuration q on the self-motion manifold that satisfies the constraint imposed by a point p, i.e., $\texttt{FK}(q) = p$, a common method employed is projection in $\mathcal{C}$-space [8], [20]. Starting with a given initial guess q_{guess}, an optimizer iteratively moves it closer to the desired self-motion manifold until a configuration q within tolerance is found. In this paper, as our projection operator, we use the Newton-Raphson IK, which is a Jacobian pseudo-inverse projection. Also, to ensure the returned configuration is valid a self-collision check is performed. We denote this function as $\texttt{Projection}(p, q_{quess})$.

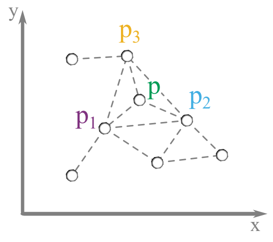
Continuity ensures that a small variation in $\mathcal{T}$-space does not lead to a discontinuous motion in $\mathcal{C}$-space. To achieve this, we enforce a motion constraint similar to [21]. An illustration with the 3-link planar manipulator is provided in Fig. 4. Let us assume we want to move the end effector from point p_i to its adjacent point p_j in a straight line P_p in $\mathcal{T}$-space. The motion constraint ensures that the corresponding configuration path P_q will not deviate significantly from the straight line L_q delineated by q_i and q_j in $\mathcal{C}$-space.

Fig. 4 illustrates visually how to perform this check. We project the bisected configuration q_{bisect} to the self-motion manifold of the middle point p_m and acquire the intermediate configuration q_m. To pass the continuity check, the deviation between q_m and q_{bisect}, should be within a threshold. As shown in Alg. 1 this process is repeated recursively to ensure the continuity check passes up to an ϵ resolution.

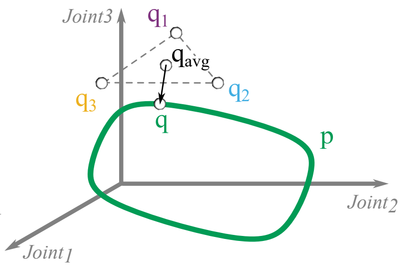
Algorithm 1: $\texttt{IsContinuous}(p_i, p_j, q_i, q_j)$

```
   /* Check threshold */
1  if $\texttt{D}_\texttt{q}(q_i, q_j) < \epsilon$ then
2      return $true$;
   /* Bisect in $\mathcal{T}$-space and $\mathcal{C}$-space */
3  $p_m \leftarrow \texttt{BisectP}(p_i, p_j)$;
4  $q_{bisect} \leftarrow \texttt{BisectQ}(q_i, q_j)$;
5  $q_m \leftarrow \texttt{Projection}(p_m, q_{bisect})$;
   /* Check deviation */
6  if $\texttt{Max}(\texttt{D}_\texttt{q}(q_i, q_m), \texttt{D}_\texttt{q}(q_m, q_j)) > c \cdot \texttt{D}_\texttt{q}(q_i, q_j)$ then
7      return $false$;
   /* Run recursively */
8  if $\texttt{IsContinuous}(p_i, p_m, q_i, q_m)$
9  and $\texttt{IsContinuous}(p_m, p_j, q_m, q_j)$ then
10     return $true$;
11 return $false$;
```

In Alg. 1, $\texttt{BisectP}$ finds the center of two $\mathcal{T}$-space points, and $\texttt{BisectQ}$ finds the center of two $\mathcal{C}$-space configurations. The parameter $c > 0.5$, regulates the maximum deviation. The value ϵ is the minimal threshold to stop the visibility checking process. In this paper, we choose c and ϵ based on the dimension of the $\mathcal{C}$-space, i.e., $c = 0.5\sqrt{(dim(\mathcal{C}))}$, and $\epsilon = 0.05\sqrt{(dim(\mathcal{C}))}$. These values are chosen as reasonable default values that scale with the dimension of the $\mathcal{C}$-space.



(a) $\mathcal{T}$-space Roadmap G_p



(b) $\mathcal{C}$-space Roadmap G_q

Fig. 5: An illustration of projecting from multiple neighbors. (a) The points p_1, p_2, and p_3 are neighboring points of p in $\mathcal{T}$-space. (b) The solid lines indicate the self-motion manifold of p. A weighted average q_{avg} is computed with neighbors' corresponding configurations q_1, q_2, and q_3 in $\mathcal{C}$-space. It is then projected onto the self-motion manifold to find q.

B. Projection From Multiple Neighbors

Now we describe the proposed method for adding new configurations in an existing G_q roadmap, also illustrated with an example in Fig. 5.

One strategy is to choose a configuration from a $\mathcal{T}$-space neighbor of p e.g., q_3, and project it to get q, i.e, $q = \texttt{Projection}(p, q_3)$. However, since point p has multiple neighbors, projecting from a single one will create a bias towards this neighbor. For example, $\texttt{Projection}(p, q_3)$ could result in a configuration that is far from q_1 and q_2, resulting in discontinuities. Instead, we adopt a more general strategy of computing a weighted average configuration q_{avg} in $\mathcal{C}$-space from all neighboring configurations q_1, q_2, and q_3 as shown in Fig. 5. By aggregating neighboring configurations, the resulting q derived from $\texttt{Projection}(p, q_{avg})$ has a higher chance of being similar to all its neighbors. This similarity feature aids in improving overall smoothness and continuity.

We now explain this process more formally, with the pseudo-code in Alg. 2. First, we denote a function that returns the k nearest neighbors $neighbors$ of point p in the $\mathcal{T}$-space graph G_p, as $\texttt{NearestNeighbors}(p, k, G_p)$. We also define the $\mathcal{T}$-space distance metric $\texttt{D}_\texttt{p}$ adapted from [22] as:

$$\begin{aligned} \texttt{D}_\texttt{p}(p_i, p_j) &= w_t \cdot \texttt{D}_{\mathbb{R}^3}(t_i, t_j) + w_o \cdot \texttt{D}_{\texttt{SO(3)}}(o_i, o_j), \text{where} \\ \texttt{D}_{\mathbb{R}^3}(t_i, t_j) &= \|t_i - t_j\| \qquad (1) \\ \texttt{D}_{\texttt{SO(3)}}(o_i, o_j) &= 1 - |o_i \cdot o_j| \end{aligned}$$

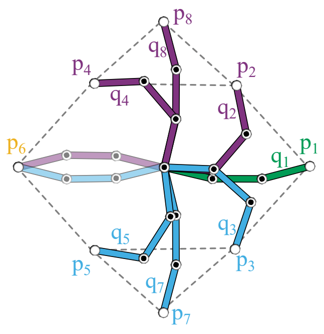
and t, o denote the translation and orientation (represented by a quaternion) component of a $\mathcal{T}$-space point. $\|t_i - t_j\|$ is the standard Euclidean norm and $|o_i \cdot o_j|$ is the absolute value of the inner product, while w_t and w_o are chosen weights to scale the distances. In this paper, we set $w_t = 1$ and $w_o = 0.3$. If $\mathcal{T} = \mathbb{R}^3$, we set $w_o = 0$.

After locating the $neighbors$ of a point p, we collect a list of configurations qs from them. Their corresponding weights ws are determined based on their $\mathcal{T}$-space distances ds and scale towards the closer neighbors. Lastly, we compute the weighted average configuration q_{avg} and project it to find q.

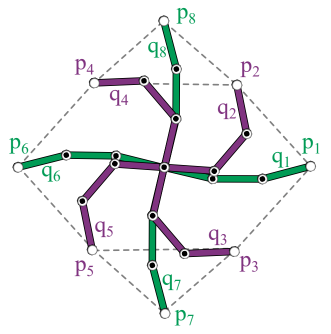
C. Seeding Strategy

The techniques discussed in the previous sections can be used to expand the roadmap G_q by incrementally adding new nodes. However, the choice of the initial configuration(s) that the roadmap is seeded with, also has a significant impact on the final result. A simple seeding strategy would be to initialize G_q with a random configuration and expand from there. Yet, this simple seeding method is prone to create discontinuities in $\mathcal{C}$-space. Fig. 6a illustrates an example with the simple 3-link manipulator. Starting from an initial seed q_1 such that $\texttt{FK}(q_1) = p_1$, one set of generated configurations are "elbow-down" configurations (blue), while the other set is "elbow up" (purple). This results in a discontinuity at the p_6 task point, since no configuration (either purple or blue) will satisfy the continuity constraint of both sides.

Ideally, the seeds should be situated within $\mathcal{C}$-space to minimize the probability of encountering discontinuities. To this aim, we propose an alternative multi-seeding strategy, as depicted in Fig. 6b. Given that the final roadmap must encompass all potential cyclic paths, we propose to initialize the roadmap with configurations from a continuous cyclic path, shown as the green configurations in Fig. 6b. While there exist algorithmic approaches to generating cyclic paths [14], [15], we opted to manually create one per robot, since only a single cyclic path is required for our purposes.



(a) Starting with One Seed



(b) Starting with Multiple Seeds

Fig. 6: (a) From an initial seed $q1$ (green), q_2, q_8 to q_4 are sequentially generated (purple), and q_3, q_7 to q_5 are also sequentially generated (blue). For q_6, however, expanding from either side will lead to a configuration that is not continuous to the other. (b) Starting from seeds q_1, q_7, q_8 and q_6 (green), deriving from an initial cyclic path, we could achieve a fully connected $\mathcal{T}$-space roadmap with all "elbow-down" configurations (purple).

Algorithm 2: $\texttt{ProjectNeighbors}(p, G_p, G_q)$

```
   /* Collect qs and ds from neighbors */
1  $neighbors \leftarrow \texttt{NearestNeighbors}(p_i, k, G_p)$;
2  $qs \leftarrow$ Empty configuration list;
3  $ds \leftarrow$ Empty distance list;
4  foreach $j \in neighbors$ do
5      if $q_j$ is in $G_q$ then
6          $qs.append(q_j)$;
7          $ds.append(D_p(p, p_j))$;
   /* Compute weights */
8  $ws \leftarrow$ Weight list with the same size as $qs$;
9  foreach $j \in neighbors$ do
10     $w_i \leftarrow (\text{Max}(ds)/d_i)^2$
11 $\texttt{Normalize}(ws)$;
   /* Compute and project from $q_{avg}$ */
12 $q_{avg} \leftarrow \texttt{WeightedAverage}(qs, ws)$;
13 return $\texttt{Projection}(p, q_{avg})$;
```

D. Global Expansion

In this section, we describe how we can use the tools mentioned in the previous sections to build the $\mathcal{C}$-space roadmap $G_q = (V_q, E_q)$. We start by discretizing $\mathcal{T}$-space and generating the $\mathcal{T}$-space roadmap $G_p = (V_p, E_p)$. Although several choices exist, in this paper we use grid discretization. Given the selected initial configurations qs_{init} and roadmap G_p, Alg. 3 describes, how we incrementally expand the roadmap in $\mathcal{C}$-space by projection from nearest neighbors in a Breadth First Search (BFS) manner.

First, we initialize the $\mathcal{C}$-space roadmap G_q, a queue Q, and the visited set V with qs_{init} (line 1-line 3). Similar to BFS, it iteratively dequeues an index i from Q and explores its k nearest neighbors in the $\mathcal{T}$-space (line 4-line 11). For each point p_i that does not have a corresponding configuration, it uses Alg. 2 to project the weighted average configuration and find the solution q_i for the point p_i (line 12). Finally, it runs `IsContinuous`(Alg. 1) checks with all neighbors to verify continuity, and the results will be used to update G_q (line 13-line 14).

Algorithm 3: `GlobalExpansion`(G_p, qs_{init})

```
   /* Initialize */
1  Initialize $\mathcal{C} - space$ roadmap $G_q$ with $qs_{init}$;
2  $Q \leftarrow$ A queue with each $i \in qs_{init}$;
3  $V \leftarrow$ A set with each $i \in qs_{init}$;
4  while $Q \neq \emptyset$ do
       /* Expand in BFS order */
5      $i \leftarrow Q.dequeue()$;
6      foreach $j \in$ NearestNeighbors$(p_i, k, G_p)$ do
7          if $j \notin V$ then
8              $V.add(j)$;
9              $Q.enqueue(j)$;
10     if $q_i$ is in $G_q$ then
11         continue;
       /* Project from neighbors */
12     $q_i \leftarrow$ ProjectNeighbors$(p_i, G_p, G_q)$;
       /* Continuity check and update */
13     Run IsContinuous check with each neighbor;
14     Add $q_i$ and valid edges into $G_q$;
```

E. Utilizing The Roadmap

Now, we introduce some potential use cases for the computed GRR roadmap.

1) IK Solver: To use the GRR map as an IK solver, we need to generalize it to the continuous space for all $p \in \mathcal{T}$. To this aim, the process described in Alg. 2 can still be employed to find the corresponding configuration q for any given point p. However, the functionality of `NearestNeighbor` will be different when discontinuous edges are present in the local neighboring subgraph of G_q. The function should only consider neighbors in the largest connected component of the subgraph, ignoring any disconnected neighbors. Another way to view this process is to use the GRR roadmap to get a good q_{guess}, and then use an off-the-shelf IK-solver to project it to the given task-point constraint.

2) $\mathcal{T}$-space Path Planning: To find a path from a starting configuration to a goal configuration, we can directly plan a path in the configuration roadmap G_q. However, if it is desired to have the end-effector follow the edges of the $\mathcal{T}$-space roadmap G_p closely, we can instead search and plan a path P_p in roadmap G_p. We then interpolate the points along P_p and for each interpolated waypoint p, we can run `ProjectNeighbors` to find its corresponding configuration q and build the $\mathcal{C}$-space path P_q. This ensures the end effector will closely follow the edges of G_p. Also, since the interpolation happens along the edges, the configuration path P_q will always be feasible and continuous because the edges have passed the `IsContinuous` check.

3) Teleoperation: Given a real-time Cartesian command from a human operator, we can produce the corresponding configuration as described in Sec. IV-E.1. However, since the human input will not necessarily follow the edges of the roadmap, the produced configuration q is not guaranteed to be feasible. It can get into self-collision or a discontinuous region if one exists. In this case, we can leverage the roadmaps and run planning as in Sec. IV-E.2 to detour from these regions.

We design the teleoperation pipeline as follows. Given the current configuration q_c and a new human input p_t as the target, we compute the next configuration q_t by running `ProjectNeighbors`. We verify the feasibility of going from q_c to q_t by testing it with `IsContinuous`. We execute it if feasible. Otherwise, instead of going towards the human input p_t, we change the target from p_t to p_t's valid nearest neighbor p_n in G_p. Such a change ensures the robot still follows human commands closely without getting into an infeasible zone. Once the human input p_t becomes feasible again, we leverage planning as described in Sec. IV-E.2 to plan a path from q_c to the new valid target q_t and continue tracking human input.

V. Experiments

We evaluate the performance of our method in two aspects. First, we assess the quality of the GRR roadmap relative to the criteria set in Sec. III. Second, we emulated teleoperation tasks and compared the results obtained from the GRR roadmaps and different IK-solvers. All the experiments were run on a machine with Intel i5-10400 and 16 GB memory.

A. Roadmap Quality Metrics

To measure the quality of the resolution roadmap G_q, two metrics are used:

1) Roadmap Connectivity: A fully connected $\mathcal{C}$-space roadmap G_q may not exist or may be difficult to find, resulting in a discontinuous graph. We quantify this concept of connectivity with the metric:

$$C(G_q) = \frac{N_{E_q}}{N_{E_p}} \tag{2}$$

where N_{E_q} is the size of the $\mathcal{C}$-space edge set E_q, and N_{E_p} is the size of the $\mathcal{T}$-space edge set E_p. Thus, a fully connected roadmap $U(G_q) = 1$ refers to a $\mathcal{C}$-space roadmap that has as many edges as the $\mathcal{T}$-space roadmap.

TABLE I: Roadmap Quality

Robot	Dim($\mathcal{C}$)	Dim($\mathcal{T}$)	Vertices	Method	Connectivity (%)	Smoothness	Building Time (min)
5-link Manipulator Position	5	2	1,013	RANDOM-GRR	**100.00**	16.853	2.023
				EXPANSION-GRR	**100.00**	**5.675**	**0.095**
5-link Manipulator Position & Fixed Rotation	5	3	1,013	RANDOM-GRR	**100.00**	24.528	1.862
				EXPANSION-GRR	**100.00**	**8.992**	**0.150**
Kinova Position	7	3	3,299	RANDOM-GRR	**100.00**	9.223	22.971
				EXPANSION-GRR	**100.00**	**2.563**	**0.201**
Kinova Position & Fixed Rotation	7	6	3,299	RANDOM-GRR	**100.00**	11.205	72.013
				EXPANSION-GRR	**100.00**	**4.299**	**0.196**

2) Roadmap Smoothness: Another metric that evaluates the roadmap quality is the smoothness, approximated by the averaged ratio of $\mathcal{C}$-space edge distance to $\mathcal{T}$-space edge distance:

$$S(G_q) = \frac{1}{N_{E_q}} \sum_{(i,j)\in E_q} \frac{\mathtt{D_q}(q_i, q_j)}{\mathtt{D_p}(p_i, p_j)} \tag{3}$$

where N_{E_q} is the size of the $\mathcal{C}$-space edge set E_q. $\mathtt{D_q}$ and $\mathtt{D_p}$ are functions to compute the distance in $\mathcal{C}$-space and $\mathcal{T}$-space.

B. Roadmap Quality Experiment

We compare the roadmap connectivity, smoothness, and building time with the GRR method proposed in [2]. Instead of using multi-seeding and projections, [2] builds the GRR map by first randomly sampling numerous $\mathcal{C}$-space configurations at each $\mathcal{T}$-space point. It then connects all the adjacent pairs that pass the continuity test. Eventually, a constraint satisfaction problem (CSP) solver is used to select a single configuration for each point. We refer to this method as RANDOM-GRR. In all experiments, we used 100 random samples per $\mathcal{T}$-space point.

We tested in simulation two robots, a 5-link planar manipulator (5-DOF), and a Kinova Gen3 arm (7-DOF) with a Robotiq-85 gripper (shown in Fig. 1. The 5-link planar manipulator joints are continuous (have no limits) and self-intersections are allowed. The tested $\mathcal{T}$-space for the planar manipulator are $\mathcal{T} = \mathbb{R}^2$ and $\mathcal{T} = SE(2)$, and the latter incorporates a fixed rotation where the end-effector is always facing right. The Kinova robot was tested with $\mathcal{T} = \mathbb{R}^3$ and $\mathcal{T} = SE(3)$, with the latter incorporating a fixed rotation where the end-effector is consistently facing downwards.

The results are given in Table I. Both RANDOM-GRR and EXPANSION-GRR can fully resolve the global redundancy. However, EXPANSION-GRR outperforms RANDOM-GRR by building a smoother roadmap up to 2 orders of magnitude faster. We attribute the improved smoothness to the proposed projections which minimize the distances between neighboring configurations. Also, by utilizing multi-seeding and projection, we can avoid the time-consuming steps caused by extensive sampling in random-GRR.

C. Teleoperation Experiment

To evaluate the performance of our method in teleoperation, we designed four geometric path tracing tasks, which provide commands to robots in $\mathcal{T}$-space:

- Random Line: A line defined by two random points.
- Self-crossing Line: A line as in Random Line, but it passes through the robot's base.
- Random Circle: A planar circle defined by a random center point, up vector, and radius.
- Partially Reachable Circle: A planar circle as in Random Circle, but part of it goes beyond the reachable space.

For each task, 100 paths are randomly generated. Intermediate waypoints are sent sequentially to the solvers at 50 Hz over 4 seconds, i.e. 200 waypoints per path to follow, simulating a human input action.

To use EXPANSION-GRR and RANDOM-GRR for teleoperation we use the teleoperation pipeline we proposed in Sec. IV-E.3, and compare against Newton-Raphson IK and Relaxed IK [9]. The weights of the Relaxed IK terms were chosen to minimize deviation from input as suggested in [9]. We measure the following three metrics:

1) Deviation From Input: The deviation from the produced path to the input path in $\mathcal{T}$-space is measured by finding the pairs of matched waypoints with dynamic time warping and then computing the average distance between these pairs.

2) Path Smoothness: We define path smoothness similar to roadmap smoothness, as the average distance ratio of the produced $\mathcal{C}$-space path to that of the produced $\mathcal{T}$-space path.

3) Success Rate: The success is determined by whether the produced $\mathcal{T}$-space path follows the input path to the goal. We marked it as a failure if the produced path gets stuck due to local minima or self-collision.

For a fair comparison, we calculate deviation and smoothness only for the problems that all methods succeed.

We ran the tasks with the Kinova arm in $\mathcal{T} = \mathbb{R}^3$ with fixed rotation and the aggregated results are summarized in Table II. The IK solvers often got stuck in local minima at line-related tasks and encountered singularities in circle-related tasks. RANDOM-GRR had the lowest success rate, as it tended to produce discontinuous paths from the non-smooth map. The proposed method EXPANSION-GRR was able to achieve 100% success rate in all types of tasks. Regarding deviation from input, EXPANSION-GRR performed as well as Newton-IK in tasks Random-Line and Random-Circle, but in the more challenging tasks Self-Crossing Line and Partially Reachable Circle, our method exhibited higher deviation due to the need for more planning. For smoothness, the IK-solvers achieved the best results for each $\mathcal{T}$-space path. This result is expected as our method prioritizes global consistency and continuity over smoothness. Nevertheless, EXPANSION-GRR still produces smoother paths compared to RANDOM-GRR.

TABLE II: Teleoperation Experiment

Type	Methods	Deviation From Input	Path Smoothness	Success Rate %
Random Line	Newton IK	**0.011**	**4.395**	90
	Relaxed IK	0.023	4.569	78
	RANDOM-GRR	1.984	6.432	49
	EXPANSION-GRR	**0.011**	5.071	**100**
Self-crossing Line	Newton IK	0.479	7.882	30
	Relaxed IK	**0.184**	**4.236**	54
	RANDOM-GRR	2.547	5.760	34
	EXPANSION-GRR	0.461	5.481	**100**
Random Circle	Newton IK	**0.022**	4.284	98
	Relaxed IK	0.023	**3.986**	98
	RANDOM-GRR	0.458	6.974	77
	EXPANSION-GRR	**0.022**	4.664	**100**
Partially Reach-able Circle	Newton IK	**0.085**	6.138	99
	Relaxed IK	0.102	**3.997**	95
	RANDOM-GRR	0.652	6.428	78
	EXPANSION-GRR	0.166	5.200	**100**

VI. Discussion

In this paper, we presented a new method to generate smooth GRR roadmaps in an efficient manner. We also demonstrated these properties compared to the prior art and illustrated how they can be used for teleoperation tasks.

One of the limitations of our work is that it uses all of the available redundancy of the robot to satisfy the $\mathcal{T}$-space constraints, making it challenging to meet other secondary objectives. In the future we would like to address this problem by building more flexible roadmaps similar to [8], [23]. The selection of initial seeds is another interesting problem to investigate. One potential avenue is to generate cyclic paths using [14], [15]. Although our method achieved full connectivity in the tested examples, this might not always be possible e.g., when obstacles are present. Nonetheless, we expect the roadmap to still be usable as in the example with the infeasible query of the self-crossing lines.

Finally, we would like to investigate applications of GRR roadmaps beyond teleoperation such as motion planning, dimensional reduction, and generating motion primitives.

References

[1] S. Chiaverini, “Redundant Robots,” in *Encyclopedia of Systems and Control*, J. Baillieul and T. Samad, Eds. London: Springer, 2015, pp. 1141–1150. [Online]. Available: https://doi.org/10.1007/978-1-4471-5058-9_173

[2] K. Hauser and S. Emmons, “Global Redundancy Resolution via Continuous Pseudoinversion of the Forward Kinematic Map,” *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 3, pp. 932–944, July 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8325410/

[3] O. Khatib, “A unified approach for motion and force control of robot manipulators: The operational space formulation,” *IEEE Journal on Robotics and Automation*, vol. 3, no. 1, pp. 43–53, Feb. 1987. [Online]. Available: https://ieeexplore.ieee.org/document/1087068

[4] S. Chiaverini, “Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators,” *IEEE Transactions on Robotics and Automation*, vol. 13, no. 3, pp. 398–410, June 1997. [Online]. Available: https://ieeexplore.ieee.org/document/585902

[5] A. A. Maciejewski and C. A. Klein, “Obstacle Avoidance for Kinematically Redundant Manipulators in Dynamically Varying Environments,” *The International Journal of Robotics Research*, vol. 4, no. 3, pp. 109–117, Sept. 1985. [Online]. Available: https://doi.org/10.1177/027836498500400308

[6] C. Brooks, W. Rees, and D. Szafir, “Assistance in Teleoperation of Redundant Robots through Predictive Joint Maneuvering,” *ACM Transactions on Human-Robot Interaction*, Nov. 2023. [Online]. Available: https://dl.acm.org/doi/10.1145/3630265

[7] C. A. Klein and C.-H. Huang, “Review of pseudoinverse control for use with kinematically redundant manipulators,” *IEEE Transactions on Systems, Man, and Cybernetics*, no. 2, pp. 245–250, Mar. 1983. [Online]. Available: http://ieeexplore.ieee.org/document/6313123/

[8] A. Albu-Schäffer and A. Sachtler, “Redundancy Resolution at Position Level,” *IEEE Transactions on Robotics*, vol. 39, no. 6, pp. 4240–4261, Dec. 2023. [Online]. Available: https://ieeexplore.ieee.org/document/10246229/

[9] D. Rakita, B. Mutlu, and M. Gleicher, “RelaxedIK: Real-time Synthesis of Accurate and Feasible Robot Arm Motion,” in *Robotics: Science and Systems*, June 2018. [Online]. Available: http://www.roboticsproceedings.org/rss14/p43.pdf

[10] Y. Wang, P. Praveena, D. Rakita, and M. Gleicher, “RangedIK: An Optimization-based Robot Motion Generation Method for Ranged-Goal Tasks,” in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, May 2023, pp. 9700–9706. [Online]. Available: https://ieeexplore.ieee.org/document/10161311/?arnumber=10161311

[11] A. Orthey, C. Chamzas, and L. E. Kavraki, “Sampling-based motion planning: A comparative review,” *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 7, no. 1, p. null, July 2024. [Online]. Available: https://doi.org/10.1146/annurev-control-061623-094742

[12] D. Berenson, S. Srinivasa, and J. Kuffner, “Task Space Regions: A framework for pose-constrained manipulation planning,” *The International Journal of Robotics Research*, vol. 30, no. 12, pp. 1435–1460, Oct. 2011. [Online]. Available: http://journals.sagepub.com/doi/10.1177/0278364910396389

[13] Z. Kingston, M. Moll, and L. E. Kavraki, “Sampling-Based Methods for Motion Planning with Constraints,” *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, no. 1, pp. 159–185, May 2018. [Online]. Available: https://www.annualreviews.org/doi/10.1146/annurev-control-060117-105226

[14] G. Oriolo and C. Mongillo, “Motion Planning for Mobile Manipulators along Given End-effector Paths,” in *International Conference on Robotics and Automation (ICRA)*. Barcelona, Spain: IEEE, 2005, pp. 2154–2160. [Online]. Available: http://ieeexplore.ieee.org/document/1570432/

[15] 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, Oct. 2017. [Online]. Available: http://ieeexplore.ieee.org/document/7982689/

[16] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, “Probabilistic roadmaps for path planning in high-dimensional configuration spaces,” *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, Aug. 1996. [Online]. Available: https://ieeexplore.ieee.org/document/508439

[17] G. Niemeyer, C. Preusche, S. Stramigioli, and D. Lee, “Telerobotics,” in *Springer Handbook of Robotics*, ser. Springer Handbooks, B. Siciliano and O. Khatib, Eds. Cham: Springer International Publishing, 2016, pp. 1085–1108. [Online]. Available: https://doi.org/10.1007/978-3-319-32552-1_43

[18] N. Boguslavskii, Z. Zhong, L. M. Genua, and Z. Li, “A Shared Autonomous Nursing Robot Assistant with Dynamic Workspace for Versatile Mobile Manipulation,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2023, pp. 7040–7045. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10342401

[19] J. Burdick, “On the inverse kinematics of redundant manipulators: characterization of the self-motion manifolds,” in *1989 International Conference on Robotics and Automation Proceedings*, vol. 1, May 1989, pp. 264–270. [Online]. Available: https://ieeexplore.ieee.org/document/99999

[20] Z. Kingston, M. Moll, and L. E. Kavraki, “Exploring implicit spaces for constrained sampling-based planning,” *The International Journal of Robotics Research*, vol. 38, no. 10-11, pp. 1151–1178, Sept. 2019. [Online]. Available: http://journals.sagepub.com/doi/10.1177/0278364919868530

[21] R. H. Taylor, "Planning and Execution of Straight Line Manipulator Trajectories," *IBM Journal of Research and Development*, vol. 23, no. 4, pp. 424–436, July 1979. [Online]. Available: https://ieeexplore.ieee.org/document/5390800

[22] J. Kuffner, "Effective sampling and distance metrics for 3D rigid body path planning," in *IEEE International Conference on Robotics and Automation*, vol. 4, Apr. 2004, pp. 3993–3998. [Online]. Available: https://ieeexplore.ieee.org/document/1308895

[23] H. Yao, R. Laha, L. F. C. Figueredo, and S. Haddadin, "Enhanced dexterity maps (edm): A new map for manipulator capability analysis," *IEEE Robotics and Automation Letters*, vol. 9, no. 2, pp. 1628–1635, 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10368298