Continuous Pseudoinversion of a Multivariate Function: Application to Global Redundancy Resolution

Kris Hauser

Department of Electrical and Computer Engineering, Duke University,
kris.hauser@duke.edu

Abstract. This paper seeks to generate a continuous pseudoinverse of a function that maps a higher dimensional compact set to a lower dimensional one. Continuity and smoothness should be attained if possible, but otherwise the volume of the discontinuity boundary should be minimized. A sampling-based approximation technique is presented that uses discretized roadmaps of both the domain and image, and minimizes discontinuities of the inverse function. The method is applied to kinematic redundancy resolution for redundant robots, which have more degrees of freedom than workspace dimensions. The output is a global redundancy resolution, which has the convenient property that whenever the robot returns to the same workspace point, it uses the same joint-space pose. If a global resolution cannot be found, then the method minimizes discontinuities and maps them in workspace. Results are demonstrated on toy problems with up to 20 DOF, and on several robot arms.

1 Introduction

Function inversion is frequently encountered in many fields of science and engineering, including robotics, computer graphics, control, and mechanical design. Past techniques typically consider calculating a point-wise inverse, i.e., find a point x such that $f(x) = y$. In contrast, this paper is interested in generating a functional map of the inverse across entire regions of space, i.e., find a function $f^{-1}(y)$ such that $f(f^{-1}(y)) = y$ across all y. This is often an underconstrained problem, because the preimage of a point in the range may contain multiple or an infinite number of points in the domain and some inverses may be wildly varying or highly discontinuous. This paper seeks a systematic way to obtain maximally continuous and smooth inverse functions in high-dimensional spaces.

We ground this problem in the terminology of forward and inverse kinematics (IK), which is likely to be more familiar for readers in robotics. For example, consider the IK problem for a 2R planar robot manipulator restricted by joint limits (Fig. 1). Here the forward map f maps joint angles to Cartesian end effector (workspace) points via forward kinematics. Points in the CCW extremes of the workspace are reachable only with "elbow-down" configurations, while points in CW extremes are reachable only with "elbow-up" configurations. In

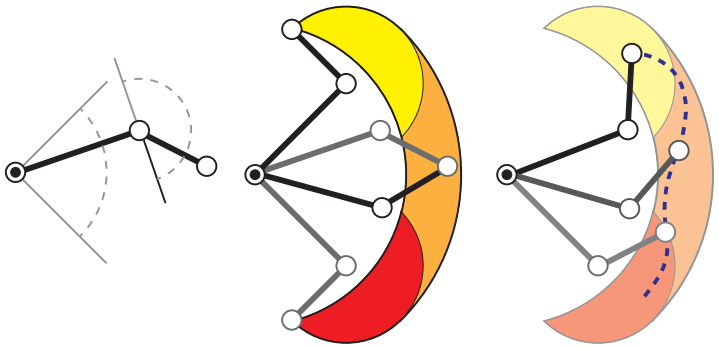

Fig. 1. A planar 2R manipulation with joint limits of $\pm 45^\circ$ and $\pm 90^\circ$, respectively. There are no joint space resolutions of Cartesian end effector paths (e.g., the dotted curve) in the interior of the reachable workspace from the upper region to the lower region.

between, both elbow-down and elbow-up configurations are valid. Observe that any workspace path from the CCW extreme to the CW extreme must cause the robot to at some point flip between elbow down and elbow up. In fact, there is no continuous solution for such a workspace path unless it touches the workspace boundary, causing the robot to pass through a singularity.

If this robot were to be changed to a kR manipulator with $k \geq 3$ (i.e., a redundant manipulator), would a continuous inverse function exist? In contrast to the 2R case, there are an continuous infinity of inverses at each point, and it may seem as though this leaves sufficient flexibility to choose an everwhere-continuous inverse map. Hence, it is perhaps surprising that this is not the case, and in fact no solution exists for many settings of joint limits and link lengths.

This paper describes the redundancy problem for general robots and workspaces, and presents approximate algorithms for computing answers to questions of redundancy resolution existence and optimality. We draw a distinction between *pointwise*, *pathwise*, and *global* redundancy resolution, listed in order of increasing restrictiveness (Fig. 2). Pathwise redundancy resolution is the problem of generating a continuous configuration-space path for a given workspace path. While local methods may get stuck, we present a simple probabilistically complete technique that inspires our work in global redundancy resolution. Global redundancy resolution has the property that every workspace cycle causes the robot to return to the same configuration. This may be a useful property to make robots behave more predictably during Cartesian movement than either pathwise or pointwise resolution, in which the robot may adopt different poses depending on the workspace path taken, or may get stuck in local minima.

We present an approximate global resolution algorithm that generates a roadmap of sampled points in the workspace and associates each point with a set of configuration samples in its preimage. A configuration space roadmap is then generated along these samples, and a constraint satisfaction optimization (MAX-

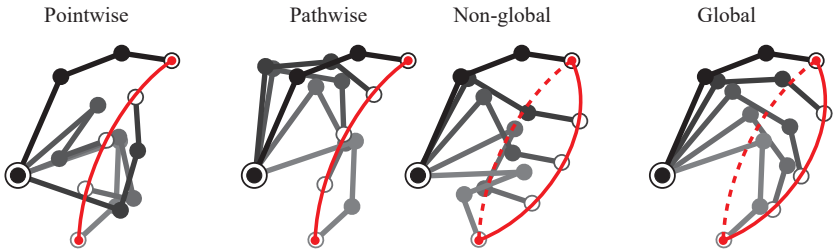

Fig. 2. Illustrating three types of redundancy resolution. Pointwise resolution gives a solution for each point, but does not guarantee continuity along a path. Pathwise resolution gives a continuous solution along a workspace path, but the robot may return to the starting point with a different configuration, and some paths may not have a valid resolution from a given starting configuration. Global resolution gives a continuous solution such that every cyclic path returns to the same configuration.

CSP) is applied to generate a pseudoinverse map that minimizes the number of discontinuous edges. The algorithm is applied to kinematic models of the Rethink Robotics Baxter, Boston Dynamics ATLAS, Kinova Jaco, NASA's Robonaut2, and JPL's Robosimian, showing that some robots are nearly globally-resolvable and others are not. Moreover, it calculates rich workspace maps that help visualize the boundaries of discontinuity in the optimized redundancy resolution.

2 Related work

Redunancy resolution is an old topic in robotics with many pointwise heuristics proposed to avoid singularities, joint limits, and obstacles [7]. Pathwise redundancy resolution has also long been studied, with several researchers identifying the problem of local minima of pointwise resolution methods [3, 6, 8, 10]. Our algorithm is most closely related to [8], who use a randomized tree-growing approach to explore self-motion manifolds along a discretized workspace path.

Unlike planning a path, which is a 1-dimensional map to configuration space, the global redundancy resolution problem is concerned with computing an m-dimensional map to configuration space. The closest related works in global resolution are [2, 5]. The method of [5] constructs a topological network of connected components of self-motion manifolds across a discretization of the workspace, and was applied to planar problems. However it is difficult to analyze connected components for arbitrary robots with constraints. The method of [2] generates IK tables via local optimization and selection of similar IK solutions to yield a smooth mapping for each leg of the Robosimian robot. These tables were then used to simplify motion planning for legged locomotion trajectories in [9]. We use a similar optimization method, although in this case, using coordinate descent, to smooth our maps in postprocessing as described in Sec. 5.2.

Global redundancy resolution has several potential applications. They could be used to select positions for a mobile base for a robot to perform certain

Cartesian movements [15]. Qualitatively, we observe these discontinuities correspond to workspace regions that are frustrating for teleoperation [4]. Finally, these mappings may also be useful for reduced-dimensionality motion planning in Cartesian space [9].

3 Problem definition

We wish to compute a continuous pseudoinverse of a functional mapping between a higher dimensional compact, bounded set and a lower dimensional compact, bounded set. These concepts and their equivalents in kinematic redundancy resolution are defined here.

Function pseudoinverse. A pseudoinverse of a surjective function $f : A \to B$ is an injective function $f^+ : f[A] \to A$ such that each element of the image $f[A]$ is mapped to an element in its preimage, i.e., $f(f^+(y)) = y$ for all $y \in f[A]$.

A pseudoinverse always exists even though an inverse may not, and if f is a bijection, then the pseudoinverse is identically the inverse. We also note that when A is of higher dimension than B, the preimage of each element in $f[A]$ is, in general, an infinite set, and hence the number of pseudoinverses is infinite. Although it may be easy to compute a pseudoinverse *pointwise*, e.g., by Newton's method, pointwise pseudoinverses often do not satisfy desirable properties, such as continuity or smoothness.

Continuous pseudoinverse. A continuous pseudoinverse f^+ is a pseudoinverse that is continuous in the parameters y across all of $f[A]$.

Through the rest of the paper we will use robot kinematics terminology as follows. Here, $\mathcal{C}$ is a configuration space (C-space) consisting of the robot's degrees of freedom, $\mathcal{W}$ is the Cartesian workspace, and $f : \mathcal{C} \to \mathcal{W}$ is the forward kinematics mapping. The workspace typically contains either position or orientation of the end effector, or both. In redundant problems, we have $dim(\mathcal{C}) > dim(\mathcal{W})$. We will use q to denote configurations and y to denote workspace points. Define the free space $\mathcal{F} \subseteq \mathcal{C}$ as the subset of configurations that lie within joint bounds and are collision-free. Define the reachable workspace $\mathcal{W}_C \subseteq \mathcal{W}$ as the subset reachable by free space configurations $\mathcal{W}_C = f[\mathcal{F}]$.

Pointwise redundancy resolution. A function f^+ is a pointwise resolution if it is a pseudoinverse of f over $\mathcal{W}_C$.

Pathwise redundancy resolution. A function f^+ is a pathwise resolution over the workspace path $y : [0, 1] \to \mathcal{W}_C$ if it is a continuous pseudoinverse over the path. In other words, $f(f^+(y(t))) = y(t)$ and $\lim_{u \to t} f^+(y(u)) = f^+(y(t))$ for all $t \in [0, 1]$. In this case we call $q(t) \equiv f^+(y(t))$ the resolved path. We also speak of *endpoint constrained pathwise resolution* where boundary conditions $q_0 = f^+(y(0))$ and/or $q_1 = f^+(y(1))$ exist at one or both of the endpoints.

Global redundancy resolution. A function f^+ is a global resolution if it is a continuous pseudoinverse of f over $\mathcal{W}_C$.

We will say that a problem is (pointwise, pathwise, or globally) resolvable if a (pointwise, pathwise, or global) resolution exists. It is straightforward to observe that global $\implies$ pathwise $\implies$ pointwise resolvability. (In fact, pointwise

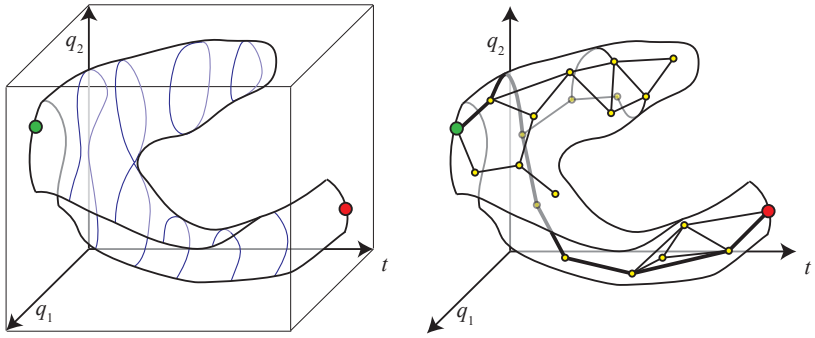

Fig. 3. Left: Illustrating the pathwise redundancy resolution problem. Right: a probabilistically complete roadmap-based solution.

resolvability holds true by definition.) As discussed above, the converse does not necessarily hold true.

4 Pathwise Redundancy Resolution

As a preliminary step, we will describe an algorithm to solve the simpler problem of pathwise redundancy resolution, illustrated in Fig. 3. At any workspace point, the kinematic constraint limits the set of valid points to a submanifold of C-space known as the self-motion manifold, and boundaries are introduced to the manifold due to feasibility constraints (e.g., joint limits). As the path parameter sweeps from 0 to 1, the self-motion manifolds sweeps out a manifold of one higher dimension in the Cartesian product $[0, 1] \times \mathcal{C}$. The goal is to find a path along this manifold with a monotonically increasing path parameter.

First, let us define a few commonly used subroutines.

- Solve(y, q_{init}) solves a root-finding problem $f(q) = y$ numerically using q_{init} as the initial point. If it fails, it returns nil. It is assumed that the result q lies close to q_{init}.
- SampleF(y) first samples a random configuration $q_{rand} \in \mathcal{C}$ and then uses Solve(y, q_{rand}). If the result is nil or infeasible, then nil is returned.
- Visible(y, t_s, t_g, q_s, q_g) is an incomplete, deterministic method for local path resolution of a path $y(t)$ over the interval $[t_s, t_g]$. We require that $f(q_s) = y(t_s)$ and $f(q_g) = y(t_g)$ be given as the endpoints of the interval. Pseudocode for Visible is given in Alg. 1, and is similar to the method of [14] except with collision handling.

Here $0.5 < c < 1$ is a parameter that controls the maximum amount of drift away from a straight line path. Without Line 5, the bisected path could grow without bound. Usually c is set close to 1. Examples in this paper use 0.9.

Using these primitives, we present a probabilistically complete path resolution method using a slightly modified probabilistic roadmap (PRM) algorithm.

Algorithm 1 Visible(y, t_s, t_g, q_s, q_g)

if $d(q_s, q_g) \leq \epsilon$ **then return** "true"
Let $y_m \leftarrow y((t_s + t_g)/2)$ and $q_m \leftarrow (q_s + q_g)/2$
Let $q \leftarrow Solve(y_m, q_m)$
if $q = nil$ or $q \notin \mathcal{F}$ **then return** "false"
if $max(d(q, q_s), d(q, q_g)) > c \cdot d(q_s, q_g)$ **then return** "false"
if Visible(y, t_s, t_m, q_s, q_m) and Visible(y, t_m, t_g, q_m, q_g) **then return** "true"
return "false"

Algorithm 2 PRM-Path-Resolution(y, N)

Initialize empty roadmap $\mathcal{R} = (V, E)$
if $q(0)$ and $q(1)$ are given **then**
    Add $(0, q(0))$ and $(1, q(1))$ to V
else
    Sample $O(N)$ start configurations using SampleF$(y(0))$
    Sample $O(N)$ goal configurations using SampleF$(y(1))$
for $i = 1, ..., N$ **do**
    Sample $t_{sample} \sim U([0, 1])$
    Sample $q_{sample} \leftarrow$SampleF$(y(t_{sample})$
    if $q_{sample} \neq nil$ **then** add (t_{sample}, q) to V
for all nearby pairs of vertices $(t_u, q_u), (t_v, q_v)$ with $t_u < t_v$ **do**
    if Visible(y, t_u, t_v, q_u, q_v) **then**
        Add the (directed) edge to E
Search $\mathcal{R}$ for a path from $t = 0$ to $t = 1$

Here, the PRM is built in the space $[0, 1] \times \mathcal{F}$ of time-configuration pairs (t, q), subject to the manifold constraint $f(q) = y(t)$. The two necessary modifications to PRM are 1) maintaining the manifold constraint, and 2) restricting forward progress along the time domain by constructing a directed graph. Pseudocode is given in Alg. 2.

5 Approximate global redundancy resolution

Although planning paths on constraint manifolds has been well studied [1, 11], an algorithm for global resolution must, essentially, plan C-space motions for all workspace paths. We assume the workspace is discretized into a network of workspace paths, whose nodes are sufficiently dense to interpolate behavior across the entire workspace via standard function approximation techniques (Sec. 6). This section describes two methods for generating the resolution, one local method and one global sampling-based method. This latter method takes inspiration from Alg.2 in that we build C-space roadmaps, except the roadmap is built along all workspace paths, and rather than finding one path we seek a connected "sheet" that spans the projection to workspace.

Let $G_W = (V_W, E_W)$ be a workspace roadmap. This can be generated either in the form of a probabilistic roadmap via random sampling, or as a grid. The solution we seek is a mapping from the vertices to C-space $g : V_W \to \mathcal{F}$ such that for any two adjacent workspace points $(y, y') \in E_W$, the straight line path $\overline{yy'}$ is locally pathwise resolvable between $g[y]$ and $g[y']$. For notational convenience, let the local reachability indicator function $R(y, y', q, q')$ be 1 if Visible$(\overline{yy'}, 0, 1, q, q')$ yields "success" and 0 otherwise.

In the case that g is not a resolution, we propose the use of the following primary error metric that measures the number of unresolved edges:

$$U(g) = |E_W| - \sum_{(y,y') \in E_W} R(y, y', g[y], g[y']) \tag{1}$$

If the problem is not resolvable, we wish to find a g that minimizes $U(g)$.

Note that for a globally resolvable problem, there are also an infinite number of pseudoinverses, some of which are smoother than others. It is then a desirable secondary objective to maximize smoothness in the redundant dimensions. Distance is a good proxy for smoothness, so for a secondary error metric we use an edge cost metric that measures total C-space path length:

$$L(g) = \sum_{(y,y') \in E_W} d(g[y], g[y'])R(y, y', g[y], g[y']). \tag{2}$$

In each of our algorithms, we will generate a configuration roadmap $\mathcal{R}_C = (V_C, E_C)$ whose vertices and edges map (surjectively) to corresponding vertices and edges of G_W. The connection is given by $Y[q]$, which maps C-space vertices to associated workspace vertices, and its inverse $Q[y]$, which map workspace vertices to a set (possibly empty) of associated C-space vertices. We will maintain:

$$E_C = \{(q, q') \quad | \quad (Y[q], Y[q']) \in E_W \text{ and } R(Y[q], Y[q'], q, q') = 1\}. \tag{3}$$

Each algorithm has a different method of sampling $\mathcal{R}_C$ and selecting the elements of the map $g[y] \in Q[y]$.

5.1 Pointwise global resolution

We first present a fast and simple pointwise method that locally propagates pointwise solutions across the workspace roadmap. Here each set $Q[y]$ contains at most 1 configuration, and hence the extraction of g is straightforward. Pseudocode is given in Alg. 3.

Here $N(y)$ is the neighborhood of a vertex y in the workspace graph. Note that the bookkeeping associated with maintaining the associations in Y and Q is fairly straightforward, but since it is rather tedious to write it will be left implicit in the remainder of the pseudocode. Specifically, in lines 6 and 7, we implicitly assume that the maps Y and Q will be updated appropriately: $Y[q] \leftarrow y$ and $Q[y] \leftarrow Q[y] \cup \{q\}$.

Algorithm 3 Pointwise-Global-Resolution(G_W, N_q)

Initialize empty roadmap $\mathcal{R}_C = (V_C, E_C)$
for each $y \in V_W$ **do**
    Let $Q_{seed} \leftarrow \cup_{w \in N(y)} Q[w]$
    for each $q_s \in Q_{seed}$ **do**
        Run $q \leftarrow$Solve(y, q_s)
        if $q \neq nil$ **then** add q to V_C and go to Step 2, proceeding to the next y.
    Run SampleF(y) up to N_q times. If any sample q succeeds, add it to V_C.
for all edges $(y, y') \in E_W$ such that $|Q(y)| > 0$ and $|Q(y')| > 0$ **do**
    Let q be the only member of $Q(y)$ and q' the only member of $Q(y')$
    if R(y, y', q, q')=1 **then**
        Add (q, q') to E_C
return $\mathcal{R}_C$

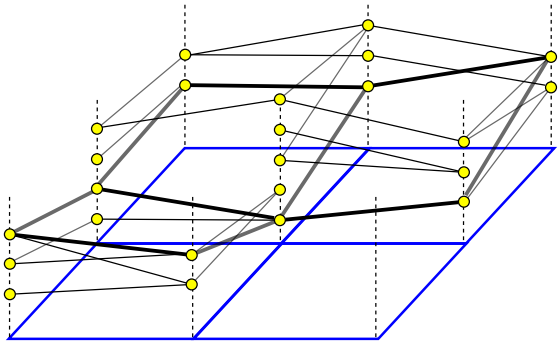

Fig. 4. Illustrating global resolution on a small problem. Each point on a workspace roadmap (3x3 grid on horizontal plane) corresponds to a certain self-motion manifold (vertical dashed lines). Feasible configuration space samples (circles) are sampled in each manifold, and edges are tested for feasibility along all workspace edges. Global resolution seeks to select a single configuration for each workspace point to form a maximally connected "sheet" in the configuration space roadmap.

5.2 Constraint satisfaction-based resolution

The pointwise method can yield poor results, leaving several edges unnecessarily unresolved. To improve performance, we present the following algorithm that samples many configurations in the preimage of each workspace point, connects them with feasible edges, and seeks a "sheet" in the C-space roadmap (Fig. 4). In order to perform this latter step, redundancy resolution is cast as a constraint satisfaction problem (CSP).

The algorithm proceeds by sampling many configurations per workspace point and adding them to $Q[y]$. Each pair of configurations along workspace edge is then tested for visibility and added to E_C. This leads to the formulation of a MAX-CSP in which a value $g[y] \in Q[y]$ is sought for each workspace point, such that for all neighboring points y', the C-space edge is visible, that

is, $(g[y], g[y']) \in E_C$. If there is no such resolution, then MAX-CSP seeks an assignment g that minimizes the number of conflicts $U(g)$. We also maintain $L(g)$ as a secondary optimization criterion.

Since we typically have thousands of workspace points and often dozens or hundreds of configurations in each point's domain, backtracking CSP methods are usually prohibitively expensive, especially when the problem is not resolvable. For example, the open source Gecode library exhausted our machine's RAM before finding even a suboptimal solution on the smallest problems in our experiments [12]. The Sugar algorithm, which converts CSPs to boolean satisfaction (SAT) problems, produced intermediate SAT problems exceeding the limit of 4 Gb on even smaller problems with 500 workspace nodes [13]. As a result we employ a heuristic local search method. The algorithm proceeds in three phases.

Heuristic descent. Our search phase proceeds down a single path of a backtracking search in the dual graph CSP, and stops when no more workspace edges can be assigned. In this formulation, the edges (y, y') are the variables, values are the $g[y]$, $g[y']$ pairs at their endpoints, and the domains are the set of all edges $(q, q') \in E_C$ such that $Y[q] = y$ and $Y[q'] = y'$. Pairs of edges that meet at the same workspace point are constrained so a configuration at one endpoint must match the value of the adjacent edge. The advantage of this formulation is that maximizing the number of values assigned leads to a direct minimization of $U(q)$. Forward checking and domain consistency are used for constraint propagation, and variable ordering heuristics of most constrained variable and most constraining variable are applied. Value selection is done using the least-constraining value heuristic. ties are broken randomly. No backtracking is performed, and as many non-conflicting variables are assigned as possible.

Min-conflicts. The second phase does a more straightforward min-conflicts operation, this time on the primal graph where variables are nodes, values are assigned configurations $g[y]$, and domains are sampled configurations $Q[y]$. Here, the algorithm iterates from the assignment produced by phase 1 by choosing a conflicting variable at random, if one exists. If none exists, then we are done. The value in its domain that minimizes conflicts with its neighbors is then selected. This process repeats some number of iterations.

Length optimization via coordinate descent. At this point, the number of conflicts is usually close to a minimum. The final phase then attempts to optimize $L(g)$ while keeping $U(g)$ constant. Here, a coordinate descent algorithm is used. For each C-space node $g[y]$, we compute the average configuration q_{avg} of its neighbors in $\mathcal{R}_C$ amongst those whose endpoints match in g. It is then used as a target for optimization $\min_q d(q, q_{avg})$, under the constraint that $f(q) = y$ must be satisfied and that reachability of q from all neighbors is satisfied. To do this, we use bisection. First, $q \leftarrow Solve(y, q_{avg})$ is calculated. If q is not reachable from its neighbors, then q_{avg} is set to the midpoint of the line segment from $g[y]$ to q_{avg}. Solving and bisection repeats until a reachable configuration is found.

The overall algorithm is listed in Alg. 4. Experiments suggest that all three of these phases cooperate to produce high-quality results. Skipping the heuristic descent step leads to poor performance for more complex problems, because

Algorithm 4 CSP-Global-Resolution($G_W, N_q, N_{mc}, N_{opt}$)

```
1: Initialize R_C ← Pointwise-Global-Resolution(G_W, N_q)
2: for each y ∈ V_W do
3:     Run SampleF(y) N_q times. Add all samples that succeed to V_C.
4: for all edges (y, y') ∈ E_W such that |Q(y)| > 0 and |Q(y')| > 0 do
5:     for all (q, q') ∈ Q(y) × Q(y') do
6:         if R(y, y', q, q') = 1 then
7:             Add (q, q') to E_C
8: g ← HeuristicDescentCSP(R_C)
9: g ← MinConflictsCSP(R_C, g, N_mc)
10: CoordinateDescent(R_C, g, N_opt)
11: return g
```

pointwise assignment leaves the solution in a deep local minimum. Skipping min-conflicts fails to clean up some errors in heuristic descent, and skipping length optimization leads to maps that are much less smooth.

The parameter values are as follows:

- N_q: the number of C-space samples drawn per workspace node. More samples are typically needed for highly redundant systems. We use 50–100 in our experiments.
- N_{mc}: the number of min-conflicts iterations. A small number of passes, such as 10× the number of conflicts, usually works well.
- N_{opt}: the number of coordinate descent iterations. We have observed most problems converging to less than 0.1% change in objective function value in about 20 iterations.

5.3 Performance considerations

In our experiments, the algorithms above range in computation time from minutes to hours for thousands of workspace points. The limiting step is C-space roadmap construction (Steps 2–7), and visibility checking in particular (Step 6), which is significantly more expensive than any other primitive operation.

Scalability-wise, the pointwise assignment algorithm performs $O(N_q|V_W|)$ C-space samples (although in practice this is closer to $O(|V_W|)$), and $O(|E_W|)$ visibility checks. The CSP algorithm performs $O(N_q|V_W|)$ configuration samples and $O(N_q^2|E_W|)$ visibility checks in Lines 2–7. With efficient implementation of CSP updates and a priority queue for sorting edges, heuristic descent runs in time $O(d_W N_q^2|E_W| \log |E_W|)$, where d_W is the degree of G_W. The min-conflicts operation finds initial conflicts in $O(d_W N_q^2|V_W|)$ and each of the N_{mc} iterations takes time $O(d_W N_q^2)$ time. Finally, optimization takes $O(N_{opt}|V_W|)$ optimization steps and $O(N_{opt}|E_W|)$ visibility checks.

The other question is whether the algorithm produces an optimal or near-optimal resolution? There are two factors influencing this: whether the C-space roadmap contains a solution, and whether the constraint satisfaction solver finds

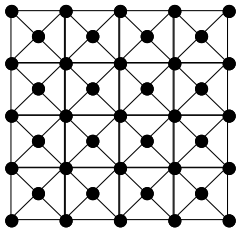

Fig. 5. Staggered grid network used to discretize the workspace.

it. For problems with narrow passages in free space, finer workspace and configuration roadmaps are needed to resolve them. As a result, it is expected that as $|G_W|$ and N_q grows, the probability that $\mathcal{R}_C$ contains a resolution will grow towards 1. In practice, it is often effective to employ a second sampling pass at the vertices of unresolved edges. As for the CSP solver, even though the algorithm performs well in practice, it is indeed a heuristic. Producing an efficient, optimal method seems challenging in light of the NP-hardness of CSPs and the poor performance of existing state-of-the-art backtracking solvers.

6 Interpolation between sample points

Once a resolution $g[y]$ over the workspace roadmap G_W is produced, we extend it to the continuous workspace $\mathcal{W}_C$ via function approximation. In particular, we yield $f^+(y) = Solve(g_{interp}(y), y)$ where $g_{interp}(y)$ is a weighted combination of resolved configurations at workspace points

$$g_{interp}(y) = \sum_i w_i(y)c_i(y)g[y_i] / \sum_j w_j(y)c_j(y) \tag{4}$$

and the sums are taken over indices of workspace points $y_i \in V_W$. The terms w_i are distance-based weights that decrease as y grows more distant from y_i, while the c_i terms are indicator functions that prevent interpolation across discontinuity boundaries in the resolution. For a regular grid, barycentric coordinates in the simplex are used as weights. If G_W is a probabilistic roadmap it is possible to use scattered data interpolation techniques.

The term c_i is 1 if the neighborhood of y is globally resolved, but if there is a discontinuity, it is used only to average points on one side of the boundary. To compute it, we take the subgraph of the C-space roadmap $\mathcal{R}_C$ corresponding to the resolved points $g[y_i]$ for all workspace points y_i with nonzero weights. We then compute connected components of this subgraph, and for all nodes in the largest connected component c_i is set to 1. It is set to 0 for all other nodes.

7 Results

In all of the following experiments the workspace was the Cartesian coordinates of a point on the robot's end effector in 2- or 3-dimensional space. A staggered

Table 1. Summary of experiments.

Robot	$dim(\mathcal{C})$	$dim(\mathcal{W})$	**Grid points**	**% disconnected**	**Distance ratio** (rad/m)
Planar 3R	3	2	2,000	1.42	0.28
Planar 20R	20	2	2,000	0.96	0.21
ATLAS	7	3	16,000	1.61	3.03
Baxter	7	3	8,000	3.92	1.95
Jaco	6	3	8,000	0.062	3.81
Robonaut2	6	3	8,000	4.50	3.12
Robosimian	7	3	4,000	13.9	2.92

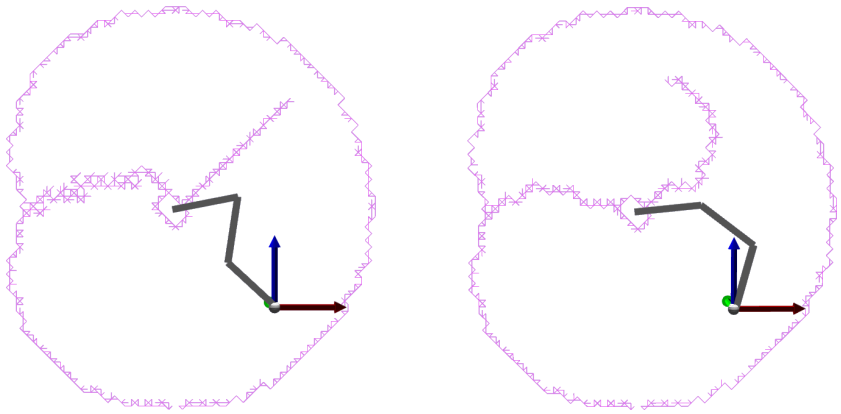

Fig. 6. Pseudoinverse discontinuity boundaries for a planar 3R arm. Left: pointwise resolution. Right: the optimized solution has slightly fewer discontinuities.

workspace grid was used in each example (Fig. 5). Statistics for these tests are given in Tab. 1. Most of these parameters are self-explanatory except for the final two columns, which give the percentage of disconnected workspace graph edges (% disconnected) and the overall ratio of configuration space distance to workspace distance summed along all edges on the roadmap (Distance ratio).

The first set of experiments are performed on planar kR robots. Fig. 6, left, shows the pointwise assignment boundaries for a 3R robot with joint limits of ± 2 rad and a workspace grid of 2,000 points. In this case, pointwise assignment works fairly well. Our method (Fig. 6, right) only reduces the number of disconnections by 24%, but reduces the total C-space path length by about 60%.

Results for a 20R robot with joint limits of ± 0.6 rad are shown in Fig. 7. In this case, the pointwise assignment causes a wide swath of disconnections in the upper part of the workspace (3.69% of reachable edges). Using the CSP method, the number of disconnections are greatly reduced (0.96% of reachable edges) and the total path length is reduced by 38%.

Our next set of experiments apply the method to robot arms with 6 or more DOFs in 3D Cartesian space. Arbitrary orientation of the end effector is permitted, and arm configurations are required to obey joint limits and avoid self-

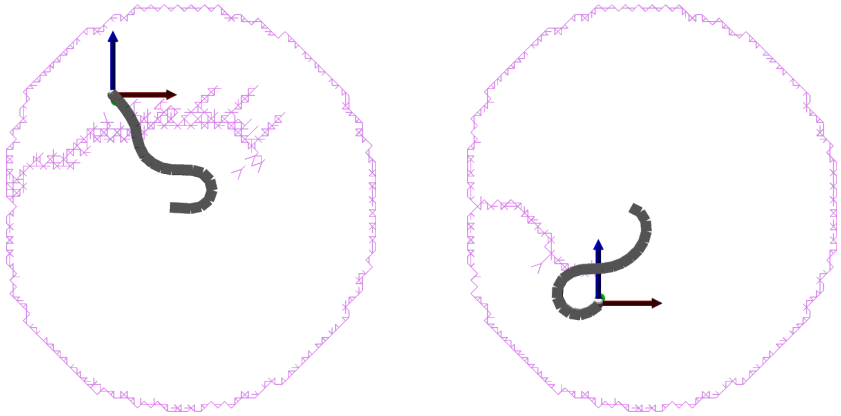

Fig. 7. A planar 20R arm. Left: pointwise resolution. Right: optimized solution.

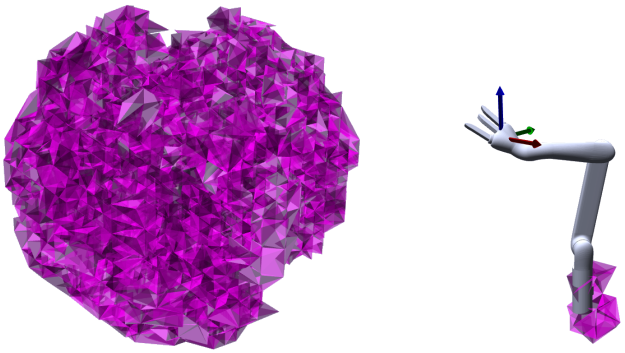

Fig. 8. Results for the Jaco arm. Left: after pointwise resolution, the discontinuity boundary spans nearly the entire workspace. Right: optimized solution is resolved almost everywhere. (For clarity, the outer workspace boundary is not drawn in these or any other 3D figures.)

collision. In each case, pointwise assignment does worse than in 2D. Fig. 8, left, shows results for a Kinova Jaco arm. The purple region illustrates the surface of disconnections. The IK problem is to move the center of the hand to a specific position, and the workspace grid has 8,000 points. A huge number of disconnections are caused by pointwise resolution. After CSP resolution (Fig. 8, right), the number of disconnections drops to a minimal number (0.062% of reachable edges). This favorable result is likely because the Jaco has three continuous rotation joints.

The next four figures show optimized results for single arms of the Boston Dynamics ATLAS (Fig. 9), the Rethink Robotics Baxter (Fig. 10), the NASA Robonaut2 (Fig. 11), and a leg of the JPL Robosimian (Fig. 12). None of these robots has continuous rotation joints. The kinematic structure and limits of each robot is different, with the ATLAS, Baxter, and Robonaut2 having an

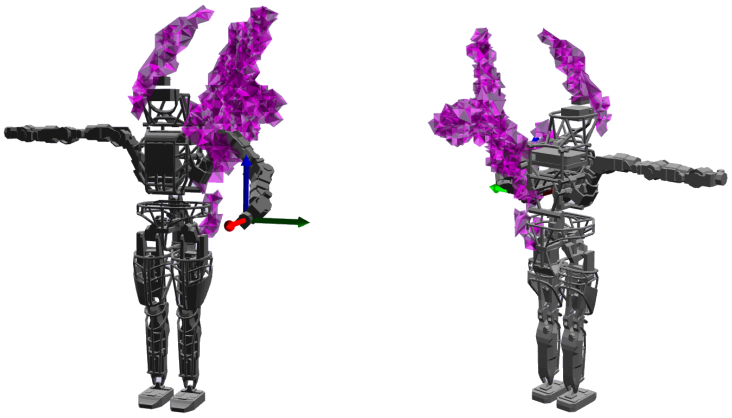

Fig. 9. Optimized results for the arm of the ATLAS humanoid. Two views of the discontinuity boundary are shown.

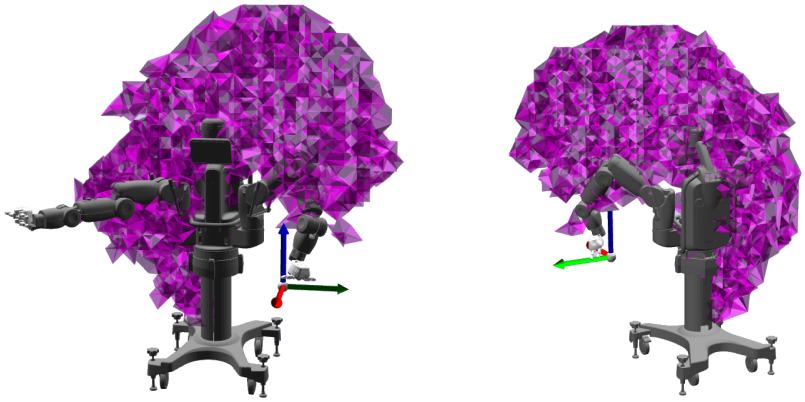

Fig. 10. Optimized results for the arm of the Baxter robot. Two views of the discontinuity boundary are shown.

anthropomorphic elbow and 3-axis shoulder, but with different axes and joint limits. The Robosimian's joints are arranged so that each pair of 2 joints have intersecting axes, and each joint can rotate 360°.

ATLAS has a discontinuous region near a singularity at the shoulder, as well as over the head where the upper arm comes close to colliding with the head. Baxter has a much larger region above the shoulder, likely because the arm naturally bends downward, and it approaches joint limits as the arm bends upward. As a result, to pass from front to back in that upward region, the hand must pass down and around. It also has discontinuities the opposite side of the torso near the rear, where the arm can reach points either around the front or around the back of the torso. Robonaut2 has essentially the opposite problem, with discontinuities along the underside of the arm. Joint limits in the shoulder mean that its hand must move up and around to pass from elbow-back to elbow-

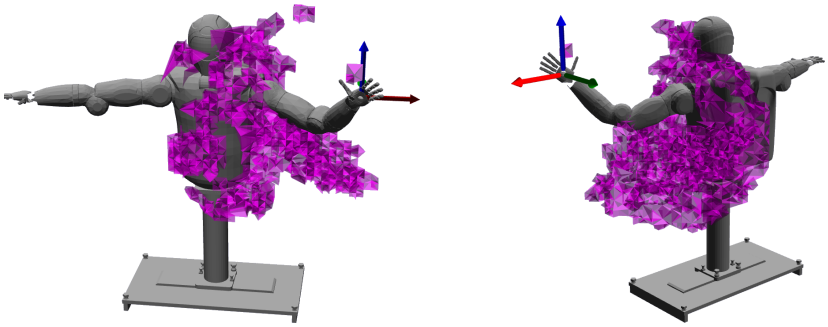

Fig. 11. Optimized results for the arm of the Robonaut2. Two views of the discontinuity boundary are shown.

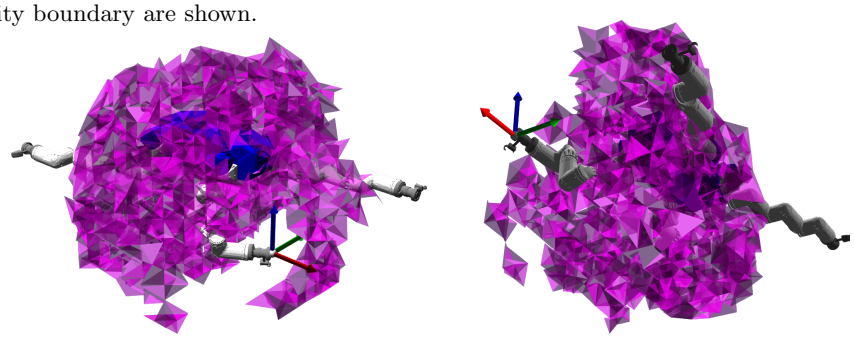

Fig. 12. Optimized results for one limb of the Robosimian quadruped. Two views of the discontinuity boundary are shown.

front configurations. Other small discontinuities are scattered about the torso, which are likely free space narrow passages caused by self-collision. Perhaps a denser sampling of configurations or workspace would help reduce these artifacts.

Robosimian is an interesting case. Although its joints have a larger range of motion than each of the four other robots, its kinematics are quite different. Its workspace is quite large, as each limb can reach completely around the body in all directions. Although it is hard to see from the photos, most of the discontinuous boundaries lie close to edge of the workspace or near the robot's body, and a large cavity of resolved space lies below the robot. Unlike the manipulator arms, it is unable to rotate a "shoulder" about an arbitrary axis, and hence it needs to perform rather large joint space motions to reconfigure itself to move the end effector when close to singularities. Also, when close to self-collision, many end effector movements are likely to cause links near the body to come into collision.

8 Conclusion

This paper defined and presented an approximate method for computing maximally continuous pseudoinverses of multivariate functions, with the application of global redundancy resolution for Cartesian workspace movements of robot

manipulators. The resulting maps may be useful for robot mechanism design, teleoperation, or reduced dimensionality motion planning. Performance of the CSP solver might be improved via connectivity analysis of self-motion manifolds to reduce the size of the constraint graph. Future work should also study application of global resolution to higher dimensional workspaces.

Acknowledgment

This work is partially supported by NSF CAREER Award #1253553 and NSF NRI #1527826.

References

1. D. Berenson, S. S. Srinivasa, D. Ferguson, and J. J. Kuffner. Manipulation planning on constraint manifolds. In *IEEE Int. Conf. Rob. Aut.*, 2009.
2. K. Byl, M. Byl, and B. Satzinger. Algorithmic optimization of inverse kinematics tables for high degree-of-freedom limbs. In *Proc. ASME 2014 Dynamic Systems and Control Conference*, Oct. 2014.
3. P. Freeman. *Minimum Jerk Trajectory Planning for Trajectory Constrained Redundant Robots*. PhD thesis, Washington University of Saint Louis, 2011.
4. M. Goel, A. A. Maciejewski, V. Balakrishnan, and R. W. Proctor. Failure tolerant teleoperation of a kinematically redundant manipulator: an experimental study. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 33(6):758–765, Nov 2003.
5. C. L. Lück. Self-motion representation and global path planning optimization for redundant manipulators through topology-based discretization. *J. Intelligent and Robotic Systems*, 19:23–28, 1997.
6. R. V. Mayorga and A. K. C. Wong. A global approach for the optimal path generation of redundant robot manipulators. *J. Robotics Systems*, 7(1):107–128, 1990.
7. Y. Nakamura. *Advanced Robotics: Redundancy and Optimization*. Addison-Wesley, New York, 1991.
8. G. Oriolo and C. Mongillo. Motion planning for mobile manipulators along given end-effector paths. In *IEEE Int. Conf. Rob. Aut.*, 2005.
9. B. Satzinger, J. I. Reid, M. Bajracharya, P. Hebert, and K. Byl. More solutions means more problems: Resolving kinematic redundancy in robot locomotion on complex terrain. In *IEEE/RSJ Int. Conf. Intel. Rob. Sys.*, 2014.
10. S. Seereeram and J. T. Wen. A global approach to path planning for redundant manipulators. *IEEE T. Robotics and Automation*, 11(1):152–160, Feb. 1995.
11. M. Stilman. Global manipulation planning in robot joint space with task constraints. *IEEE Trans. Robotics*, 26:576–584, 2010.
12. G. Tack. *Constraint Propagation - Models, Techniques, Implementation*. PhD thesis, Saarland University, Germany, 2009.
13. N. Tamura, A. Taga, S. Kitagawa, and M. Banbara. Compiling finite linear csp into sat. *Constraints*, 14(2):254–272, June 2009.
14. R. H. Taylor. Planning and execution of straight line manipulator trajectories. *IBM J. Res. Dev.*, 23(4):424–436, July 1979.
15. F. Zacharias, C. Borst, M. Beetz, and G. Hirzinger. Positioning mobile manipulators to perform constrained linear trajectories. In *IEEE/RSJ Int. Conf. Intel. Rob. Sys.*, 2008.