Global Redundancy Resolution via Continuous Pseudoinversion of the Forward Kinematic Map

Kris Hauser, *Member, IEEE,* and Scott Emmons

***Abstract*—This paper presents a novel approach to kinematic redundancy resolution for redundant robots, which have more degrees of freedom than workspace dimensions. It introduces the concept of 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. The problem is cast as a continuous pseudoinversion of the forward kinematic map. 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 constructs roadmaps of both the domain and image, and minimizes discontinuities of the inverse function using a MAX-SAT constraint satisfaction problem. Applications of this map include teleoperation, dimensionality reduction in motion planning, and workspace visualization. Results are demonstrated on toy problems with up to 20 DOF, and on several robot arms.**

Note to Practitioners: ***Abstract*—Determining whether a robot manipulator can cover a range of movement in a Cartesian workspace under joint limits and collision constraints is typically addressed by an engineer's intuition and trial-and-error. This paper presents an algorithm to solve this problem systematically. The method optimizes a mapping from workspace to joint space to minimize the number of discontinuities. The resulting maps can be used to select continuous inverse kinematic solutions to follow workspace paths, and their visualizations can aid in workcell design, robot selection, and robot placement.**

Primary and Secondary Keywords ***Index Terms*—Primary Topics: Robot kinematics, Inverse problems. Secondary Topic Keywords: Manufacturing automation, Topology.**

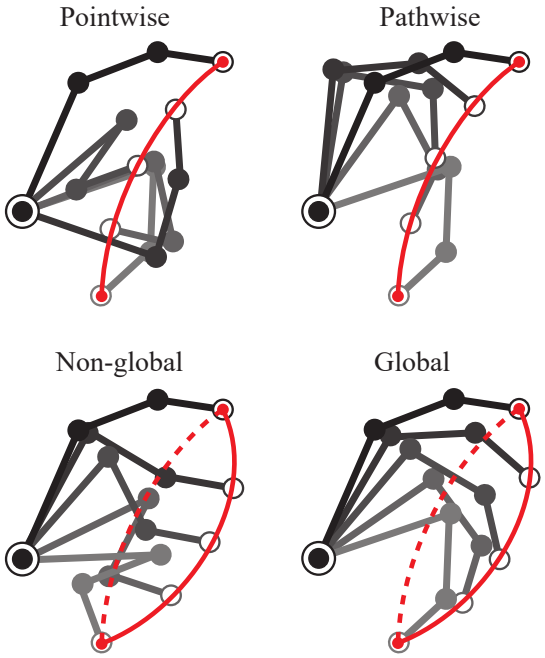


Fig. 1. Illustrating three types of redundancy resolution. Pointwise resolution gives a solution for each point, but does not guarantee continuity along a path. Pathwise resolution gives a continuous solution along a workspace path, but is *non-global*: the robot may return to the starting point with 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.

I. Introduction

REDUNDANCY resolution is the problem of determining joint-space motions that achieve workspace (Cartesian) motions for robots that have more degrees of freedom than the workspace. *Pointwise* methods determine inverse kinematics (IK) solutions using only local knowledge of the workspace motion, and suffer from the problem that the robot may get stuck at joint limits or at self collision [6]. *Pathwise* redundancy resolution seeks to generate a continuous configuration-space path for a given workspace path. This has the potential disadvantage of being unpredictable, in that the success of resolution may depend on the initial configuration, and a closed workspace cycle may not return the robot to the same configuration [13]. This paper formulates the notion of a *global* redundancy resolution, which 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 (Fig. 1).

The global redundancy resolution problem may be generalized to one of continuous multivariate pseudoinversion. Function inversion is frequently encountered in many fields of science and engineering, including robotics, computer graphics, control, and mechanical design. Pointwise inversion techniques, like IK, seek a point x such that $f(x) = y$ for a given 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 an algorithm to obtain maximally continuous and smooth inverse functions in high-dimensional spaces.

For example, consider the IK problem for a 2R planar robot manipulator restricted by joint limits (Fig. 2). Here the forward kinematics map f maps joint angles to Cartesian end effector

K. Hauser is with the Department of Electrical and Computer Engineering and Department of Mechanical Engineering and Materials Science, Duke University, Durham, NC, 27708 USA e-mail: kris.hauser@duke.edu.

S. Emmons is with University of North Carolina at Chapel Hill and Duke University e-mail: scott.emmons@duke.edu

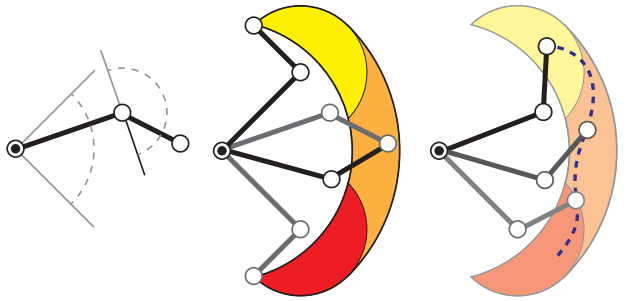


Fig. 2. A planar 2R manipulator with joint limits of $\pm 45^\circ$ and $\pm 90^\circ$, respectively. There are no joint space resolutions of Cartesian end effector paths (e.g., the dotted curve) in the interior of the reachable workspace from the upper region to the lower region.

(workspace) points. 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 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 a 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.

We present an approximate global redundancy 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-CSP) is applied to generate a pseudo-inverse 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.

II. Related work

Redunancy resolution has been studied extensively, with many pointwise heuristics proposed to avoid singularities, joint limits, and obstacles [9], [12]. Several researchers have identified the problem of local minima of pointwise resolution methods, leading to pathwise resolution techniques [5], [9], [11], [13], [14], [16]. The pathwise algorithm we present in Sec. IV-A is most closely related to the work of [14], who use a randomized tree-growing approach to explore self-motion manifolds along a discretized workspace path. Recently, a sampling-based pathwise redundancy planner was developed that enforces joint space paths to be cyclic [13]

A related problem is that of path planning for closed-chain mechanisms. One class of planners analyzes the shape and connectivity of the closed-chain configuration space (C-space) manifold, with exact planners developed for robots with spherical joints [23] and planar star-shaped robots [17]. Although such algorithms provide global and exact connectivity of the manifold, they are limited to restrictive problem classes and have not been extended to handle joint limit and collision constraints in 3D. To address these shortcomings, sampling-based motion planners [1], [4], [7], [19] have been applied to this problem as well. These planners construct roadmaps whose nodes are collision-free configurations sampled at random, and edges are collision-free paths between them. To sample closed-chain configurations, open-chain configurations are first sampled at random, and then loop-closure constraints are enforced. Local paths are checked by interpolating along the constraint manifold using local optimization.

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], [10]. The method of [10] 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 [15]. We use a similar optimization method, although in this case, using coordinate descent, to smooth our maps in postprocessing as described in Sec. IV-D.

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 [24]. Qualitatively, we observe these discontinuities correspond to workspace regions that are frustrating for teleoperation [6]. Finally, these mappings may also be useful for reduced-dimensionality motion planning in Cartesian space [15].

This paper is an extended version of a conference manuscript [8]. This version describes related work and the algorithm in greater detail; presents an enhanced visualization technique; and presents new work in scaling to 6D workspaces. New experiments are conducted to illustrate the potential for applying the technique to workspace design problems on the Baxter robot and a model of the Staubli TX90L industrial manipulator.

III. Problem definition

We wish to compute a continuous pseudoinverse of a robot's forward kinematics map $f : \mathcal{C} \to \mathcal{W}$. Here, $\mathcal{C}$ is a configuration space (C-space) consisting of the robot's degrees of freedom. $\mathcal{W}$ is the Cartesian workspace, which typically contains either

position or orientation of the robot's 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 $\mathcal{W}_C = f[\mathcal{F}]$.

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 $f[A]$.

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 resolvability $\implies$ pathwise resolvability $\implies$ pointwise resolvability. As discussed in the introduction, the converse does not necessarily hold true.

IV. Approximate Global Redundancy Resolution

A global redundancy resolution must, essentially, resolve all workspace paths *simultaneously*. Our approximation assumes the workspace is discretized into a network of workspace paths, and resolves all paths simultaneously across the network. We first describe a probabilistically-complete sampling-based algorithm to solve the simpler problem of pathwise redundancy resolution, illustrated in Fig. 3. For global resolution, we also 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 workspace dimensions.

A. Pathwise Redundancy Resolution

Consider a parameterized, continuous workspace path. 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). Hence, 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.

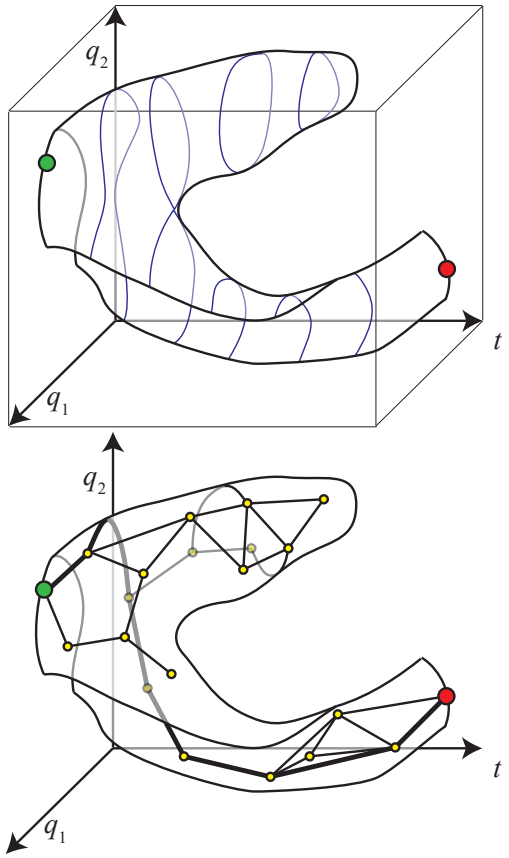


Fig. 3. Top: Illustrating the pathwise redundancy resolution problem. Bottom: a probabilistically complete roadmap-based solution.

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 [22] 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 com-

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

1: **if** $d(q_s, q_g) \leq \epsilon$ **then return** "true"
2: Let $y_m \leftarrow y((t_s + t_g)/2)$ and $q_m \leftarrow (q_s + q_g)/2$
3: Let $q \leftarrow Solve(y_m, q_m)$
4: **if** $q = nil$ or $q \notin \mathcal{F}$ **then return** "false"
5: **if** $max(d(q, q_s), d(q, q_g)) > c \cdot d(q_s, q_g)$ **then return** "false"
6: **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"
7: **return** "false"

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

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

plete path resolution method using a slightly modified probabilistic roadmap (PRM) algorithm. 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.

B. Workspace and C-Space Roadmaps

The first step in our global redundancy resolution technique is to define a workspace roadmap $G_W = (V_W, E_W)$. This can be generated either in the form of a grid or a probabilistic roadmap. It is assumed the nodes in V_W are sufficiently dense to interpolate behavior across the entire workspace via standard function approximation techniques (Sec. IV-F). The solution we seek is a mapping from the vertices to C-space $g : V_W \rightarrow \mathcal{F}$ 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 following primary cost function that measures the number of unresolved edges:

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

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

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

If the problem is not resolvable, we wish to find a g that minimizes $U(g)$. It may also be possible to weight edges by their importance to the task, such as by assigning higher cost to regions of interest in the workspace.

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 cost we measure total C-space path length:

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

Our algorithms 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. Given Y, we define:

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

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

C. Pointwise global resolution

For comparison, we describe 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 7 and 9, we implicitly assume that the maps Y and Q will be updated appropriately: $Y[q] \leftarrow y$ and $Q[y] \leftarrow Q[y] \cup \{q\}$.

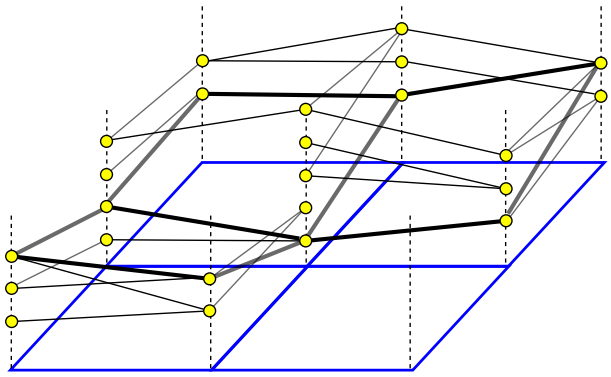


Fig. 4. Illustrating global resolution on a small problem. Each point on a workspace roadmap (3x3 grid on horizontal plane) corresponds to a certain self-motion manifold (vertical dashed lines). Feasible configuration space samples (circles) are sampled in each manifold, and edges are tested for feasibility 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.

D. Constraint satisfaction-based resolution

The pointwise method often leaves many edges unnecessarily unresolved. To improve performance, the following algorithm 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 [20]. 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 [21]. 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.

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

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

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 heuristic descent 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. After conflicts are minimized, the final phase attempts to optimize $L(g)$ while keeping $U(g)$ constant. Coordinate descent is used as follows. 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 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 $10\times$ 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.

E. 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 2–8. 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 $O(d_W N_q^2)$ 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 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.

F. 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

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

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.

G. Computing the Discontinuity Boundary

A representation of the optimized discontinuity boundary is useful to retrieve for visualization and planning. In a d-dimensional workspace, the boundary is a $d-1$-dimensional non-manifold surface. We represent it as a a mesh of line segments / triangles in 2D / 3D spaces, respectively. Let $E_D \subseteq E_W$ denote the set of conflicting workspace edges that have feasible assigned configurations at each endpoint, but do not have a feasible assigned configuration-space path. To visualize the boundary of the reachable workspace we can also include edges from reachable vertices to unreachable ones in E_D. We compute a mesh that intersects all edges in E_D and no edge in $E_W \setminus E_D$.

To do so we first compute an edge-conforming simplicial decomposition of G_W. Then, for each simplex s in the decomposition and conflicting edge e on the simplex boundary, we construct a small portion of the boundary connecting e to the centroid s. Let C be the centroid and V be the midpoint of e. For 2D workspaces, we simply construct a line segment $\overline{VC}$. For 3D workspaces, we construct two triangles, VCC' and VCC'' where C' and C'' are the centroids of the simplices adjoining s.

We then proceed to simplify the mesh to yield less jagged boundaries when multiple conflict edges are adjacent. The vertices of the mesh that correspond to simplex centroids or face boundaries are candidates for collapsing. In 2D, if a centroid is connected to two adjacent conflict edges, it is collapsed to one of the edges. In 3D, if a centroid connects to three edges, all of its incident triangles are collapsed to a single face. If it connects two that border a single face, the centroid is collapsed to a neighboring centroid.

Finally, the boundary mesh is smoothed to minimize curvature, constrained so that each vertex along an edge stays on the edge and is reachable from both endpoints of the edge, if possible. To do so, for each edge $\overline{PQ} \in E_D$ we compute the maximum interpolation parameter b such that $(1-b)P + bQ$ is reachable with a continuous motion from P. We also compute the minimum parameter a such that $(1-a)P + aQ$ is reachable from Q. Then the parameter defining the mesh vertex V on e is constrained to the range $[a, b]$. If $a > b$, then $V = (1-u)P + uQ$ is set to the midpoint of the discontinuity with $u = (a+b)/2$.

TABLE I
SUMMARY OF EXPERIMENTS IN SECTION V.

Robot	$dim(\mathcal{C})$	$dim(\mathcal{W})$	**Grid points**	**% disconnected**	**Distance ratio** (rad/m)
Planar 3R	3	2	1,985	1.42	0.28
Planar 20R	20	2	1,985	0.96	0.21
Jaco	6	3	7,471	0.062	3.81
ATLAS	7	3	8,778	1.61	3.03
Baxter	7	3	7,471	3.92	1.95
Robonaut2	6	3	8,778	4.50	3.12
Robosimian	7	3	5,029	13.9	2.92

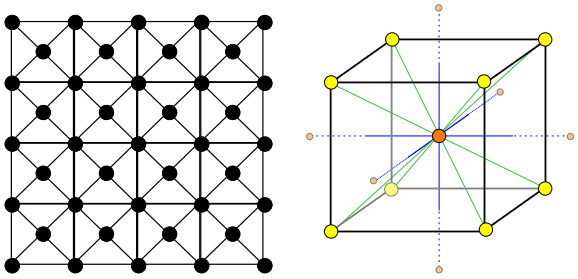


Fig. 5. Left: Staggered grid network used to discretize 2D workspaces. Right: staggered grid unit used in 3D.

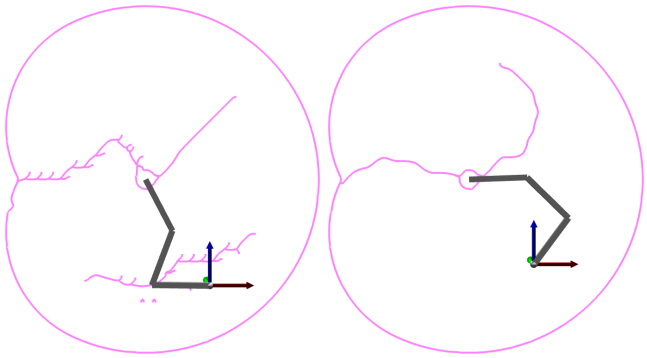


Fig. 6. Pseudoinverse discontinuity boundaries for a planar 3R arm. Left: pointwise resolution. Right: the optimized solution has fewer spurious discontinuities.

V. RESULTS

In all of the experiments in this section, the workspace consists of the Cartesian coordinates of a point on the robot's end effector in 2- or 3-dimensional space. Staggered workspace grids are used (Fig. 5). A summary of these tests is given in Tab. I. The final two columns 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). Low numbers are preferred to ensure a robot can be operated in Cartesian mode with minimal obstructions. However, it must be noted that these numbers assume uniform utility at all points in the workspace. To assess whether a robot is suitable for a given task, it is usually more informative to examine the discontinuity boundary visually as shown in the figures below, or to define custom workspace regions as we do in Sec. VI.

The first set of experiments are performed on planar kR robots. Fig. 6, left, shows the pointwise assignment boundaries

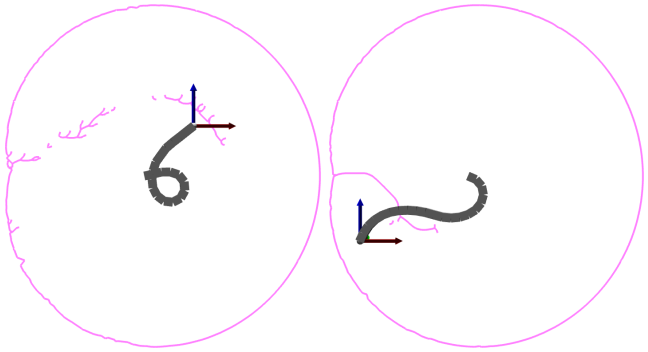


Fig. 7. A planar 20R arm. Left: pointwise resolution. Right: optimized solution.

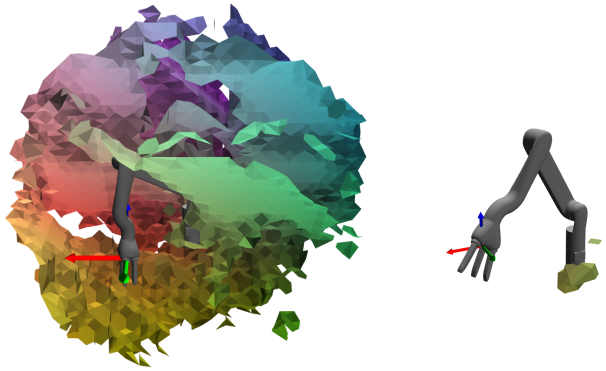
for a 3R robot with joint limits of ± 2 rad and a workspace grid of 2,000 points. In this case, pointwise assignment works fairly well. Our method (Fig. 6, right) only reduces the number of disconnections by 24%, but reduces the total C-space path length by about 60%.

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%.

The next set of experiments apply to robot arms with 6 or more DOFs in 3D Cartesian space, including the Kinova Jaco (Fig. 8), single arms of the Boston Dynamics ATLAS (Fig. 9), the Rethink Robotics Baxter (Fig. 10), the NASA Robonaut2 (Fig. 11), and a limb of the JPL Robosimian (Fig. 12). The kinematic structure and limits of each robot differs: the Jaco has three continuous rotation joints, and the ATLAS, Baxter, and Robonaut2 have an anthropomorphic elbow and 3-axis shoulder, but with different axes and joint limits. In the Robosimian limb, each pair of 2 joints have intersecting axes, and each joint can rotate 360°. Arbitrary orientation of the end effector is permitted, and configurations are required to obey joint limits and avoid self-collision.

Pointwise assignment does badly in 3D, as illustrated on the Jaco (Fig. 8.a). The purple region illustrates the surface of disconnections. A huge number of disconnections are caused by pointwise resolution. After CSP resolution (Fig. 8.b), the number of disconnections drops to a minimal number (0.062% of reachable edges). A rationale for this favorable result is that the Jaco' continuous rotation joints gives it significant flexibility to reorient its configuration across its workspace.

ATLAS has a discontinuous region near a singularity at the



(a) (b)

Fig. 8. Results for the Jaco arm. (a) After pointwise resolution, the discontinuity boundary divides up nearly the entire workspace. (b) The optimized solution is resolved almost everywhere. For clarity, the outer workspace boundary is not drawn in any 3D figures except when noted. Boundaries are color-coded and shaded by directional light. (Best viewed in color)

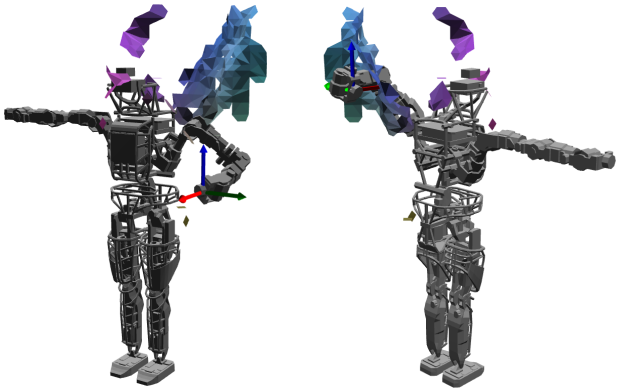


Fig. 9. Optimized results for the arm of the ATLAS humanoid. Two views of the discontinuity boundary are shown.

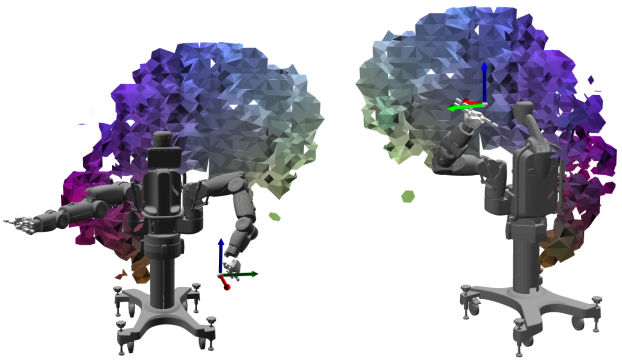


Fig. 10. Optimized results for the left arm of the Baxter robot, with free rotation. Two views of the discontinuity boundary are shown.

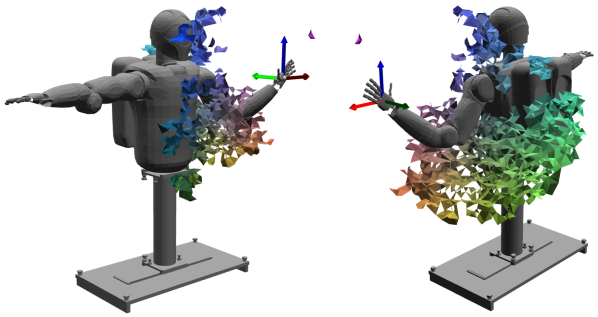


Fig. 11. Optimized results for the arm of the Robonaut2. Two views of the discontinuity boundary are shown.

shoulder, as well as over the head where the upper arm comes close to colliding with the head. Baxter has a much larger discontinuous region above the shoulder, likely because the arm naturally bends downward, and it approaches joint limits as the arm bends upward. Hence, 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-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. 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. Most of the optimized discontinuities lie close to edge of the workspace or near the robot's body, and a large cavity of resolved space lies below the robot. When rotation is constrained to point downward, as when walking on flat ground, the reachable space is more limited.

VI. Application to Robot Workspace Design

The presented technique can be used for task-driven workspace design, in which we ask questions such as "which robot would be able to move workpieces from area X to area Y, where should workpieces be placed so a robot can reach them, and in which orientation should an end effector be mounted?" Using this algorithm as a design tool, multiple designs can be compared visually by placing the discontinuity boundary in a given workcell, or quantitatively by calculating the fraction of Cartesian space disconnected by the resolution. It is trivial to incorporate environmental collision constraints into the free space because self-collision is already being checked.

As a practical example, consider the Amazon Picking Challenge (APC) [3]. The APC is a recurring robotic warehouse manipulation competition in which a robot is to retrieve objects from a shelving unit and place them into an order box. The shelving unit contains several small bins, which have openings on a vertical face of the unit. Moreover, they are relatively

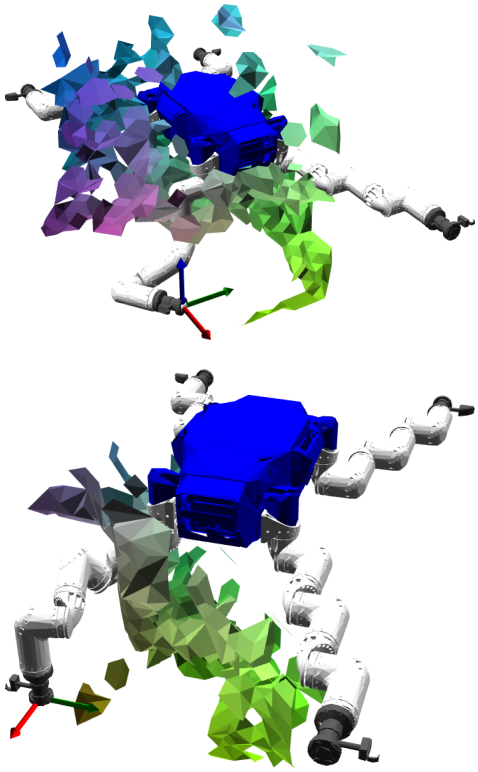


Fig. 12. Optimized results for one limb of the Robosimian. Top: rotation is unconstrained. Bottom: end effector rotation is constrained to point downward.

deep, requiring the robot's gripper to move in continuous "forward and backward" paths from the robot's point of view. The Baxter robot platform was a popular choice for this competition (4 teams in 2015 and 6 in 2016, including the author's teams). Anecdotally, we found that it was difficult to place Baxter at a position and height so that it could reach deeply into all bins in the shelf, and our motion planner often struggled to find feasible paths. Ultimately, we placed the robot using trial-and-error.

We retrospectively applied our algorithm to study this problem more rigorously, comparing the redundancy resolution for the gripper in downward-facing and forward-facing orientations. Fig. 13 illustrates the results, demonstrating that the forward-facing orientation creates many more discontinuities directly in the area in front of the robot where the shelf is naturally located. Hence, we conclude that Baxter's kinematics are not well-suited for forearm-parallel grippers to reach into deep horizontal bins. Rather, the downward orientation is advantageous for maximum horizontal flexibility of movement. The maximum reachable height at this orientation also suggests that the robot should be elevated 30-50cm higher than the standard pedestal, and the gripper should be extended perpendicularly from the forearm to avoid collisions with bin boundaries.

For the team's entry to the 2017 APC (rebranded as the Amazon Robotics Challenge), we conducted a prospective analysis. The 2017 competition rules require teams to design their own shelving system, which gives significantly greater flexibility in designing the workspace. We also decided to change robots to the Staubli TX90L 6DOF industrial manipulator with a thin, long end effector. We considered the questions of 1) should the shelving system be oriented with horizontal or vertical bin openings, and 2) where should it be placed to guarantee reachability? Fig. 14 displays the results from applying our method. For shelves with horizontal openings, the area directly in front of the robot is fraught with discontinuities. Shifting the robot to the side greatly improves reachability, and elevating the robot improves it even further. A downward-pointing gripper can easily reach all around the robot in a torus, with maximum reach and gripper-base self-collision the only constraints on movement.

VII. Performance and Scalability Improvements

The algorithm presented above is practical for relatively low dimensional spaces where the number of workspace nodes $|V_W|$ is in the thousands and the number of configuration samples per node N_q is at most one hundred. For the larger examples presented above, computation times are approximately 1 day and memory (RAM) consumption is a few gigabytes. To scale to higher-dimensional workspaces, such as the 6D space of positions and orientations, much larger workspace grids and more configuration samples are needed. To do so, we developed an implementation that uses a smarter discretization of configuration space, out-of-core memory storage, and parallel processing.

A. Self-Motion Manifold Visibility PRM Implementation

A speed and memory bottleneck in our algorithm is the $O(N_q^2)$ all-pairs C-space visibility tests conducted per workspace edge. Most of these visibility tests are unnecessary since only one configuration per node will ultimately be picked. Also, many configurations per workspace node are roughly equivalent in that they can be connected to the same sets of neighbors, so adding them to the CSP increases computation time and memory usage without significant improvements to solution quality. We apply a technique inspired by the Visibility PRM algorithm [18], which is a PRM-like planner that avoids performing edge visibility checks when two nodes are in the same connected component of the roadmap, since adding such edges do not improve the roadmap's connectivity. This eliminates many unnecessary and costly collision checks.

In this variant, we apply this reasoning to the self-motion manifolds at workspace points. Observe that, except for singular points and points on the feasible space boundary, all configurations in a given self-motion manifold connected component (SMMCC) at a workspace point x can be smoothly modified to reach a nearby workspace point $x + \delta x$ with δx sufficiently small. As a result, the connectivity of SMMCCs along the workspace graph is approximately equivalent to the connectivity of individual configurations. Since there are far fewer SMMCCs than configurations (usually just a handful), this can lead to large performance improvements.

The new approach operates as follows:

1) Construct Visibility PRM SMMs at each workspace point via sampling (Fig. 15.a),

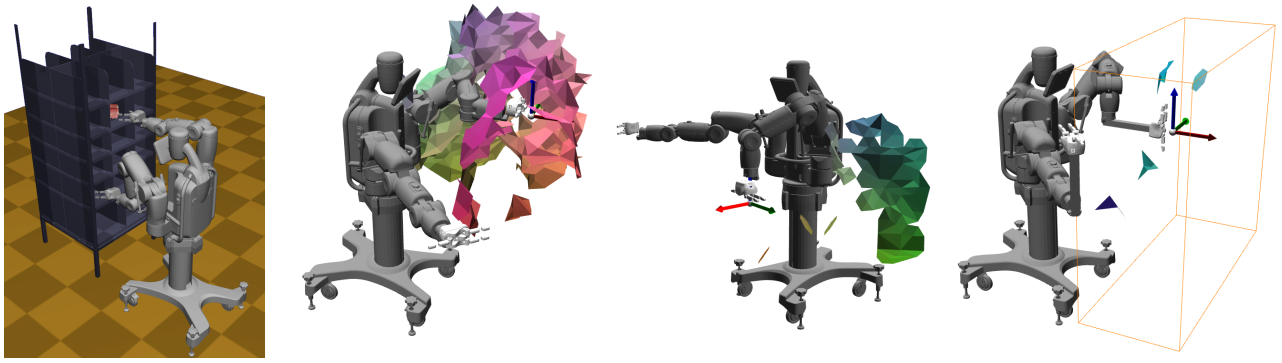


Fig. 13. From left to right: Baxter shown next to the Amazon Picking Challenge shelving unit; optimized result with gripper facing forward; result with gripper facing down; candidate alternate design with a sideways gripper extension. The extension allows the gripper to freely enter the shelf horizontally in the area front of the robot. Wireframe box indicates the desired workspace.

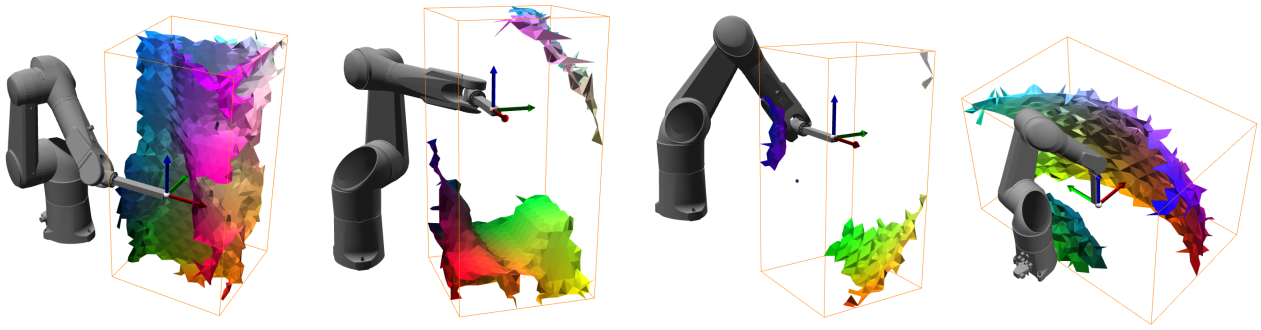


Fig. 14. Testing reachability at different positions and orientations of the Staubli manipulator with a long gripper. Left: if a vertical shelf were placed directly in front of the robot, singularities prevent freedom of movement. Middle left: shifted to the side, most discontinuities are removed. Middle right: shifting downward removes most discontinuities. Right: a downward gripper orientation admits free movement in a torus around the robot. Wireframe box indicates the desired workspace, and boundaries of the reachable set are drawn.

2) Construct C-space edges connecting SMMCCs across workspace edges, stopping when only a single feasible edge is found (Fig. 15.b),
3) Solve a CSP only on SMMCCs (Fig. 15.c), and
4) "Lift" the SMMCC resolution to a C-space resolution.

To test C-space edges efficiently between SMMCCs in Step 2, we test candidate edges in order of increasing C-space distance. We also only test at most the c shortest candidate edges plus an additional c longer edges, chosen at random.

For Step 3, the CSP is constructed with one variable per workspace node as usual, but whose domain is the index of the SMMCC in the SMM. This drastically reduces the set of possible values in the CSP. A constraint is imposed along each workspace edge dictating that there must exist a feasible edge between selected components.

The SMMCC redundancy resolution needs to be remapped, since each incoming edge to the SMMCC does not necessarily match the configuration of an outgoing edge. If used directly, the resolved SMMCC roadmap may require the robot to re-configure in-place to move about in the workspace (Fig. 15.c). Instead Step 4 tries to find one configuration in each resolved SMMCC that is connected to each neighboring SMMCC via a C-space edge. This is usually (but not always) possible. Our process for doing so is as follows.

First, we sequentially and greedily attempt to determine

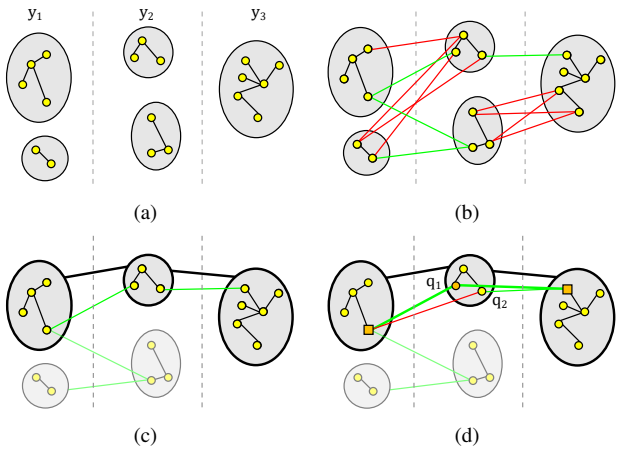


Fig. 15. Illustrating the visibility-based redundancy resolution. (a) Self-motion manifold connected components (SMMCCs) are built at each workspace node y_1, y_2, y_3 using Visibility-PRM. (b) Edges connecting SMMCCs are checked. (c) A CSP solves for a resolution among SMMCCs. (d) A C-space resolution is determined via lifting. At y_2, the Find-Hub procedure tries to find "hub" configurations that connect to all adjacent "anchors" (squares). q_1 is found to be a hub, but q_2 fails the check to the anchor at y_1.

"hub" configurations for each SMMCC. Let y be the workspace node and S be the SMCC under consideration. A configuration $q \in S$ is a hub if it has a connection in C-space to the resolved SMMCC S' of every adjacent workspace node

y'. Moreover, we require that if y' was previously resolved with hub q', then q and q' are connected.

The Find-Hub procedure operates as follows. First, the set of configurations at neighboring SMMCCs S' that are endpoints of existing feasible C-space paths are called "anchors" and their starting points are considered "candidates." First, all candidates are checked for being hubs to all anchors (Fig. 15.d). If this fails, configurations between candidates on the SMMCC subgraph are checked for being hubs to all anchors. Finally, if this fails, the candidate covering the maximum number of anchors is determined and all edges to configurations $q' \in S'$ are checked (as long as y' does not have a determined hub).

Find-Hub is performed for all workspace nodes, and if any fail, we perform a fallback lifting phase. We then perform all-pairs redundancy resolution locally around the workspace nodes V_F for which Find-Hub fails. Let V_1 be the 1-ring of V_F (the set of nodes 1 step away from V_F) and let V_2 be its 2-ring. All-pairs edge checks are conducted between all nodes in the resolved SMMCCs of $V_F \cup V_1$, and edge checks are conducted between the hubs of V_2 and all configurations in the SMMCCs of V_1. A C-space CSP is then solved over the resulting C-space roadmap with the variables being the configurations assigned to nodes $V_F \cup V_1$.

B. Parallel and Out-of-Core Memory Implementation

We store a key-value database containing information associated to workspace nodes and edges. Node data consists of the workspace parameter, a list of configurations, and the topology of the self-motion manifold at that node. Edge data consists of a list of the feasible edges of the C-space graph, which are represented by the indices of the configurations at adjoining workspace nodes. Each entry contains tens or hundreds of kb of data depending on N_q. As many entries of the database as possible are stored in resident memory up to a given cache size C. When C is exceeded, entries are dumped from RAM to disk in least-recently used (LRU) fashion.

Parallel processing is performed by a master process that assigns jobs to worker threads, retrieves the results, and inserts results into the cached database. First, the master assigns workers to sample SMMs at workspace nodes. Next, the workers are assigned to perform connectivity checks along each workspace edge. This step requires the master to read back the entire graph from disk. However, the performance drop is negligible because the process is computation bound by a large margin.

Finally, to create the CSP, the entire graph must be read again from disk, and each SMM connected component is assigned as a value for each workspace node. The number of components and edges per node is typically small, so the CSP can be solved in main memory.

The resulting resolution stores at most one configuration per workspace node, so it can be easily kept in memory (165 mb in our largest examples). The optimization postprocessing steps are also performed in main memory.

C. Experiments

Tests on our enhanced visibility-based algorithm are conducted on a 64-core shared memory server machine with 128 GB RAM. Fig. 16 illustrates the speed improvements of the enhanced technique compared to the basic resolution. These tests were conducted on the ATLAS robot example. Total time is broken up into Intra, Inter, and CSP components. Referring to the steps in the algorithm outlined in Sec. VII-A, these components indicate the work done at a single workspace nodes (Step 1), workspace edges (Step 2), and to resolve the CSP (Steps 3+4), respectively. Compared to the basic algorithm, Intra and CSP computations are more expensive due to the additional work done in the visibility graph construction and lifting steps. However, the Inter computations are massively sped up, leading to a net improvement of one or two orders of magnitude.

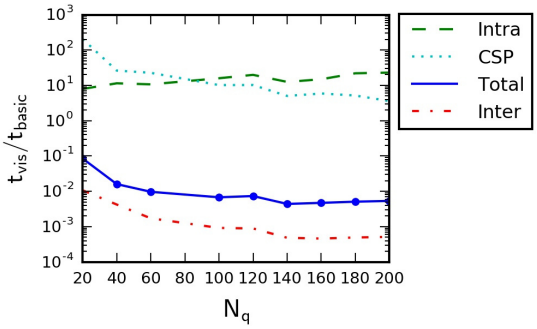


Fig. 16. Speed improvement of the main phases of the visibility-based algorithm relative to the basic algorithm as a function of the C-space sample count N_q, including the work done at single workspace nodes (Intra), at workspace edges (Inter), and solving the CSP (CSP). As N_q increases the total speedup approaches two orders of magnitude.

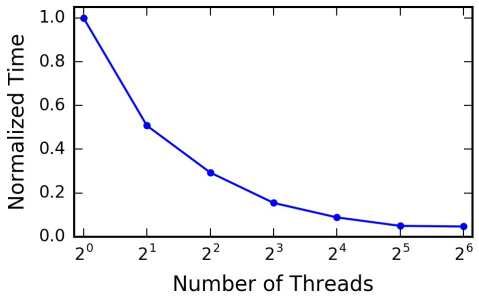


Fig. 17. Normalized running time as a function of the number of threads in our parallel implementation, showing linear speedups.

Fig. 17 displays the time taken by the parallel implementation relative to the single-threaded implementation. Speedups are nearly linear. Other experiments found that the cache size C had very little effect on performance, which demonstrates that the algorithm is CPU-bound.

Ultimately, the improved implementation can generate large resolutions, consisting of hundreds of thousands of workspace nodes and billions of configurations and visibility tests, in hours on the server. Fig. 18 demonstrates an example with the Baxter robot on a 6D position and orientation workspace with 451,440 workspace nodes. The space SO(3) was discretized using a grid on the quaternion sphere. With $N_q = 200$, computation on our server took 4 days. The figure illustrates two slices of the resolution at different end-effector orientations.

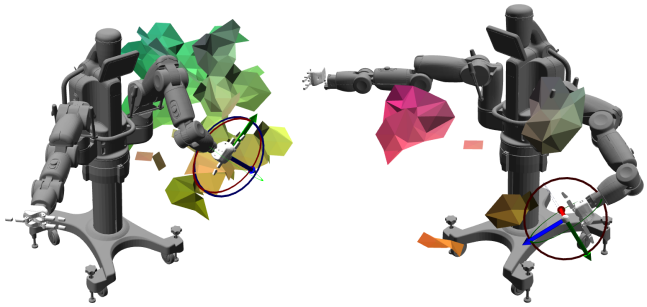


Fig. 18. The redundancy resolution for a 6D Cartesian position and orientation workspace with 451,440 nodes. 3D slices through the position dimensions are shown for two end-effector orientations.

VIII. Conclusion

This paper presented a method for global redundancy resolution for Cartesian workspace movements of robot manipulators. The method computes an approximate, maximally continuous pseudoinverse of the multivariate forward kinematic map using a roadmap-based approach, which is solved as a constraint satisfaction problem (CSP). Software for the algorithms presented in this paper and more examples are available at http://motion.pratt.duke.edu/redundancyresolution/.

Future work should investigate how the resulting maps may be used for robot mechanism design, teleoperation, or reduced dimensionality motion planning. We are also interested in studying the extension of global resolution to higher dimensional workspaces, and applying it to problems like mobile manipulation.

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] N. Correll, K. Bekris, D. Berenson, O. Brock, A. Causo, K. Hauser, K. Okada, A. Rodriguez, J. Romano, and P. Wurman. Analysis and observations from the first amazon picking challenge. *IEEE Transactions on Automation Science and Engineering*, PP(99), 10 2016.

[4] J. Cortes. *Motion Planning Algorithms for General Closed-Chain Mechanisms*. PhD thesis, Laboratoire d'Analyse et d'Architecture des Systémes (LAAS), Toulouse, France, 2003.

[5] P. Freeman. *Minimum Jerk Trajectory Planning for Trajectory Constrained Redundant Robots*. PhD thesis, Washington University of Saint Louis, 2011.

[6] 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.

[7] L. Han. A kinematics-based probabilistic roadmap method for closed chain systems. In *Workshop Alg. Found. Rob.*, 2000.

[8] K. Hauser. Continuous pseudoinversion of a multivariate function: Application to global redundancy resolution. In *Workshop Alg. Found. Rob.*, San Francisco, USA, 2016.

[9] C. A. Klein and C. H. Huang. Review of pseudoinverse control for use with kinematically redundant manipulators. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13(2):245–250, March 1983.

[10] 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.

[11] 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.

[12] Y. Nakamura. *Advanced Robotics: Redundancy and Optimization*. Addison-Wesley, New York, 1991.

[13] G. Oriolo, M. Cefalo, and M. Vendittelli. Repeatable motion planning for redundant robots over cyclic tasks. *IEEE Transactions on Robotics*, 33(5):1170–1183, 2017.

[14] G. Oriolo and C. Mongillo. Motion planning for mobile manipulators along given end-effector paths. In *IEEE Int. Conf. Rob. Aut.*, 2005.

[15] 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.

[16] 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.

[17] N. Shvalb, G. Liu, M. Shoham, and J. Trinkle. Motion planning for a class of planar closed-chain manipulators. *Int. J. Rob. Res.*, 26(5):457 – 473, 2007.

[18] T. Siméon, J. Laumond, and C. Nissoux. Visibility based probabilistic roadmaps for motion planning. In *Int. Conf. on Intel. Robots and Systems*, 1999.

[19] M. Stilman. Global manipulation planning in robot joint space with task constraints. *IEEE Trans. Robotics*, 26:576–584, 2010.

[20] G. Tack. *Constraint Propagation - Models, Techniques, Implementation*. PhD thesis, Saarland University, Germany, 2009.

[21] N. Tamura, A. Taga, S. Kitagawa, and M. Banbara. Compiling finite linear csp into sat. *Constraints*, 14(2):254–272, June 2009.

[22] R. H. Taylor. Planning and execution of straight line manipulator trajectories. *IBM J. Res. Dev.*, 23(4):424–436, July 1979.

[23] J. Trinkle and R. J. Milgram. Complete path planning for closed kinematic chains with spherical joints. *Int. J. Rob. Res.*, 21(9):773–789, 2002.

[24] 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.

Kris Hauser is an Associate Professor at Duke University with joint appointments in the Departments of Electrical & Computer Engineering and Mechanical Engineering and Materials Science. He received his PhD in Computer Science from Stanford University in 2008, bachelor's degrees in Computer Science and Mathematics from UC Berkeley in 2003, and was a postdoc at UC Berkeley. He joined the faculty of Indiana University from 2009–2014, where he started the Intelligent Motion Lab, and began his current position at Duke in 2014. He is a recipient of a Stanford Graduate Fellowship, Siebel Scholar Fellowship, Best Paper Award at IEEE Humanoids 2015, and an NSF CAREER award.

Research interests include robot motion planning and control, semiautonomous robots, and integrating perception and planning, as well as applications to intelligent vehicles, robotic manipulation, robot-assisted medicine, and legged locomotion.

Scott Emmons Scott Emmons is an undergraduate student studying math and computer science at the University of North Carolina at Chapel Hill and Duke University through the Robertson Scholars Leadership Program. As an undergraduate, he is exploring applied math and computer science research. He has experience in robotics, network science, and information visualization.