Redundancy resolution based on optimization

Jane Li

Assistant Professor
Mechanical Engineering Department, Robotic Engineering Program
Worcester Polytechnic Institute

Quiz (10 pts)

- (6 pts) Explain the optimization/tradeoff underlying the damped least square method

$$\min_{\dot{q}} \frac{\mu^2}{2}\|\dot{q}\|^2 + \frac{1}{2}\|\dot{x} - J\dot{q}\|^2 = H(\dot{q})$$

- (2 pts) List two metrics that measure the distance from singularity

- (2 pts) How to guarantee the secondary task will not interfere the primary task

Singularity avoidance – Damped Least Squares

unconstrained minimization of a **suitable** objective function

$$\min_{\dot{q}} \frac{\mu^2}{2}\|\dot{q}\|^2 + \frac{1}{2}\|\dot{x} - J\dot{q}\|^2 = H(\dot{q})$$

compromise between large joint velocity and task accuracy

SOLUTION

$$\dot{q} = J_{DLS}(q)\dot{X} = J^T\left(JJ^T + \mu^2 I_M\right)^{-1}\dot{X}$$

- To render robust behavior when crossing the singularity, we can add a small constant along the diagonal of $(J(q)^T J(q))$ to make it invertible when it is singular

Distance to singularity

- Manipulability index – Jacobian matrix determinant

$$\mu = \sqrt{|\mathbf{J}\mathbf{J}^T|}$$

- Which is indeed

$$\mu = \prod_{i=1}^{M} \sigma_i$$

- Is it a good measurement?

Distance to singularity

- Manipulability index – condition number

$$\kappa = \frac{\sigma_{\max}}{\sigma_{\min}}$$

Range?

- Alternatively, can use isotropy

$$Isotropy = \frac{\sigma_{\min}}{\sigma_{\max}}$$

- Is it good enough?

Distance to singularity

- Manipulability index – the smallest singular value

$$\sigma_{\min}$$

- Direction of velocity disadvantage

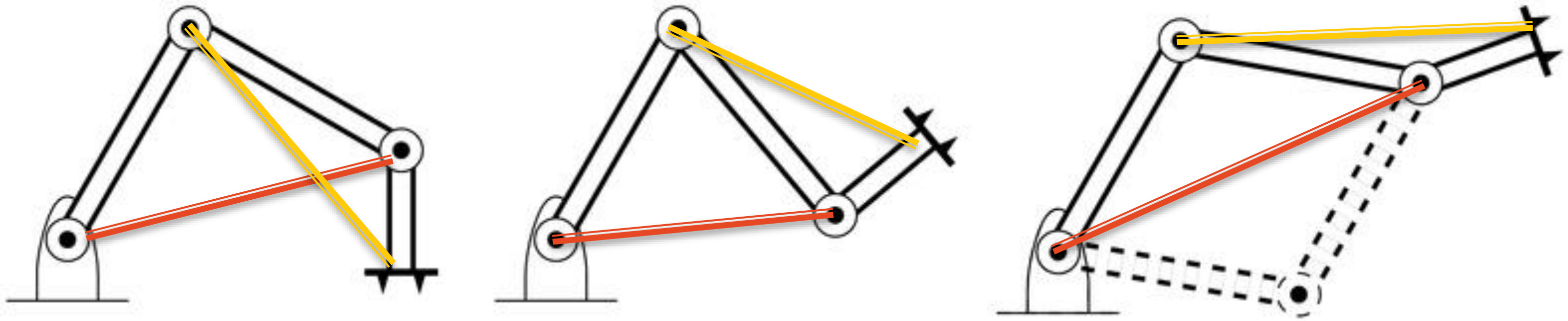
- Is it good enough?

Distance to singularity

- Manipulability index

$$\mu' = \sum_{i=1}^{M} \sqrt{|\mathbf{J}_i \mathbf{J}_i^T|}$$

- What does it imply?
  - Manipulability of every sub-manipulator (non-redundant)



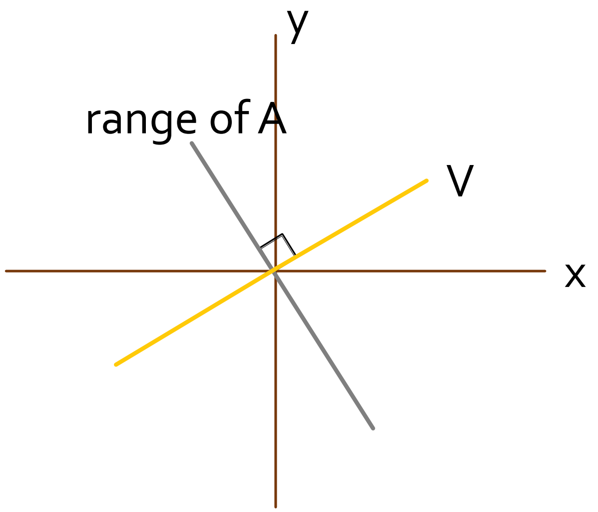
The Null-space of Jacobian

- Secondary tasks is satisfied in the ***null-space*** of the Jacobian pseudo-inverse

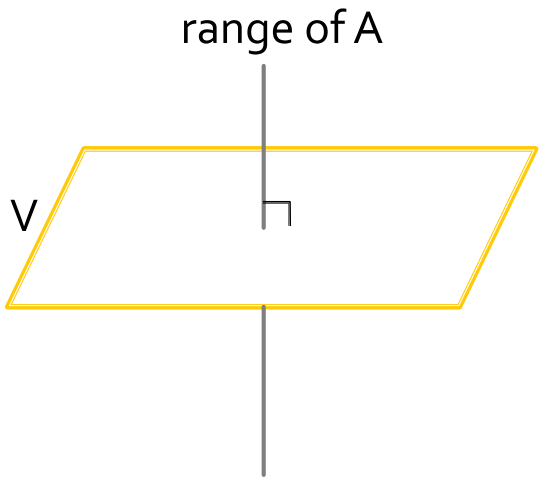
  - In linear algebra, the ***null-space*** of a matrix A is the set of vectors V such that, for any v in **V**, $0 = A^T v$.

  - **V** is orthogonal to the range of A

  -



2D example



3D example

The Null-space of Jacobian

- Given the null space of Jacobian, the secondary task will not disturb the primary task

- The null-space projection matrix for the Jacobian pseudo-inverse is:

$$N(q) = I - J(q)^{\dagger} J(q)$$

-

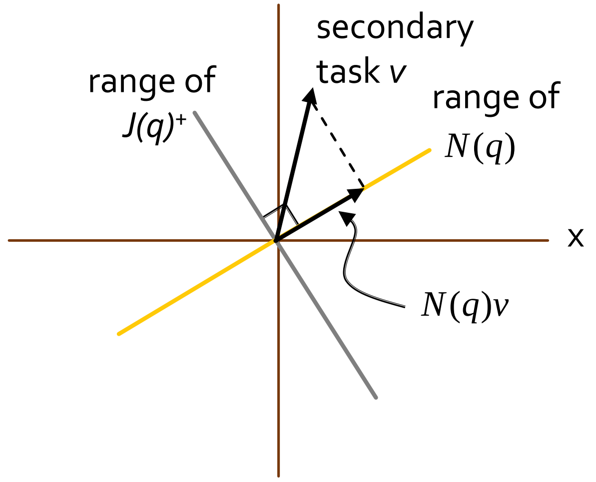
The Null-space of Jacobian

- Project a task space velocity vector into the null-space

$$\dot{q} = J(q)^{\dagger}\dot{x} + (I - J(q)^{\dagger}J(q))J_c(q)^{\dagger}\dot{x}_c$$

Primary task — $J(q)^{\dagger}\dot{x}$

Secondary task — $(I - J(q)^{\dagger}J(q))J_c(q)^{\dagger}\dot{x}_c$



Redundancy resolution based on optimization

Still a problem …

- Methods for redundancy resolution has been studied for decades, yet there are still unsolved problems
- Multi-objective Optimization
  - What are the optimization criteria?
  - How to assign weighting coefficients?

Robot manipulator – Performance to optimize

- Manipulability
- Force/velocity transmission efficiency
- Energy
- Motion smoothness
- Task accuracy

Performance indices	Formula	Comments
Determinant of Jacobian (1984)	$w_n = \sqrt{JJ^T}$	Uniformity of the torque-velocity gain
Condition number (1982)	$\kappa = \frac{\sigma_{max}}{\sigma_{min}}$	Variance in velocity/force transmission
Isotropy (1987)	$Iso = \frac{\sigma_{min}}{\sigma_{max}}$	same as condition number
Min eigen-value of Jacobian (1987)	$Iso = \frac{\sigma_{min}}{\sigma_{max}}$	Efficiency of force/velocity transmission
Dynamic Manipulability (1985)	$G = J^{-T}MJ^{-1}$	Uniformity of this torque-acceleration gain
Distance from singularity (1987)	$H = \left\|\prod_i^p \Delta_i\right\|^{1/p}$	Related to manipulability by $w_n = \sqrt{\sum_i^p \Delta_i}$
Acceleration radius (1988)	$\tau = M(\ddot{\theta}) + C(\theta, \dot{\theta})\dot{\theta}$	acceleration capability of the end-effector
Force transmission ratio (1988)	$\alpha = [(u^T(JJ^T)u]^{1/2}$	Force gain along task-compatibility direction
Velocity transmission ratio (1988)	$\beta = [u^T(JJ^T)^{-1}u]^{1/2}$	Velocity along task-compatibility direction
Min Jerk model (1984)	$\min(\frac{\partial^3 x}{\partial t^3})$	Motion smoothness
Min (commanded) torque-change (1985,1989)	$\min(\frac{\partial \tau}{\partial x})$	Motion smoothness
Min work model (1983)	$\min(W)$	Energy
Min variance model (1989)	$\min[var(x - x_d)]$	Task accuracy

Common Objectives for Redundant Resolution

- Tracking end-effector trajectory → primary task
- Obstacle avoidance
  - Pseudoinverse – Incorporate obstacle as secondary constraints
  - Artificial potential field – repulsive obstacle + attractive target
- Motion limits
  - Position, velocity, acceleration
  - Avoid vibration, improve motion smoothness

Consistent and predictable robot behavior

- To be consistent and predictable, robot motion needs to be repetitive in both task and configuration space
  - Close path in task space → close path in configuration space

- Unpredictable robot behavior
  - Joint angle drift
  - Readjusting the manipulators‘ configuration with self-motion at every cycle → inefficient

Methods

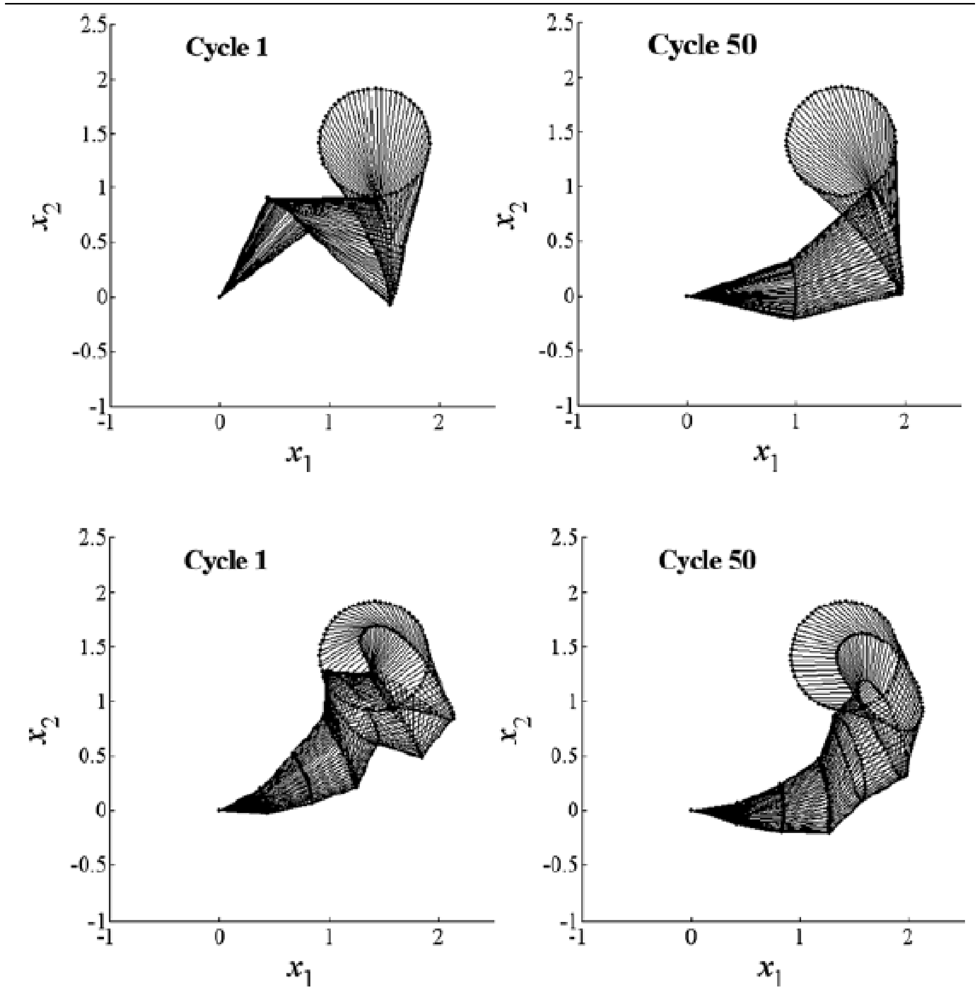
- Baseline = Closed-loop pseudo-inverse
- Define a cost function to optimize for motion repetition, and solve it using
  - Genetic Algorithm [1]
  - Dynamical quadratic programming [2]
- Continuous pseudo-inverse and global redundancy resolution [3]

Closed-loop pseudo-inverse

- Compute the joint position through <u>time integration</u> pseudo-inverse

$$\Delta q = \mathbf{J}^{\dagger} \Delta x$$

Unpredictable, not repeatable arm configurations



Closed-loop pseudo-inverse + Genetic Algorithm

$$\Delta q = \mathbf{J}^{\dagger} \Delta x$$

$$\Delta \mathbf{x}^* = \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \\ \Delta x_{m+1} \\ - - - \\ \vdots \\ \Delta x_n \end{bmatrix}$$

Generated by GA

$$\mathbf{J}^* = \begin{bmatrix} -l_1 S_1 - \cdots - l_n S_{1\ldots n} & \ldots & -l_n S_{1\ldots n} \\ l_1 C_1 + \cdots + l_n C_{1\ldots n} & \ldots & l_n C_{1\ldots n} \\ - - - - - - - - & - - - & - - - - \\ j_{(m+1)1} & \ldots & j_{(m+1)n} \\ \ldots & \ldots & \ldots \\ j_{n1} & \ldots & j_{nn} \end{bmatrix}$$

Begin
- $T = 0$
- *calculate* $\Delta \mathbf{x} = \mathbf{x}_{ref} - \mathbf{x}_{ini}$, $\mathbf{J}$
- *initialize random population*

$$P(T) = \left[\left[\mathbf{J}^{(T,1)} \,\vdots\, \Delta\mathbf{x}^{(T,1)}\right], \ldots, \left[\mathbf{J}^{(T,N)} \,\vdots\, \Delta\mathbf{x}^{(T,N)}\right]\right]$$

- *get* $\Delta \mathbf{q} = \mathbf{J}^{*-1}(\mathbf{q}) \Delta \mathbf{x}^*$ *and* $\mathbf{q} = \int \Delta \mathbf{q}$
- *evaluate* $P(T)$
- ***repeat***
  - Use GA to update P(T)
  - **Cost function?**
  - *get* $\Delta \mathbf{q} = \mathbf{J}^{*-1}(\mathbf{q}) \Delta \mathbf{x}^*$ *and* $\mathbf{q} = \int \Delta \mathbf{q}$
  - *evaluate* $P(T)$
  - $T = T + 1$
- ***until*** *termination condition is TRUE*
- *get new* $\mathbf{q}$

End

Cost function for GA

$$f_1 = A\dot{\mathbf{q}}^T\dot{\mathbf{q}} + B\left(\frac{\mathbf{q}-\mathbf{q}_0}{\Delta t}\right)^T\left(\frac{\mathbf{q}-\mathbf{q}_0}{\Delta t}\right)$$

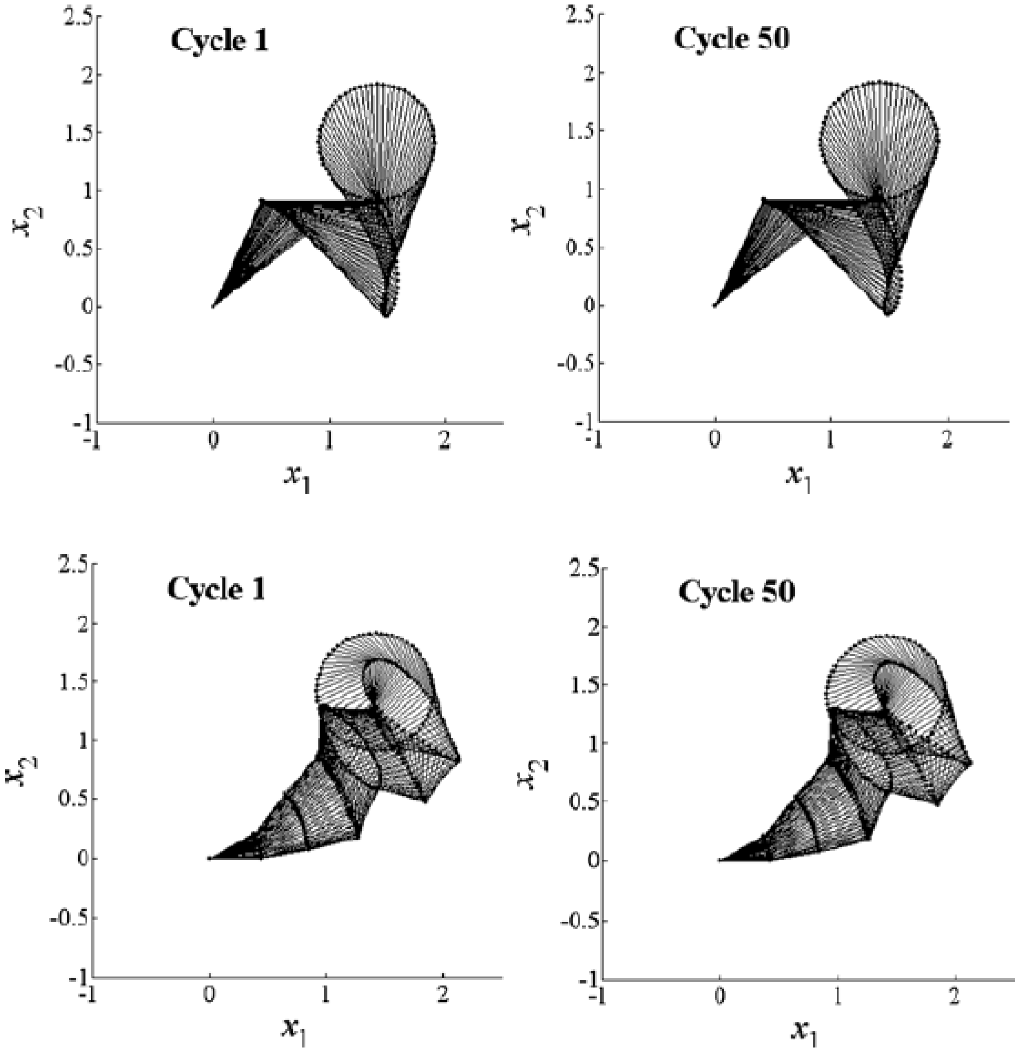
Weighted least squares

Minimize the displacement between initial and current joint configurations over a time step

Simulation Result

CLGA	$r = 0.7$	$r = 1.0$	$r = 2.0$
$3R$	9.96E−04	8.84E−04	1.08E−03
$4R$	7.12E−04	7.38E−04	5.70E−04
$5R$	6.73E−04	5.42E−04	6.15E−04
$6R$	5.98E−04	4.81E−04	8.57E−04
$7R$	1.26E−03	5.44E−04	5.39E−04

CLP	$r = 0.7$	$r = 1.0$	$r = 2.0$
$3R$	1.35E+01	6.41E+00	5.80E−01
$4R$	8.2E+00	4.4E+00	5.8E−01
$5R$	7.2E+00	2.2E+00	4.4E−01
$6R$	5.4E+00	4.9E+00	3.0E−01
$7R$	4.2E+00	2.4E+00	2.0E−01



Multi-objective optimization

- Formulation of Optimization Problem

$$\text{minimize} \quad \frac{(\dot{\theta} + \mathbf{p})^{\mathrm{T}} (\dot{\theta} + \mathbf{p})}{2}$$

$$\text{subject to} \quad J_{\mathrm{e}}(\theta)\dot{\theta} = \dot{\mathbf{r}}_{\mathrm{d}}$$

$$J_{\mathrm{o}}\dot{\theta} \leqslant \mathbf{b}_{\mathrm{o}}$$

$$\zeta^{-} \leqslant \dot{\theta} \leqslant \zeta^{+}$$

- $\frac{(\dot{\theta} + \mathbf{p})^{\mathrm{T}} (\dot{\theta} + \mathbf{p})}{2}$ → **Repetitive motion**
- $J_{\mathrm{e}}(\theta)\dot{\theta} = \dot{\mathbf{r}}_{\mathrm{d}}$ → **Tracking EE trajectory**
- $J_{\mathrm{o}}\dot{\theta} \leqslant \mathbf{b}_{\mathrm{o}}$ → **Obstacle constraints**
- $\zeta^{-} \leqslant \dot{\theta} \leqslant \zeta^{+}$ → **Joint limit**

Formulation of Optimization Problem

$$\text{minimize} \quad \frac{(\dot{\theta}+\mathbf{p})^{\mathrm{T}}(\dot{\theta}+\mathbf{p})}{2}$$

$$\text{subject to} \quad J_{\mathrm{e}}(\theta)\dot{\theta} = \dot{\mathbf{r}}_{\mathrm{d}}$$

$$J_{\mathrm{o}}\dot{\theta} \leqslant \mathbf{b}_{\mathrm{o}}$$

$$\zeta^{-} \leqslant \dot{\theta} \leqslant \zeta^{+}$$

Repetitive motion

$$\mathbf{p} = \eta(\theta(t) - \theta(0))$$

$$\frac{\|\dot{\theta}(t) + \eta(\theta(t) - \theta(0))\|_2^2}{2}$$

$$\mathbf{z}(t) = \theta(t) - \theta(0)$$
$$\eta > 0 \in R$$

$$\Rightarrow \quad \dot{\mathbf{z}}(t) = -\eta\mathbf{z}(t) \quad \Rightarrow \quad \|\mathbf{z}(t)\|_2 = \exp(-\eta t)\|\mathbf{z}(0)\|_2 \to 0$$

$$\theta(t) = \theta(0), \; t \to \infty$$

Dynamical quadratic programming

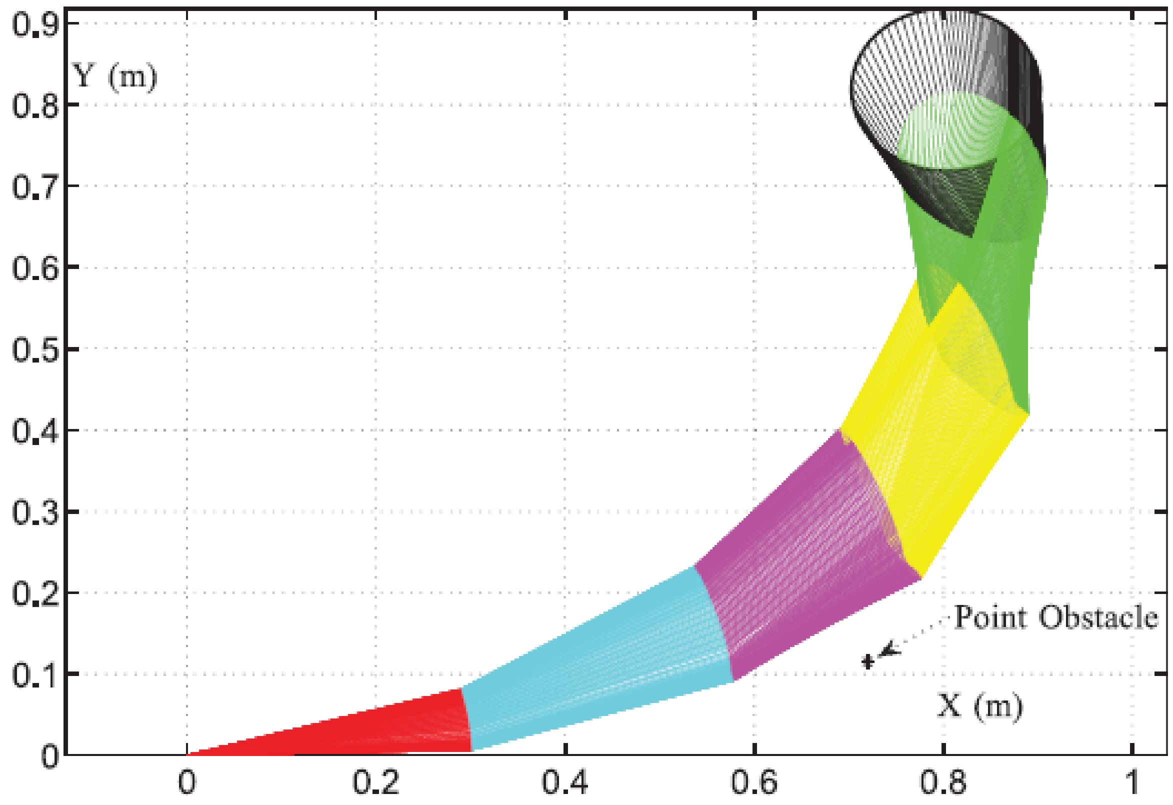
$$\begin{aligned} \text{minimize} \quad & \frac{(\dot{\theta} + \mathbf{p})^{\mathrm{T}}(\dot{\theta} + \mathbf{p})}{2} \\ \text{subject to} \quad & J_{\mathrm{e}}(\theta)\dot{\theta} = \dot{\mathbf{r}}_{\mathrm{d}} \\ & J_{\mathrm{o}}\dot{\theta} \leqslant \mathbf{b}_{\mathrm{o}} \\ & \zeta^{-} \leqslant \dot{\theta} \leqslant \zeta^{+} \end{aligned}$$

$$\Rightarrow$$

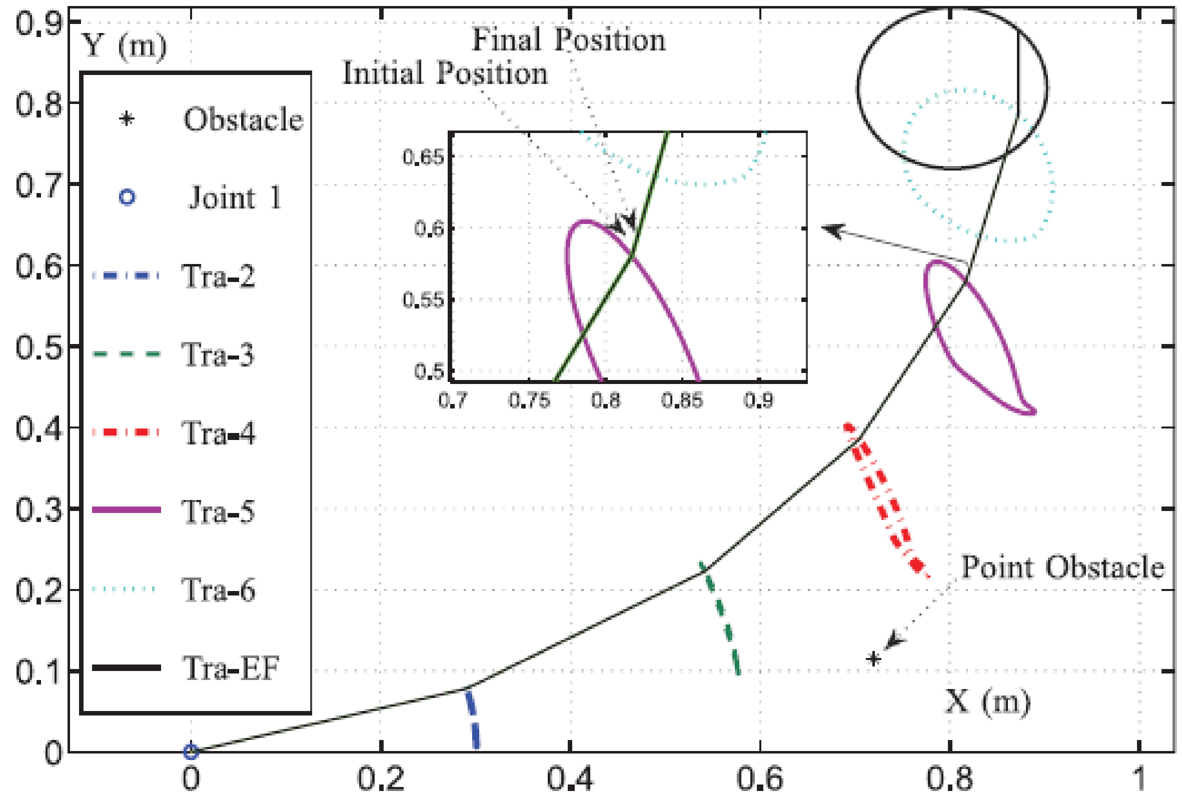
$$\begin{aligned} \text{minimize} \quad & \frac{\mathbf{x}^{\mathrm{T}} Q \mathbf{x}}{2} + \mathbf{p}^{\mathrm{T}}\mathbf{x} \\ \text{subject to} \quad & A\mathbf{x} = \mathbf{d} \\ & C\mathbf{x} \leqslant \mathbf{b} \\ & \zeta^{-} \leqslant \mathbf{x} \leqslant \zeta^{+} \end{aligned}$$

- Dynamical quadratic program (DQP) with equality, inequality, and bound constraints
  - Can be solved by piecewise-linear projection equation (PLPE) neural network

Simulation Result [1]

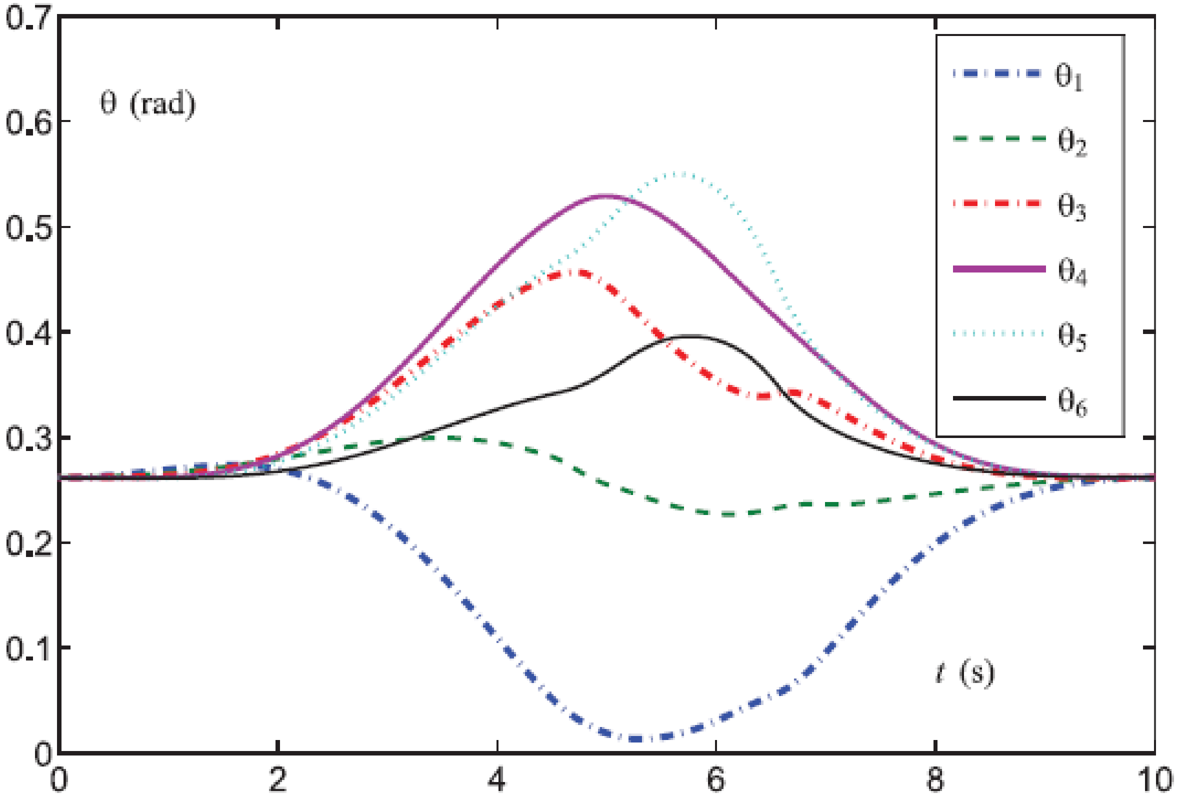


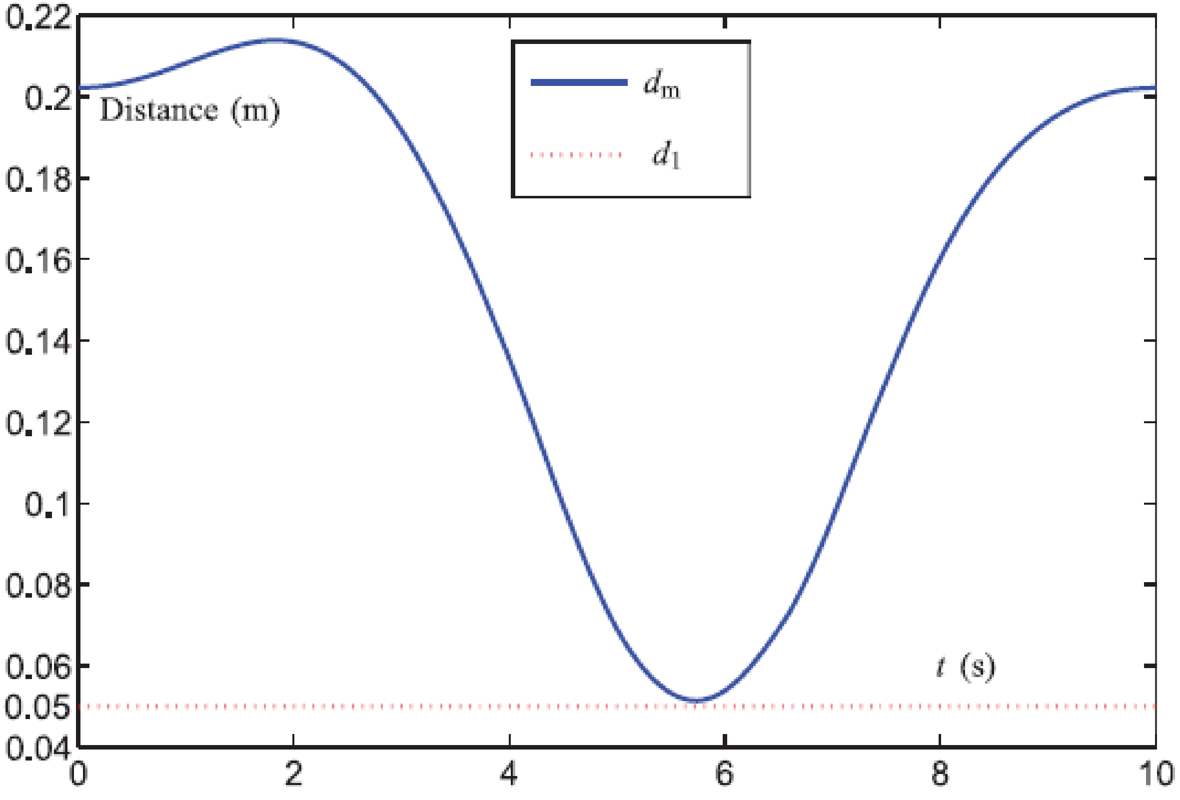
Simulation motion



Motion of each joint in task space

Simulation Result



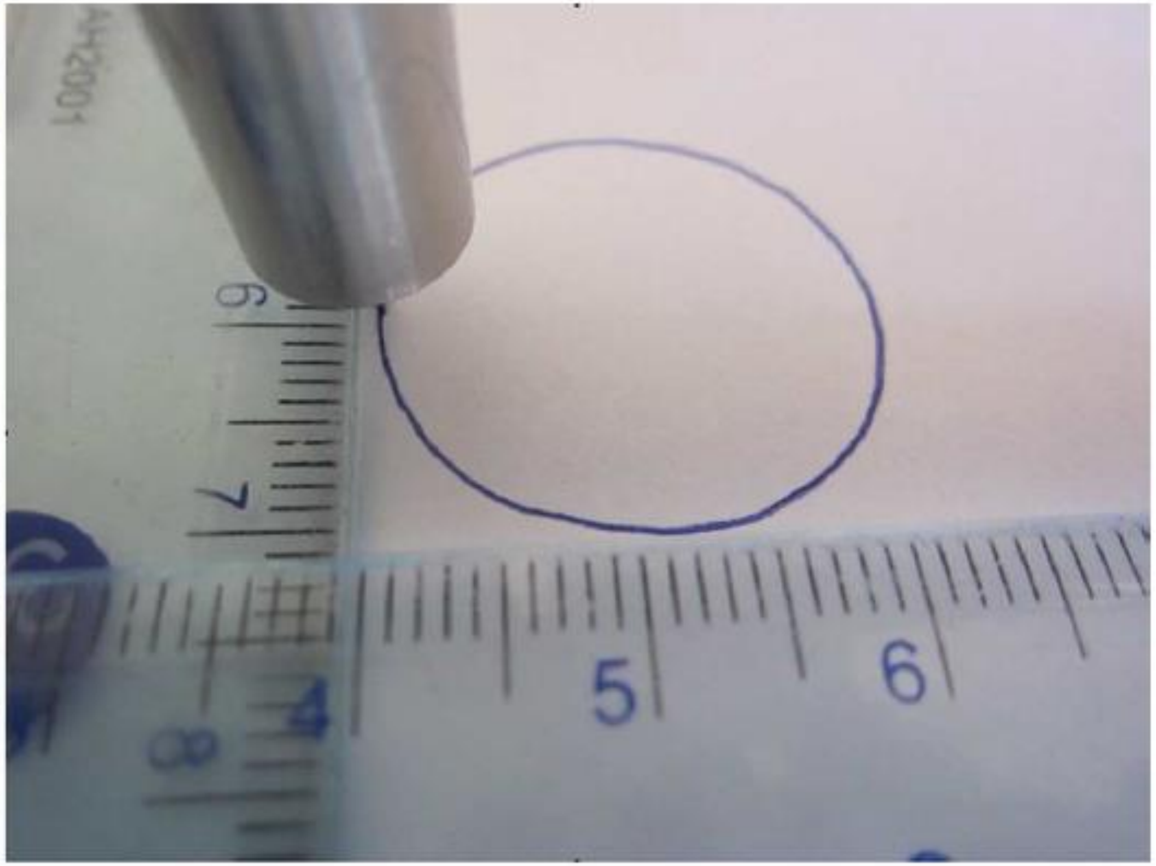


Joint angles

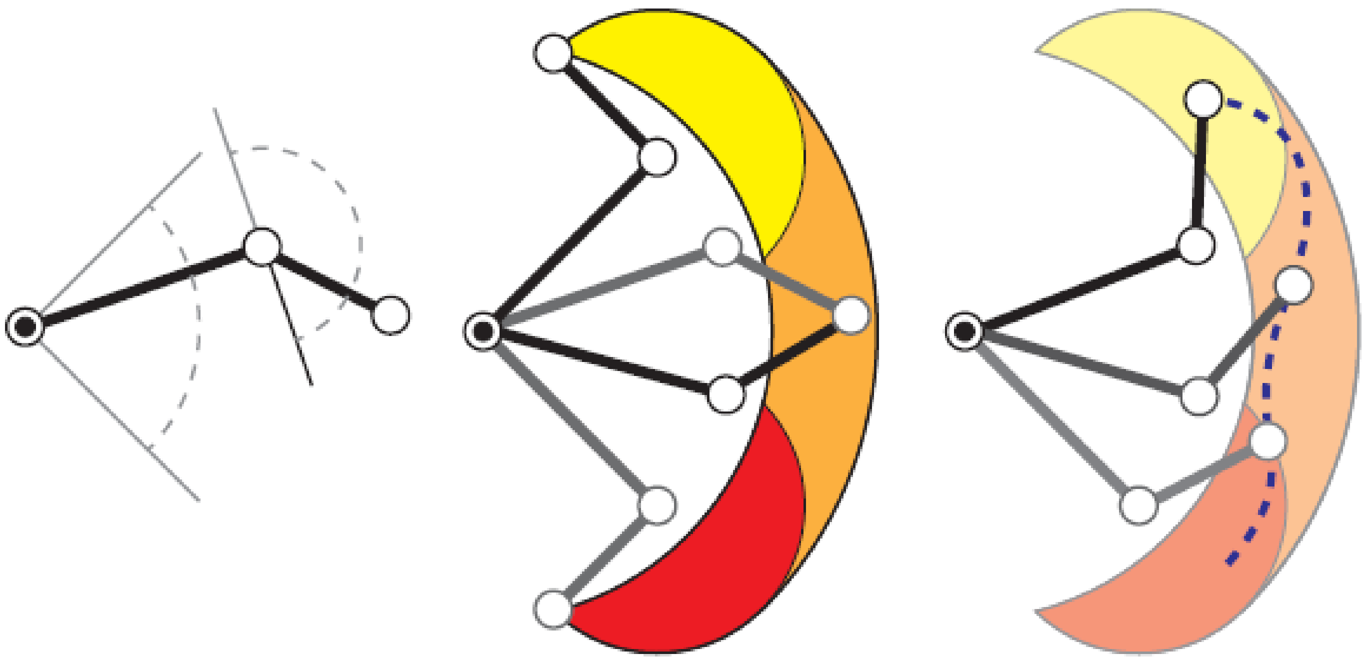
Minimal link-obstacle distance

Experiment

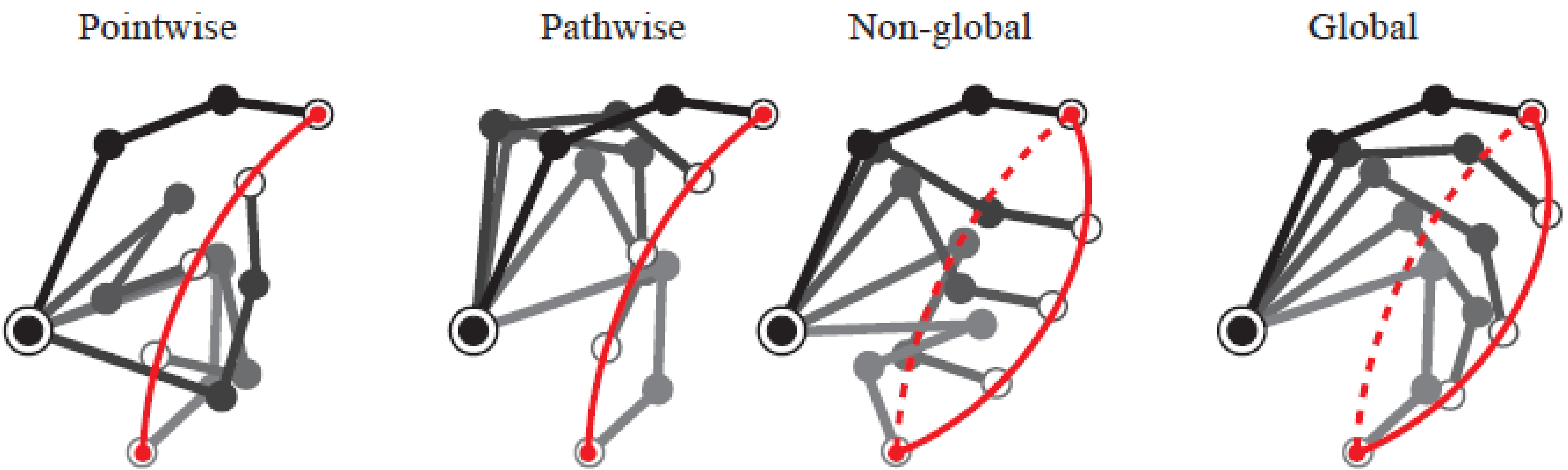

Control unit
Host computer
Joint 1
Joint 2
Joint 3
Joint 4
Joint 6
Base



Practical needs in robot control



Continuous, globally consistent redundancy resolution

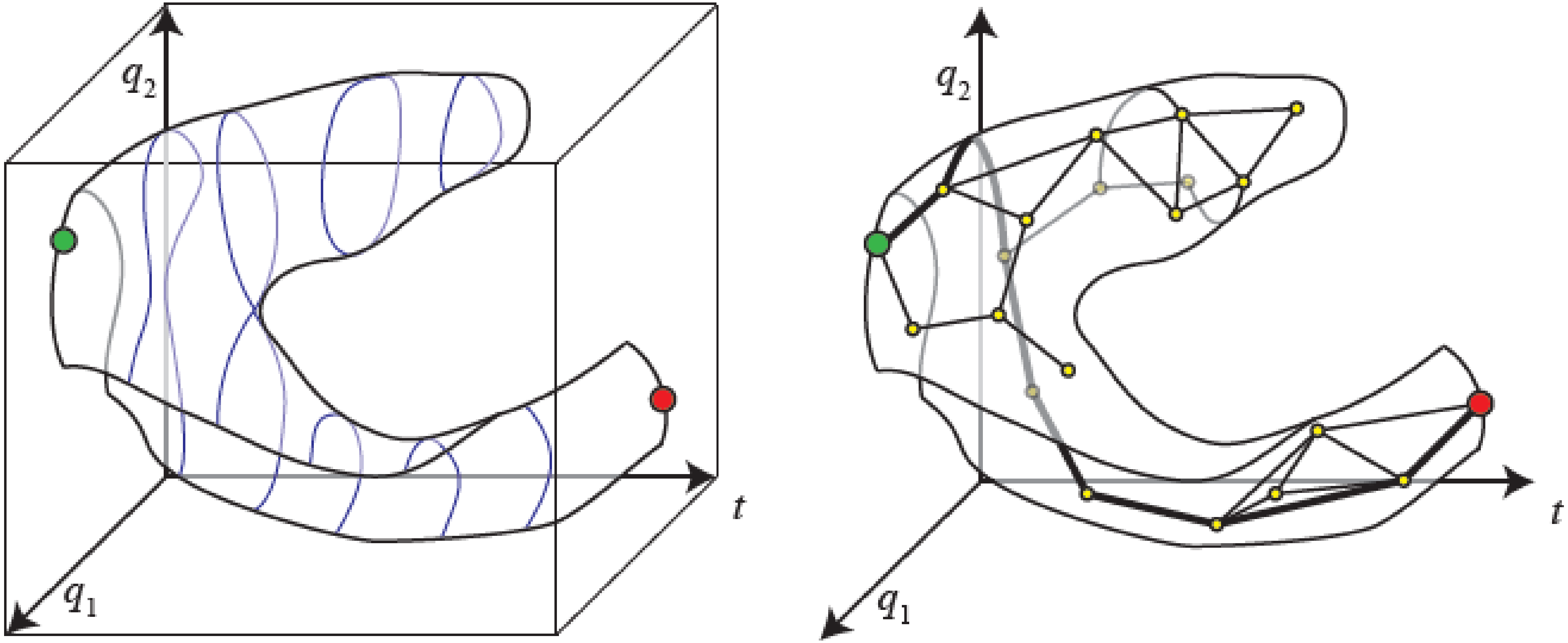


Continuity and global consistency

- Continuity of redundancy resolution
  - Starting joint configuration was chosen "badly", then the robot tracking a simple path could get stuck when it hits joint limits.

- Globally consistent redundancy resolution
  - When tracking a cyclic path (forward and backward), the robot should return to the same joint configuration that it started from

Pathwise Redundancy Resolution



PRM-Path Resolution

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

```
Initialize empty roadmap 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 ~ U([0, 1])
    Sample q_sample ←SampleF(y(t_sample))
    if q_sample ≠ 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 𝓡 for a path from t = 0 to t = 1
```

Add start and end points in configuration space

PRM-Path Resolution

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

```
Initialize empty roadmap 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 ~ U([0, 1])
    Sample q_sample ←SampleF(y(t_sample))
    if q_sample ≠ nil then add (t
for all nearby pairs of vertices (
    if Visible(y, t_u, t_v, q_u, q_v) the
        Add the (directed) edge
Search R for a path from t = 0 to t = 1
```

– 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.

– 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}.

PRM-Path Resolution

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 $i = 1, \ldots, N$ **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 $t = 1$

Sampling in the time domain – every node added subject to the manifold constraints

PRM-Path Resolution

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 $t = 1$

Local planner – directed edges restrict forward progress along the time domain

PRM-Path Resolution

- Local planner

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

```
if d(q_s, q_g) ≤ ε then return "true"
Let y_m ← y((t_s + t_g)/2) and q_m ← (q_s + q_g)/2
Let q ← Solve(y_m, q_m)
if q = nil or q ∉ F then return "false"
if max(d(q, q_s), d(q, q_g)) > c · 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"
```

– 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}.

Approximate global redundancy resolution

- Assign a single robot configuration to each target point
- Pointwise global resolution
- Constraint-satisfaction-based resolution

Pointwise global resolution

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** add q to V_C and go to Step 2, proceeding to the next y.
7: Run SampleF(y) up to N_q times. If any sample q succeeds, add it to V_C.
8: **for** all edges $(y, y') \in E_W$ such that $|Q(y)| > 0$ and $|Q(y')| > 0$ **do**
9: Let q be the only member of $Q(y)$ and q' the only member of $Q(y')$
10: **if** R(y, y', q, q')=1 **then**
11: Add (q, q') to E_C
return $\mathcal{R}_C$

$N(y)$ is the neighborhood of a vertex y in the workspace graph

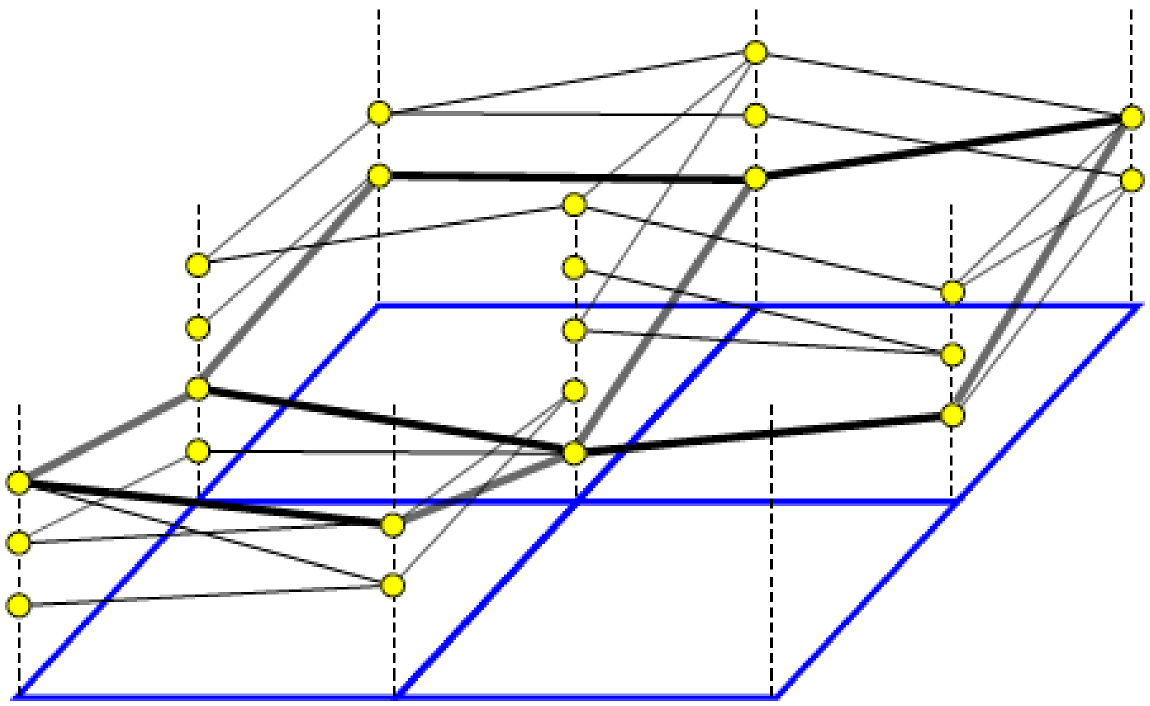
Pointwise global resolution

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** add q to V_C and go to Step 2, proceeding to the next y.
7: Run SampleF(y) up to N_q times. If any sample q succeeds, add it to V_C.
8: **for** all edges $(y, y') \in E_W$ such that $|Q(y)| > 0$ and $|Q(y')| > 0$ **do**
9: Let q be the only member of $Q(y)$ and q' the only member of $Q(y')$
10: **if** R(y, y', q, q')=1 **then**
11: Add (q, q') to E_C
return $\mathcal{R}_C$

Keep only one configuration

Pointwise global resolution



Limitation of pointwise method

- Pointwise method can yield poor results
  - Several edges unnecessarily unresolved
- Constraint-satisfaction problem
  - Sample many configurations in the preimage of each workspace point
  - Connect them with feasible edges
  - Seek a “sheet” in the C-space roadmap that satisfies the constraints

Constraint-satisfaction-based resolution

- Primary error metric
  - Measures the number of unresolved edges
- Secondary error metric
  - Maximize smoothness in the redundant dimensions

Minimize the number of unsolvable edges

- Let $G_W = (V_W, E_W)$ be the workspace roadmap

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

Local reachability indicator function – check for locally pairwise resolvable

- Seek the mapping ***g*** from task space vertices to C-space vertices

Maximize pseudo-inverse smoothness

- Distance is a good proxy for smoothness.
  - Use total C-space path length to measure smoothness

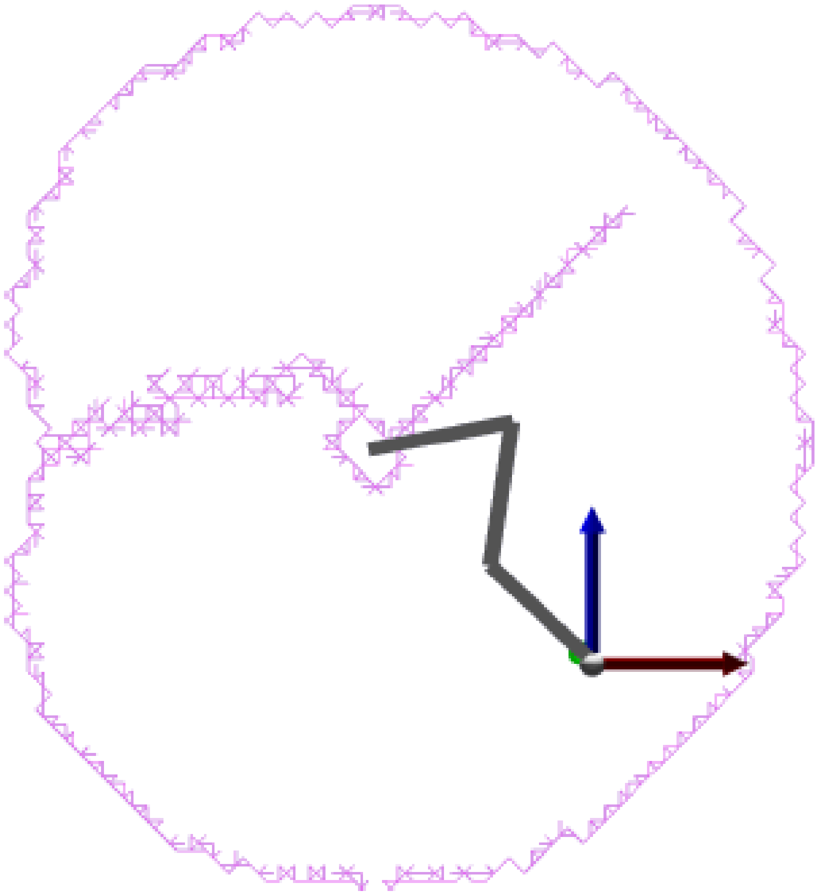
$$L(g) = \sum_{(y,y')\in E_W} d(g[y], g[y'])R(y, y', g[y], g[y'])$$

Ensure connection in C-space and task space

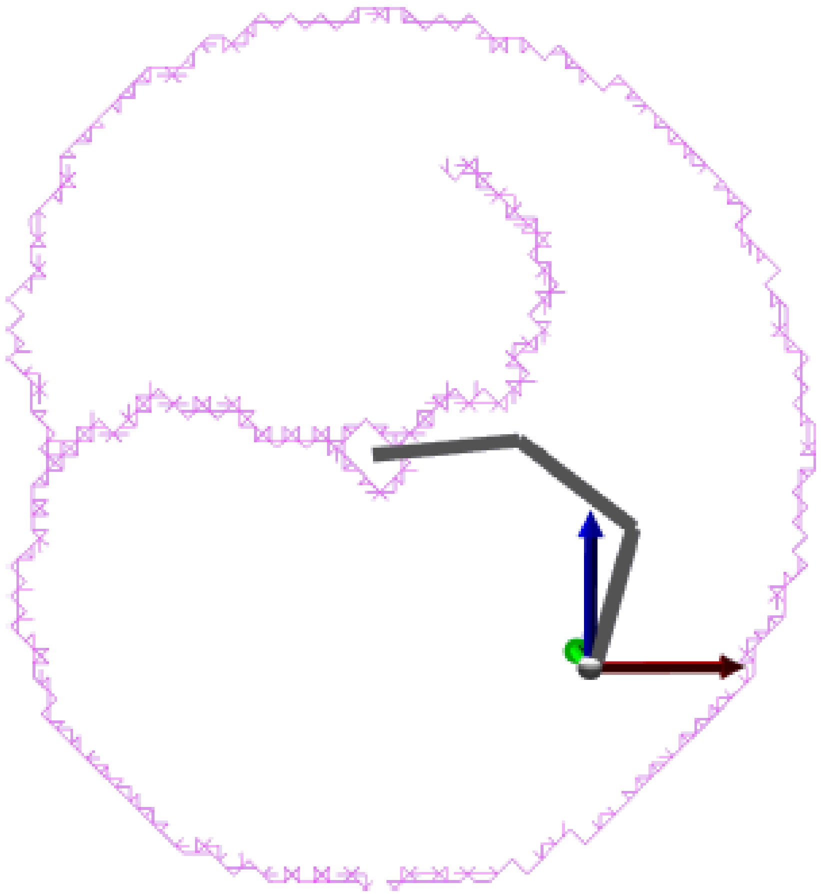
- Given the C-space roadmap $R = (V_c, E_c)$, make sure

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

Discontinuity boundary for 3-DOF arm

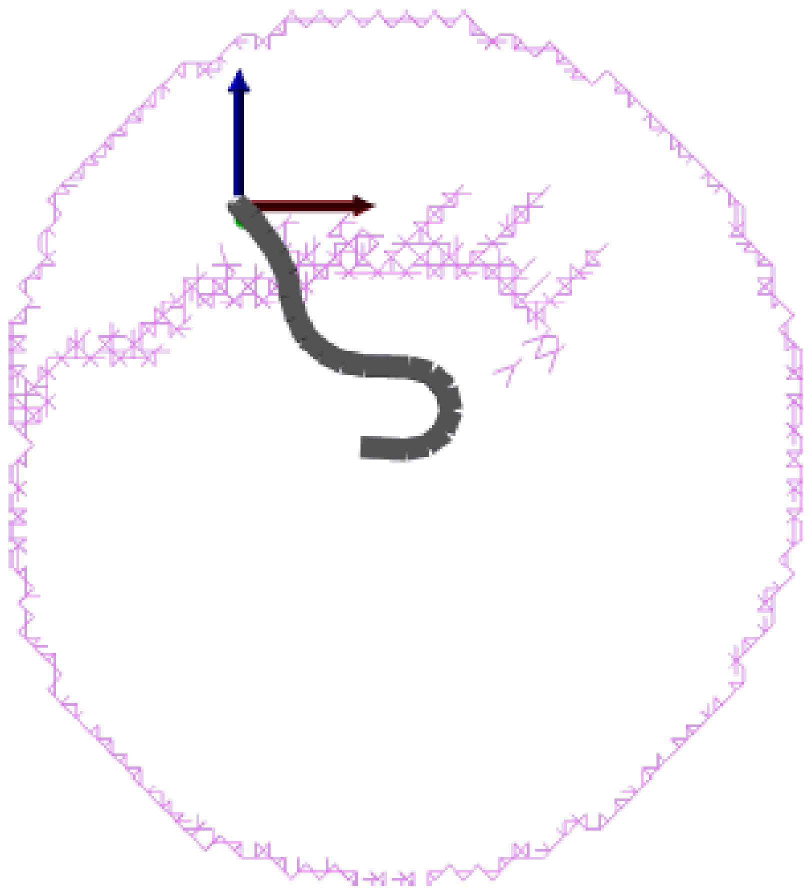


Pointwise solution

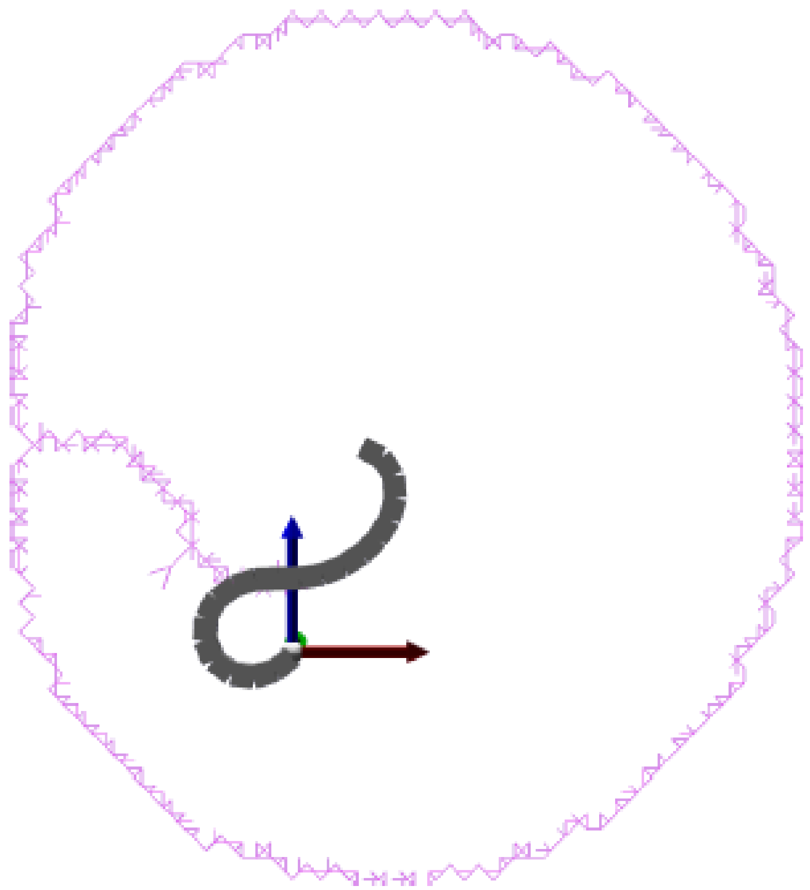


Optimization-based solution

Discontinuity boundary for 3-DOF arm



Pointwise solution



Optimization-based solution

Reference

- [1] da Graça Marcos, M., Machado, J. T., & Azevedo-Perdicoúlis, T. P. (2010). An evolutionary approach for the motion planning of redundant and hyper-redundant manipulators. *Nonlinear Dynamics*, *60*(1-2), 115-129.

- [2] Chen, D., & Zhang, Y. (2017). A hybrid multi-objective scheme applied to redundant robot manipulators. IEEE Transactions on Automation Science and Engineering, 14(3), 1337-1350.

- [3] Hauser, K. (2017). Continuous pseudoinversion of a multivariate function: Application to global redundancy resolution.
- http://motion.pratt.duke.edu/redundancyresolution/