Manipulation Motion Planning

Jane Li

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

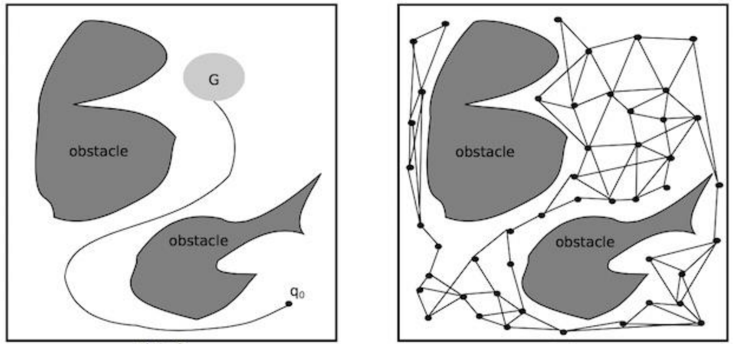
Quiz (10 pts)

- (3 pts) Compare the testing methods for testing path segment and finding first collision

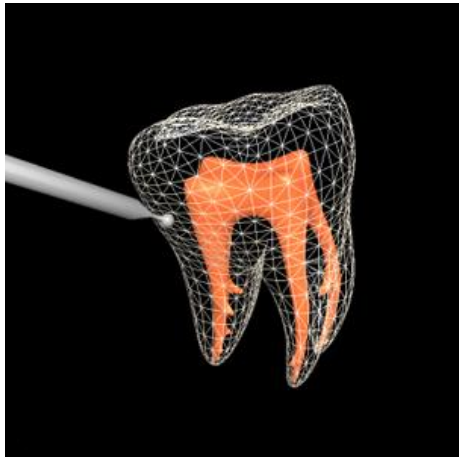
- Compare the non-holonomic RRT with holonomic RRT: given a new node to connect to,
  - (3 pts) how to extend toward this node?
  - (3 pts) how to connect to this node for the last step?

Testing Path Segment vs. Finding First Collision

- PRM planning
  - Detect collision as quickly as possible → Bisection strategy



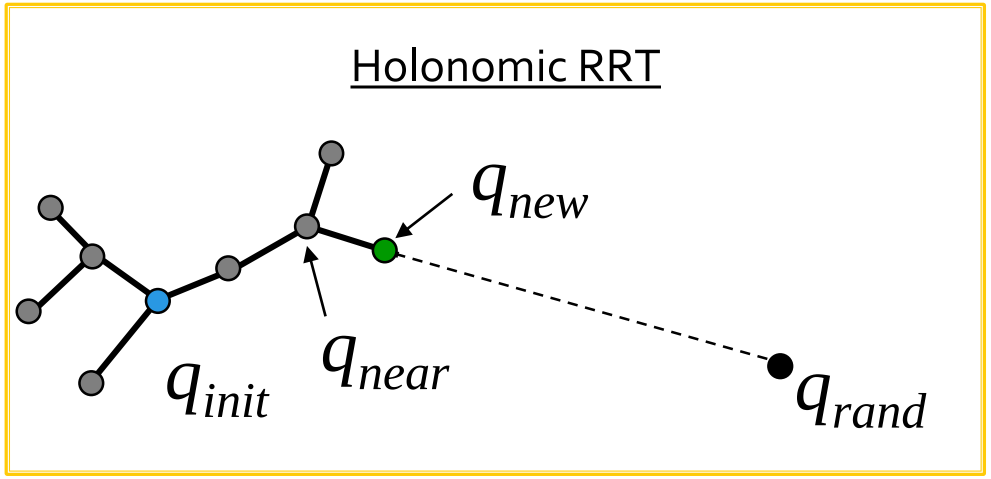
- Physical simulation, haptic interaction
  - Find first collision → Sequential strategy

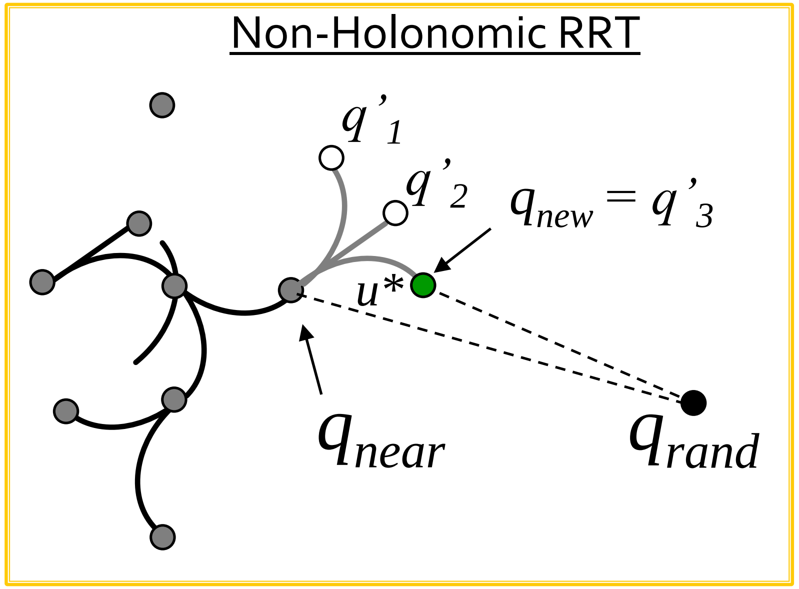


RRTs for Non-Holonomic Systems

- Apply motion primitives (i.e. simple actions) at q_{near}

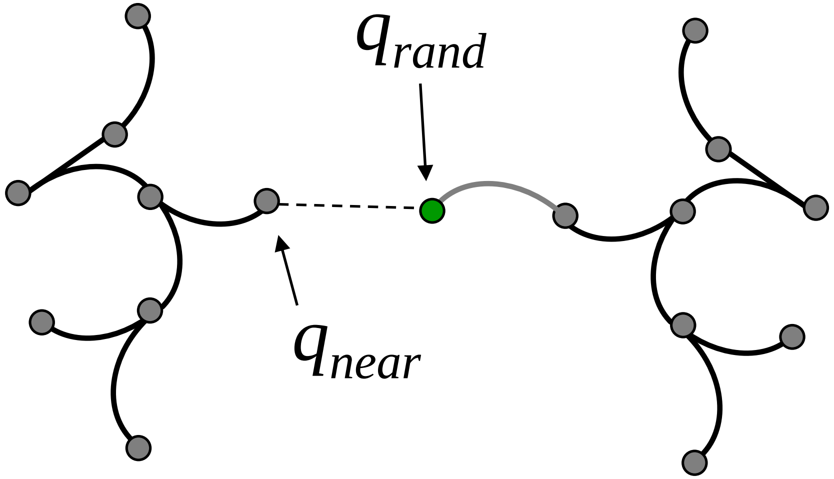
$$q' = f(q,u) \text{ --- use action } u \text{ from } q \text{ to arrive at } q' \qquad \text{chose } u_* = \arg\min(d(q_{rand}, q'))$$





- You probably won't reach q_{rand} by doing this
  - Key point: No problem, you're still exploring!

BiDirectional Non-Holonomic RRT



How to bridge between the two points?

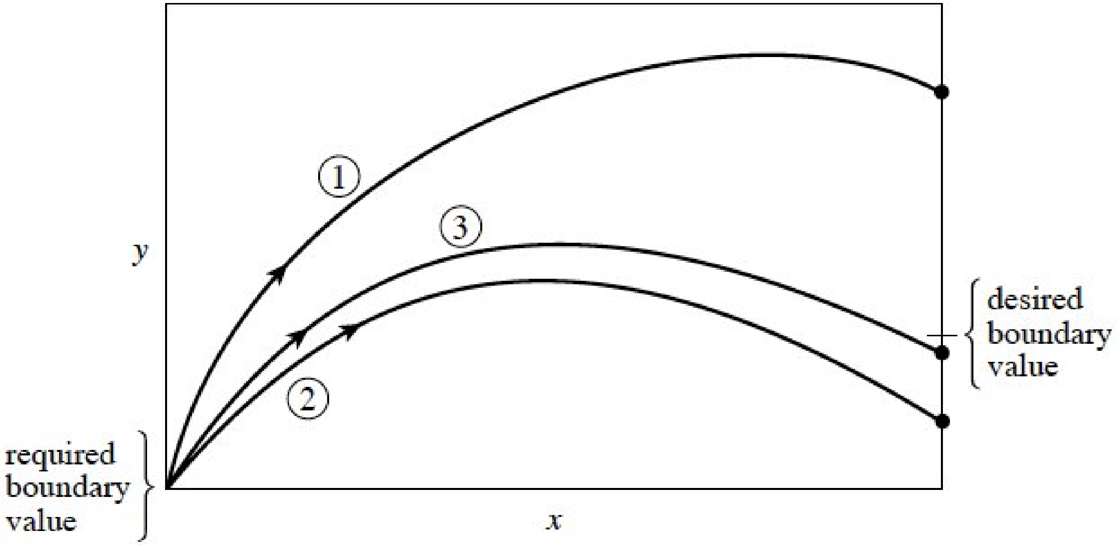
Shooting Method

- “Shoot” out trajectories in different directions until a trajectory of the desired boundary value is found.
  - System

$$\frac{dy}{dx} + f(x, y) = 0$$

  - Boundary condition

$$y(0) = 0,\ y(1) = 1$$



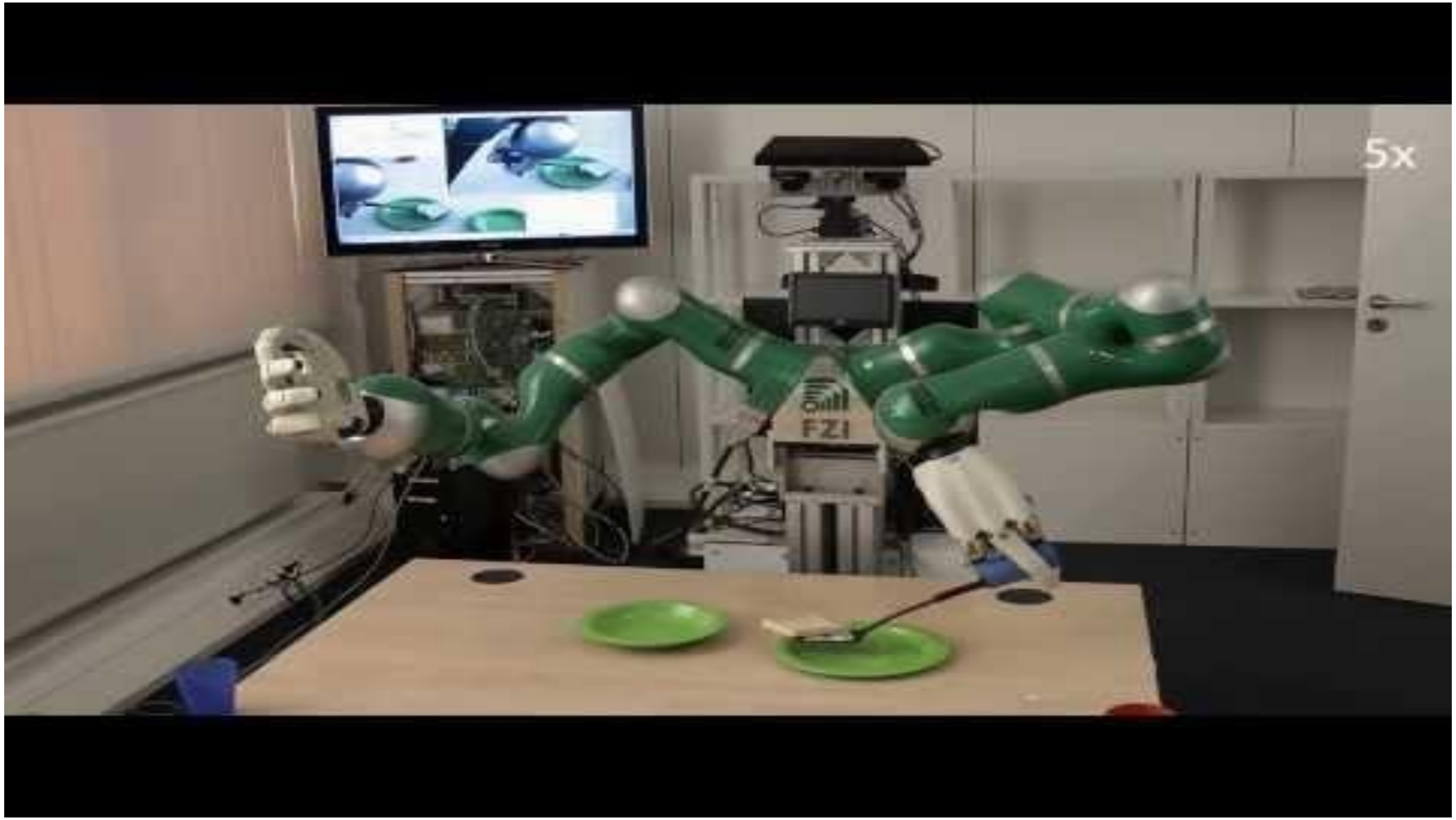
Manipulation motion planning

Recap

- We have learned the planning algorithms that can generalize across many types of robots
  - Discrete planning
  - Sampling-based planning

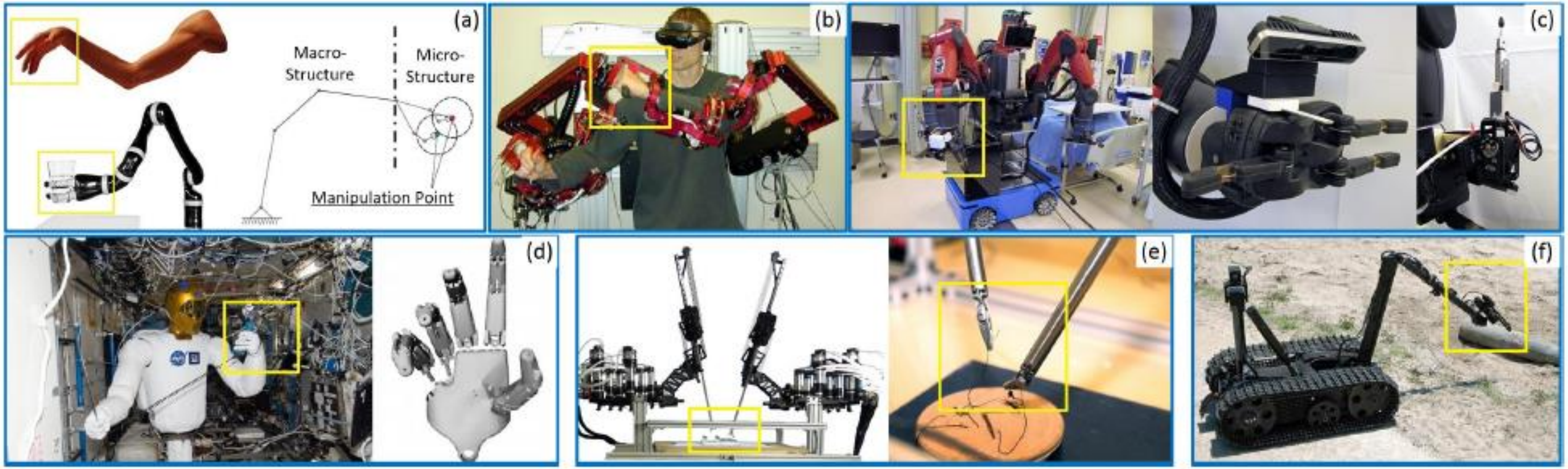
- Theoretically, we should be all set. However …
  - When it comes to manipulator robots, we may have to handle an application-specific problem

Bimanual humanoid robot



Mobile manipulator robot

Kinematically redundant manipulators



Research Questions

- How to resolve the kinematic redundancy?

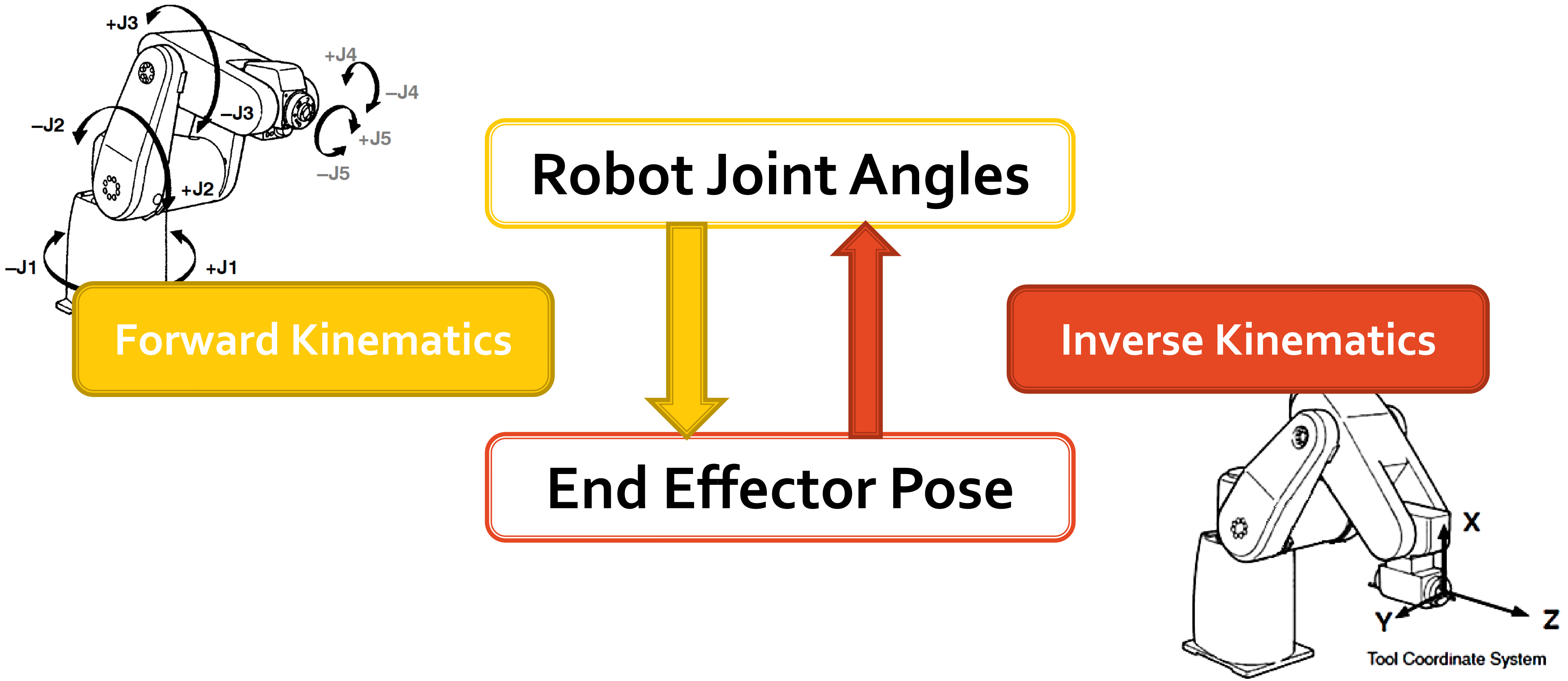
- How to coordinate macro- and micro-structures?
  - Arm-hand structure
  - Body-arm structure

- How to handle bimanual coordination?

Overview

- How to resolve the kinematic redundancy?
  - Solution to Inverse kinematics
  - Pseudo-inverse
  - Additional constraints and optimization criteria

Forward and inverse kinematics



Robot Joint Angles

Forward Kinematics

Inverse Kinematics

End Effector Pose

Tool Coordinate System

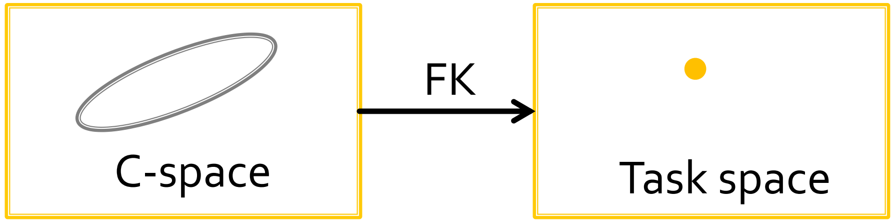
Kinematic Redundancy

$f: Q \rightarrow R$

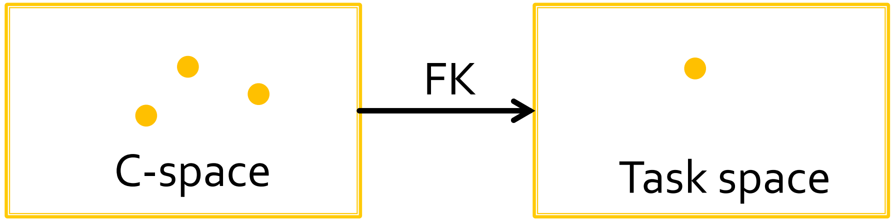
joint space (dim Q = N)

task space (dim R = M)

- If N>M,
  - FK maps *a continuum* of configurations to *one* end-effector pose:



- If N=M,
  - FK maps a finite number of configurations to one end-effector pose:



- If N<M,
  - Target pose not reachable

Kinematics at different levels

- Direct kinematics

$$x = FK(\mathbf{q})$$

- First-order differential kinematics – Jacobian

$$\dot{x} = J(\mathbf{q})\dot{\mathbf{q}}$$

- Second-order differential kinematics

$$\ddot{x} = J(\mathbf{q})\ddot{\mathbf{q}} + \dot{J}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}$$

C-space and Task Space

- Our primary concern is the **end-effector pose** in task space

- IK solver needs to compute a ***C-space*** motion that does the right thing *in **task space***

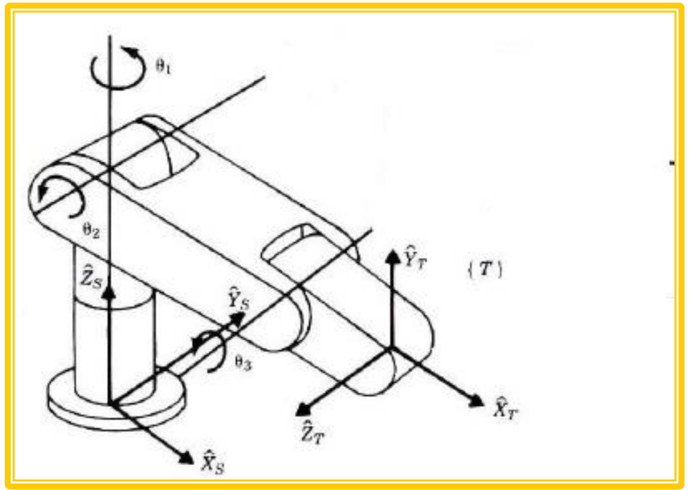
Inverse Kinematics at position levels

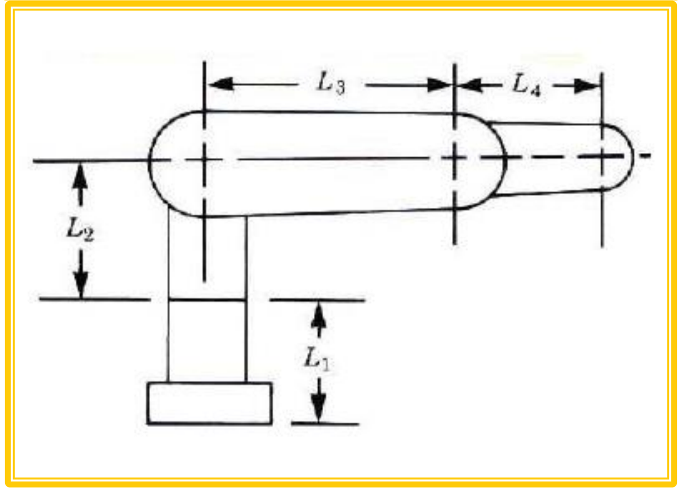
- Direct kinematics

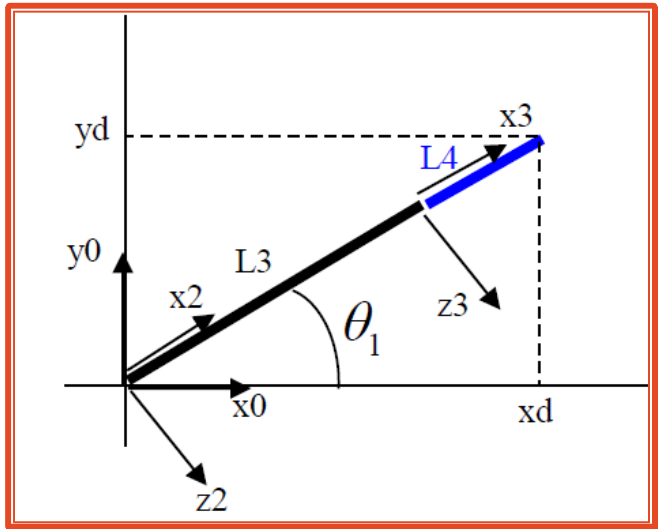
$$x = FK(\mathbf{q})$$

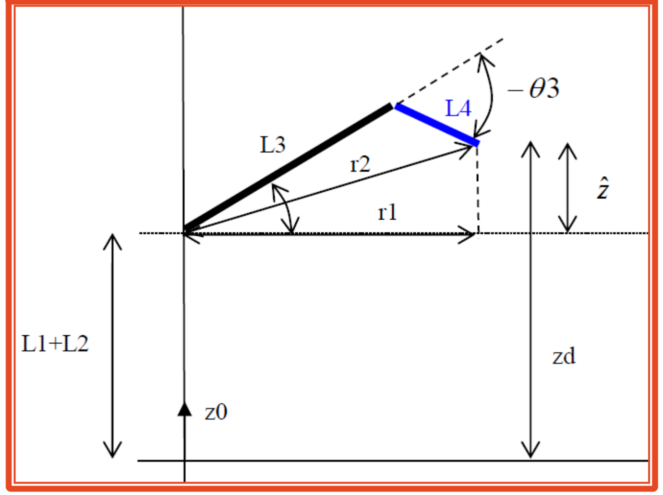
- IK solution
  - Analytical solution – robot geometry
  - Algebraic solution – homogeneous transformation matrices

Analytical solution







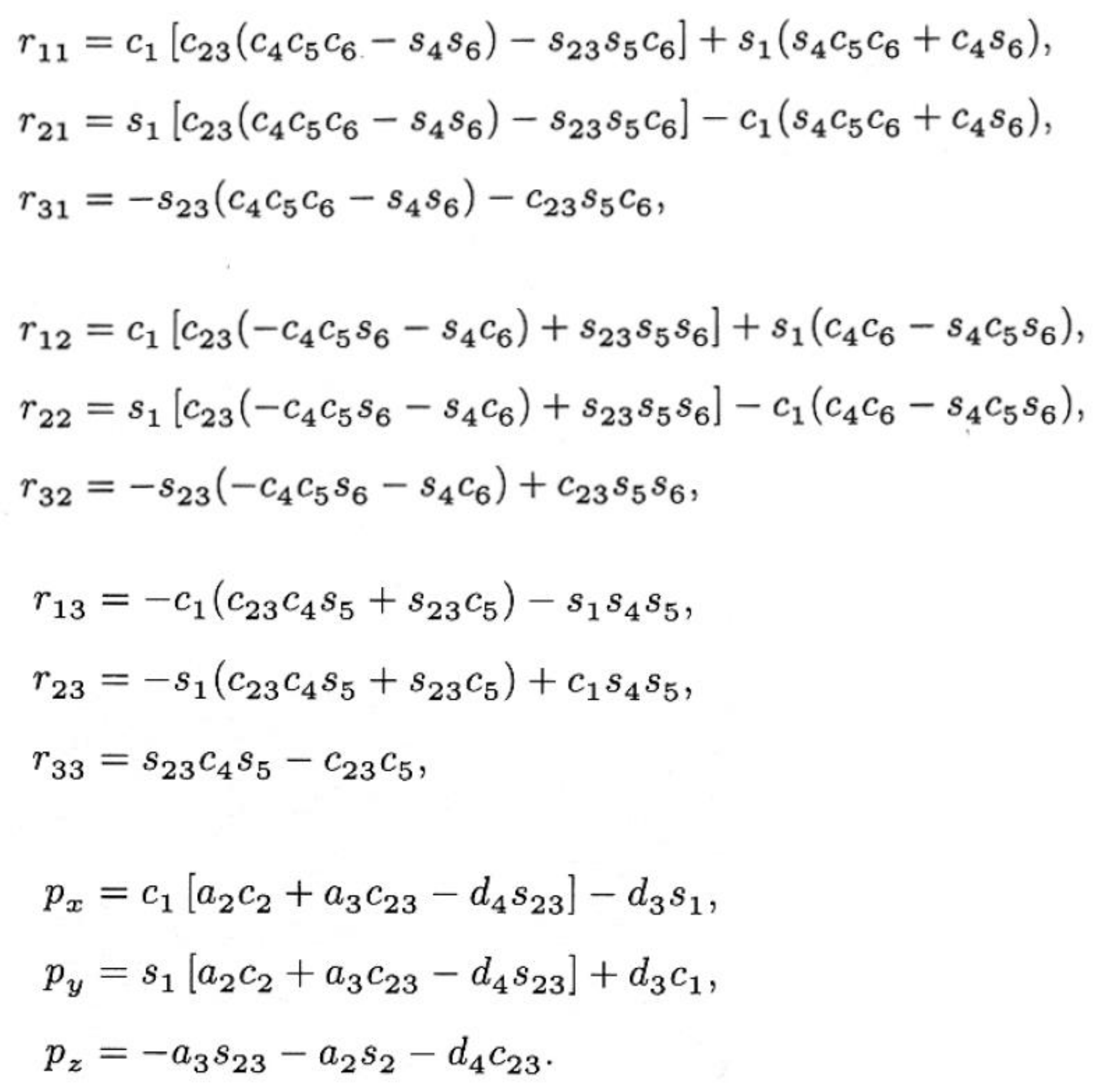


Algebraic Solution

$$ {}^{0}_{N}T = {}^{0}_{1}T \ldots {}^{N-1}_{\ \ N}T = \begin{bmatrix} \boxed{\begin{matrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{matrix}} & \boxed{\begin{matrix} p_x \\ p_y \\ p_z \end{matrix}} \\ \begin{matrix} 0 & \quad 0 & \quad 0 \end{matrix} & 1 \end{bmatrix} $$

Algebraic Solution

$$ {}^0_6T = {}^0_1T\,{}^1_2T\,{}^2_3T\,{}^3_4T\,{}^4_5T\,{}^5_6T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} $$



$$ r_{11} = c_1\left[c_{23}(c_4c_5c_6 - s_4s_6) - s_{23}s_5c_6\right] + s_1(s_4c_5c_6 + c_4s_6), $$

$$ r_{21} = s_1\left[c_{23}(c_4c_5c_6 - s_4s_6) - s_{23}s_5c_6\right] - c_1(s_4c_5c_6 + c_4s_6), $$

$$ r_{31} = -s_{23}(c_4c_5c_6 - s_4s_6) - c_{23}s_5c_6, $$

$$ r_{12} = c_1\left[c_{23}(-c_4c_5s_6 - s_4c_6) + s_{23}s_5s_6\right] + s_1(c_4c_6 - s_4c_5s_6), $$

$$ r_{22} = s_1\left[c_{23}(-c_4c_5s_6 - s_4c_6) + s_{23}s_5s_6\right] - c_1(c_4c_6 - s_4c_5s_6), $$

$$ r_{32} = -s_{23}(-c_4c_5s_6 - s_4c_6) + c_{23}s_5s_6, $$

$$ r_{13} = -c_1(c_{23}c_4s_5 + s_{23}c_5) - s_1s_4s_5, $$

$$ r_{23} = -s_1(c_{23}c_4s_5 + s_{23}c_5) + c_1s_4s_5, $$

$$ r_{33} = s_{23}c_4s_5 - c_{23}c_5, $$

$$ p_x = c_1\left[a_2c_2 + a_3c_{23} - d_4s_{23}\right] - d_3s_1, $$

$$ p_y = s_1\left[a_2c_2 + a_3c_{23} - d_4s_{23}\right] + d_3c_1, $$

$$ p_z = -a_3s_{23} - a_2s_2 - d_4c_{23}. $$

IK strategies

- Do not care about the redundant DOFs motion
  - Standard IK solvers, using pseudo-inverse
- Utilize redundant DOFs to handle additional constraints
  - Obstacle
- Utilize redundant DOFs to optimize performance
  - What are the performance indices?

Inverse Kinematics at velocity level

- First-order differential kinematics

$$\dot{x} = J(\mathbf{q})\dot{\mathbf{q}}$$

- IK solution
    - Inverse the Jacobian (non-redundant manipulator)
    - Pseudo-inverse of Jacobian

Jacobian

- Start with Forward Kinematics function

$$x = FK(q)$$

- Take the derivative with respect to time:

$$\frac{dx}{dt} = \frac{d[FK(q)]}{dt} = \frac{dFK(q)}{dq}\frac{dq}{dt}$$

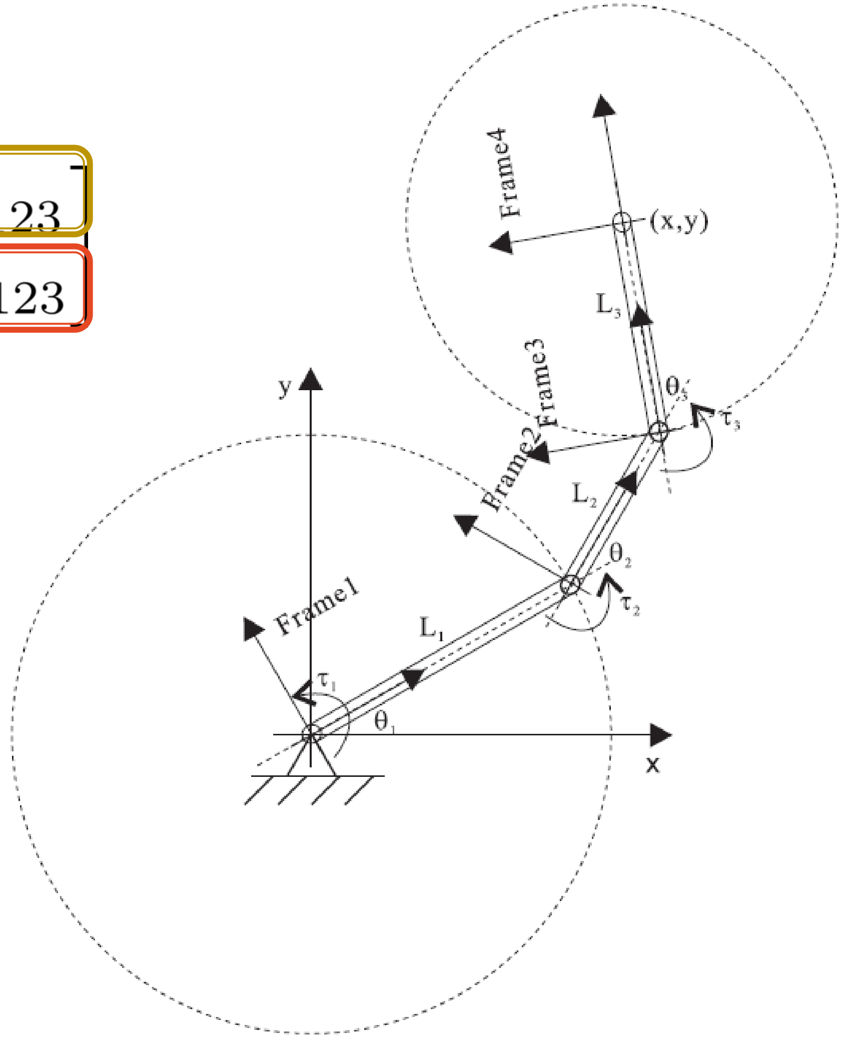
- Now we get the standard Jacobian equation:

$$\frac{dx}{dt} = J(q)\frac{dq}{dt} \quad \Rightarrow \quad \frac{dFK(q)}{dq} = J(q)$$

Example

$$\mathbf{T}(\theta_1, \theta_2, \theta 3) = \begin{bmatrix} c_{123} & -s_{123} & \boxed{L_1 c_1 + L_2 c_{12} + L_3 c_{123}} \\ s_{123} & c_{123} & \boxed{L_1 s_1 + L_2 s_{12} + L_3 s_{123}} \end{bmatrix}$$

$$\mathbf{J} = \begin{bmatrix} \frac{\partial x}{d\theta_1} & \frac{\partial x}{d\theta_2} & \frac{\partial x}{d\theta_3} \\ \\ \frac{\partial y}{d\theta_1} & \frac{\partial y}{d\theta_2} & \frac{\partial y}{d\theta_3} \end{bmatrix}$$



Inverting the Jacobian

- If N=M,
  - Jacobian is square → Standard matrix inverse
- If N>M ,
  - Pseudo-Inverse
  - Weighted Pseudo-Inverse
  - Damped least squares

Pseudo-Inverse

- Pseudo-inverse matrix
  - The unique matrix satisfying the Moore–Penrose conditions

$$\mathbf{J}\mathbf{J}^{\dagger}\mathbf{J} = \mathbf{J} \qquad (\mathbf{J}\mathbf{J}^{\dagger})^{T} = \mathbf{J}\mathbf{J}^{\dagger}$$

$$\mathbf{J}^{\dagger}\mathbf{J}\mathbf{J}^{\dagger} = \mathbf{J}^{\dagger} \qquad (\mathbf{J}^{\dagger}\mathbf{J})^{T} = \mathbf{J}^{\dagger}\mathbf{J}$$

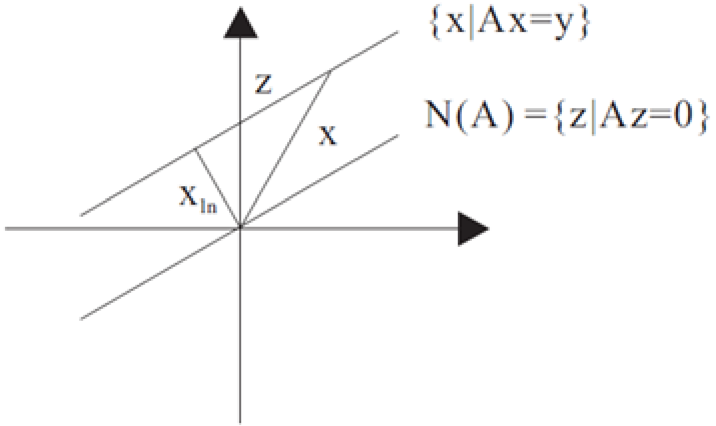
  - For redundant manipulator

$$\boxed{\mathbf{J}^{\dagger} = \mathbf{J}^{T}(\mathbf{J}\mathbf{J}^{T})^{-1}}$$

Pseudo-Inverse

- Pseudo-Inverse specifies a unique solution for inverse kinematics

- Implicitly, it performs the following optimization
  - Minimize $\frac{1}{2}\dot{\theta}^T\dot{\theta}$, given $\dot{x} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}}$



Weighted Pseudo-Inverse

- Multiply weighting coefficient matrix to Pseudo-Inverse Jacobian
  - $\dot{\mathbf{q}} = \mathbf{J}_w^\dagger(\mathbf{q})\dot{x}$ where $\mathbf{J}^\dagger = W^{-1}\mathbf{J}^T(\mathbf{J}W^{-1}\mathbf{J}^T)^{-1}$

- Optimization?
  - Minimize $\frac{1}{2}\dot{\theta}^T W \dot{\theta}$, given $\dot{x} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}}$

Weighted Pseudo-Inverse

- How to choose the weighting coefficient matrix?
  - W>0 and symmetric
  - Large weight → small joint velocity
  - Weights ~ inverse of the joint angle range

Singularity

- Singular Value Decomposition (SVD)

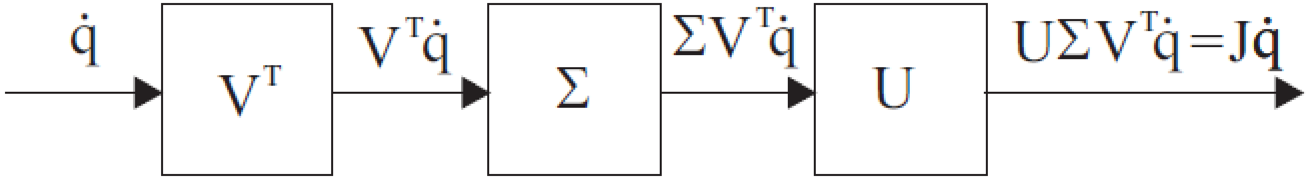
$$\mathbf{J}_{M\times N} = \mathbf{U}_{M\times M}\mathbf{\Sigma}_{M\times N}\mathbf{V}^T_{N\times N}$$

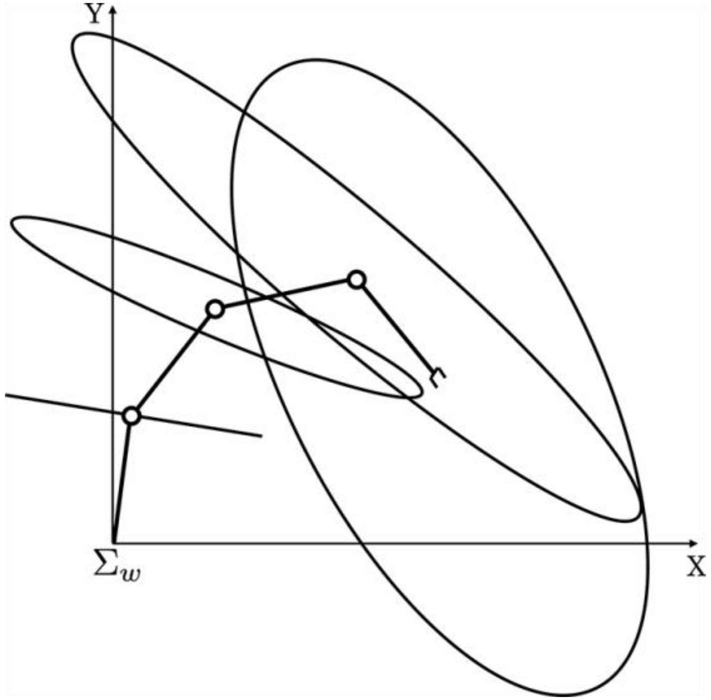
$$\Sigma = \left(\begin{array}{cccc|c} \sigma_1 & & & & \\ & \sigma_2 & & & \\ & & \ddots & & 0_{M\mathrm{x}(N-M)} \\ & & & \sigma_M & \end{array}\right)$$

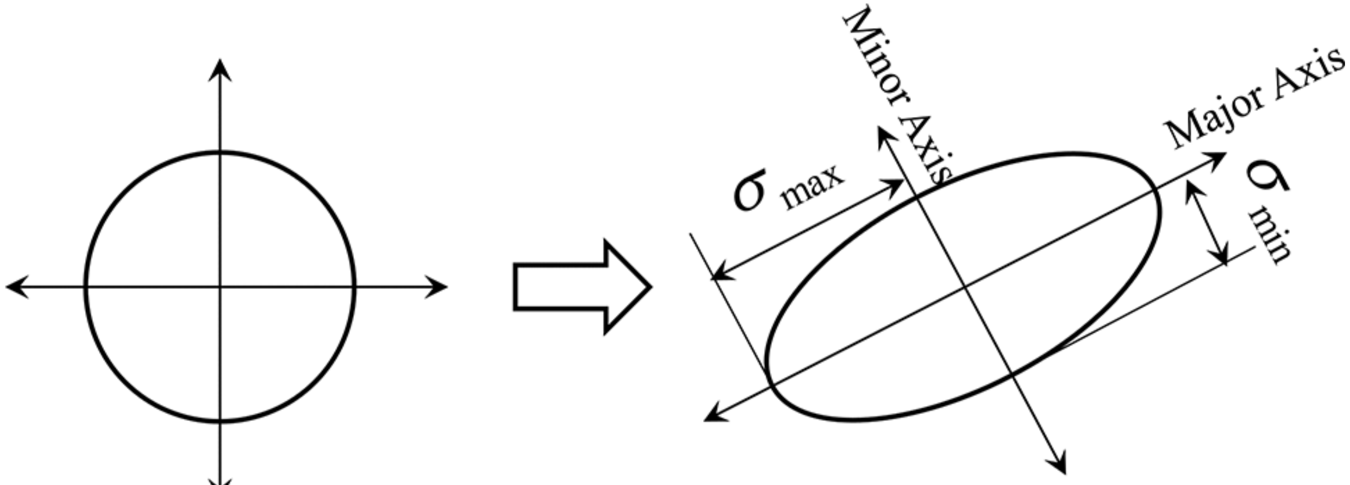
$$\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_\rho > 0, \quad \sigma_{\rho+1} = \ldots = \sigma_M = 0$$

singular values of J

Singularity







$$q^T q \le 1$$

$$V^T \left(JJ^T\right)^+ V \le 1$$

Singularity

$$J = U\Sigma V^T \quad \text{where } \Sigma = \left(\begin{array}{cccc|c} \sigma_1 & & & & \\ & \sigma_2 & & & 0_{\mathrm{Mx(N-M)}} \\ & & \ddots & & \\ & & & \sigma_{\mathrm{M}} & \end{array}\right)$$

$$J^\dagger = V\Sigma^\dagger U^T \quad \text{where } \Sigma^\dagger = \left(\begin{array}{ccc} \frac{1}{\sigma_1} & & \\ & \ddots & \\ & & \frac{1}{\sigma_\rho} \\ & & 0 \\ \hline & 0_{\mathrm{(N-M)xM}} & \end{array}\right)$$

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

$$\sigma_{\min}$$

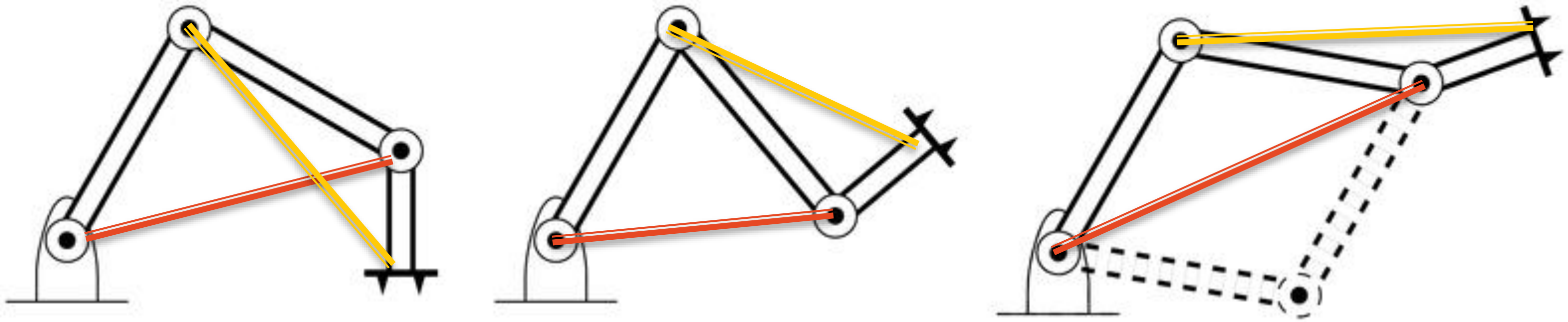
- Direction of velocity disadvantage
- Is it good enough?

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)



Singularity avoidance – Damped Least Squares

unconstrained minimization of a **suitable** objective function

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

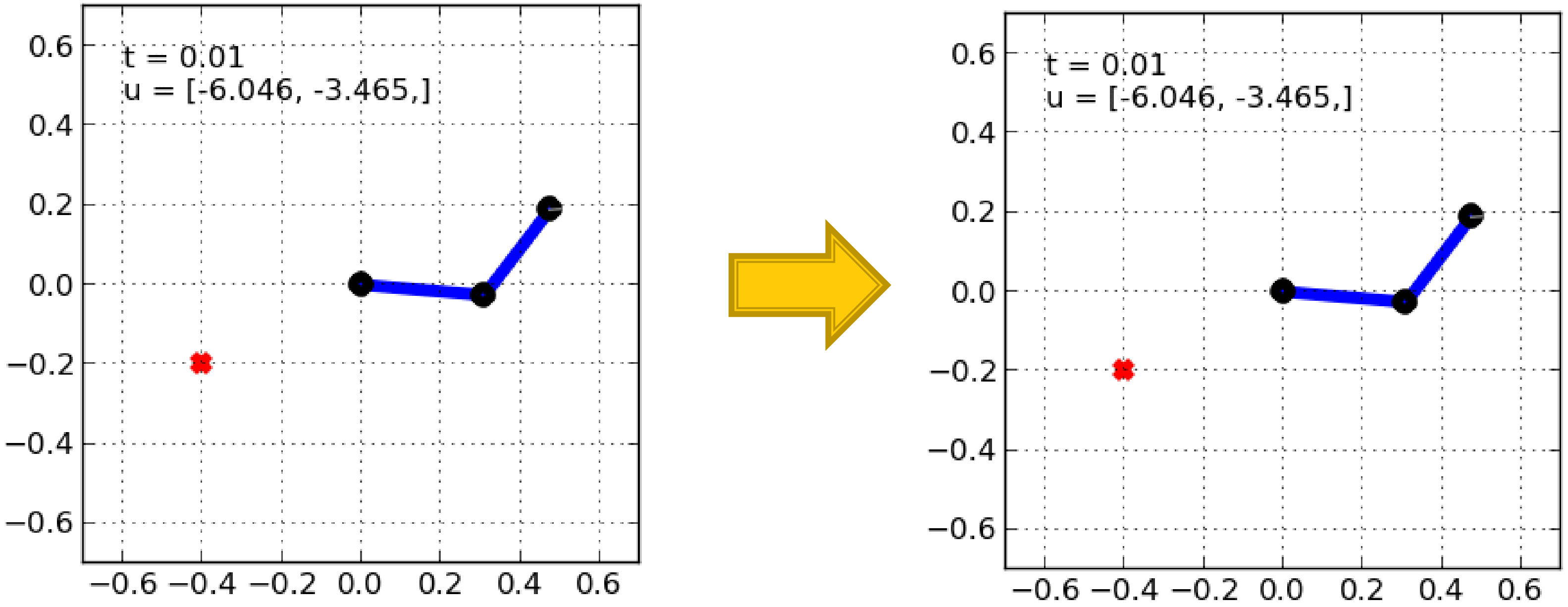
compromise between large joint velocity and task accuracy

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

Damped Least Squares

- The matrix will be invertible but this technique introduces a small inaccuracy



Damped Least Squares

- Induced error by damped least squares

$$\dot{e} = \mu^2 \left(JJ^T + \mu^2 I_M \right)^{-1} \dot{x} \quad \text{(as in N=M case)}$$

$$\text{using SVD of } J = U\Sigma V^T \Rightarrow J_{DLS} = V\Sigma_{DLS}U^T \text{ with } \Sigma_{DLS} = \left(\begin{array}{cc} \boxed{\text{diag}\{\frac{\sigma_i}{\sigma_i^2 + \mu^2}\}} & \\ \rho \times \rho & \text{diag}\{\frac{1}{\mu^2}\} \\ \hline 0_{(N-M)x\rho} & 0_{(N-M)x(N-\rho)} \end{array} \right)$$

- Choice of the damping factor $\mu^2(q) \geq 0$,
  - As a function the minimum singular value → measure of distance to singularity
  - Induce the damping only/mostly in the non-feasible direction of the task

Augmented Jacobian

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

$$\dot{x} = J(q)\dot{q}$$

  - Additional constraint

$$x_c = FK_c(q)$$

  - Secondary task

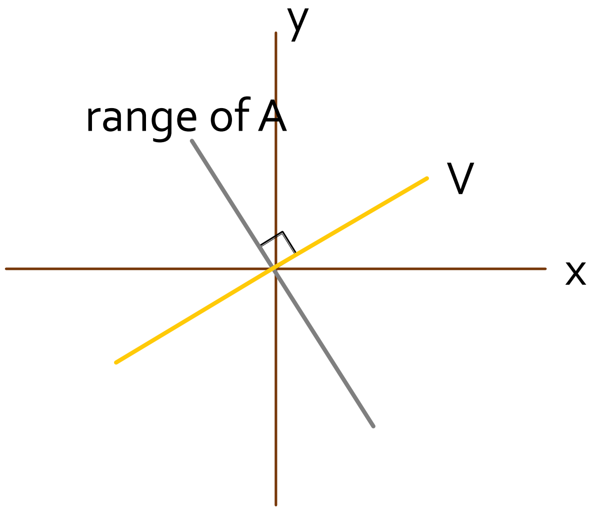
$$\dot{x}_c = J_c(q)\dot{q} \quad \text{where} \quad J_c(q) = \frac{\partial FK_c}{\partial q}$$

$$J_a(q) = \begin{bmatrix} J(q) \\ J_c(q) \end{bmatrix}$$

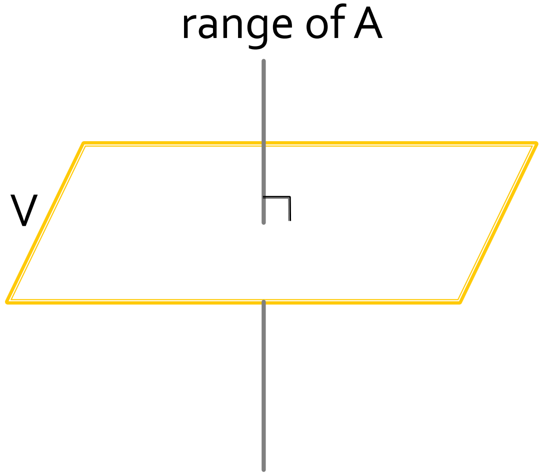
Full rank square Jacobian
Invertible!

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:

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

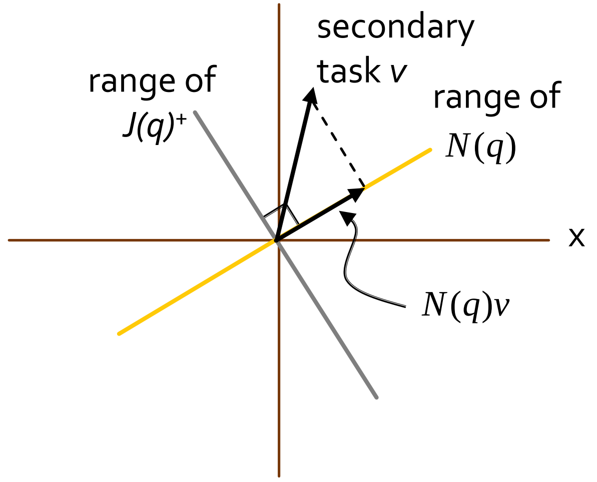
$$\mathbf{J}^{\dagger}\mathbf{J}\mathbf{J}^{\dagger} = \mathbf{J}^{\dagger}$$

-

The Null-space of Jacobian

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

$$\dot{q} = \underbrace{J(q)^{\dagger}\dot{x}}_{\boxed{\textbf{Primary task}}} + (\underbrace{I - J(q)^{\dagger}J(q))J_c(q)^{\dagger}\dot{x}_c}_{\boxed{\textbf{Secondary task}}}$$



The Null-space of Jacobian

- The null-space is often used to "push" IK solvers away from
  - Joint limits, obstacles
  - How to define the secondary task for the constraints in both task and joint space?

$$\dot{q} = \underbrace{J(q)^{\dagger}\dot{x}}_{\text{Primary task}} + \underbrace{(I - J(q)^{\dagger}J(q))\boxed{J_c(q)^{\dagger}\dot{x}_c}}_{\text{Secondary task}} \rightarrow \boxed{\dot{q}_c}$$

The Null-space of Jacobian

- For non-linear systems, magnitude differences in primary and secondary can cause numerical problems
  - One can overwhelm the other when you normalize later
  - Introduce a normalization factor

$$\dot{q} = \underbrace{J(q)^\dagger \dot{x}}_{\text{Primary task}} + \underbrace{\boxed{\beta}(I - J(q)^\dagger J(q))\dot{q}_c}_{\text{Secondary task}}$$

Conflicts with primary?

Recursive Null-space Projection

- What if you have three or more tasks?
  - The *i*-th task is:

$$T_i = J_i^{\dagger}(q)\dot{x}_i$$

  - The *i*-th null-space is:

$$N_i(q) = I - J_i^{\dagger}(q)J_i(q)$$

  - The recursive null-space formula is then:

$$\dot{q} = T_1 + N_1(T_2 + N_2(T3 + N_3(T_4 + \cdots N_{n-1}T_n)))$$

Inverse Kinematics at acceleration level

- Second-order differential kinematics

$$\ddot{x} = J(\mathbf{q})\ddot{\mathbf{q}} + \dot{J}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}$$

- IK solution

$$\ddot{q} = \mathbf{J}^{\dagger}(\mathbf{q})(\ddot{x} - \dot{\mathbf{J}}\dot{q}) + (\mathbf{I} - \mathbf{J}^{\dagger}\mathbf{J})\ddot{q}_0$$

  - $\ddot{q}_0 = 0$ is an arbitrary joint-space acceleration

Inverse Kinematics at acceleration level

- Choose $\ddot{q}_0 = 0$
- We have

$$\ddot{q} = \mathbf{J}^{\dagger}(\mathbf{q})(\ddot{x} - \dot{\mathbf{J}}\dot{q})$$

Minimum-norm acceleration solution

Reference

- Chapter 10 Redundant Robots in *Handbook of Robotics*, 2nd ed

End