ROBOT INTELLIGENCE LAB

Null-Space Projected IK

Exploiting Redundancy

Sungjoon Choi, Korea University

Inverse Kinematics

- Given current joint position $\mathbf{q}_{\text{curr}}$ and the goal position in the task space $\mathbf{x}_{\text{goal}}$, the next joint position is computed as:

$$\mathbf{q}_{\text{next}} \leftarrow \mathbf{q}_{\text{curr}} + \alpha\delta\mathbf{q}$$

$$\text{where } \delta\mathbf{q} = (J^T J + \lambda I)^{-1} J^T (\mathbf{x}_{\text{goal}} - \mathbf{x}_{\text{curr}})$$

Null Space Projection

ROBOT INTELLIGENCE LAB

- Suppose that we want to solve:

$$\min_{A\mathbf{x}=\mathbf{0}} \frac{1}{2}\|\mathbf{x}-\mathbf{z}\|_2^2$$

- The Lagrangian of this problem becomes:

$$\mathscr{L}(\mathbf{x}, \boldsymbol{\lambda}) = \frac{1}{2}\|\mathbf{x}-\mathbf{z}\|_2^2 + \boldsymbol{\lambda}^T A\mathbf{x}$$

$$\max_{\boldsymbol{\lambda}} \min_{\mathbf{x}} \frac{1}{2}\|\mathbf{x}-\mathbf{z}\|_2^2 + \boldsymbol{\lambda}^T A\mathbf{x}$$

- For the fixed $\boldsymbol{\lambda}$, the inner problem becomes:

$$\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{x}-\mathbf{z}\|_2^2 + \boldsymbol{\lambda}^T A\mathbf{x}$$

  with the optimal solution being $\mathbf{x}^* = \mathbf{z} - A^T\boldsymbol{\lambda}$.

- Substituting $\mathbf{x}_*$ into the Lagrangian,

$$\max_{\boldsymbol{\lambda}} -\frac{1}{2}\boldsymbol{\lambda}^T(AA^T)\boldsymbol{\lambda} + \boldsymbol{\lambda}^T A\mathbf{z}$$

  with the optimal solution being $\boldsymbol{\lambda}_* = (AA^T)^{-1}A\mathbf{z}$

- The optimal solution becomes $\mathbf{x}_* = \mathbf{z} - A^T\boldsymbol{\lambda}_* = \mathbf{z} - A^T(AA^T)^{-1}A\mathbf{z} = (I - A^T(AA^T)^{-1}A)\mathbf{z} = (I - A^\dagger A)\mathbf{z}$

Null Space Projected IK

- We can simply project $\delta\mathbf{q}$ into the null space by pre-multiplying

$$(I - J^{\dagger}J) \in \mathbb{R}^{N \times N}$$

where $J^{\dagger}$ is the pseudo-inverse of J and N is the number of revolute joints to control.

Null Space IK in Joint Space

- Given current joint position $\mathbf{q}_{\text{curr}}$, the goal position in the task space $\mathbf{x}_{\text{goal}}$, and the **null space goal joint position $\mathbf{q}_{\text{ns}}$** the next joint position is computed as:

$$\mathbf{q}_{\text{next}} \leftarrow \mathbf{q}_{\text{curr}} + \alpha\delta\mathbf{q} + \beta(I - J^{\dagger}J)(\mathbf{q}_{\text{ns}} - \mathbf{q}_{\text{curr}})$$

$$\text{where } \delta\mathbf{q} = (J^T J + \lambda I)^{-1} J^T(\mathbf{x}_{\text{goal}} - \mathbf{x}_{\text{curr}})$$

$$\mathbf{q}_{\text{next}} \leftarrow \mathbf{q}_{\text{curr}} + \alpha\underbrace{\boxed{(J^T J + \lambda I)^{-1} J^T(\mathbf{x}_{\text{goal}} - \mathbf{x}_{\text{curr}})}}_{\text{Original Task}} + \beta\underbrace{\boxed{(I - J^{\dagger}J)(\mathbf{q}_{\text{ns}} - \mathbf{q}_{\text{curr}})}}_{\text{Null Space Task}}$$

Null Space IK in Joint Space

ROBOT INTELLIGENCE LAB

```
% Get IK ingredients
[J_use,ik_err] = get_ik_ingredients(chain,...
    'joint_names_to_ctrl',joint_names_to_ctrl,...
    'joint_idxs_to_ctrl',joint_idxs_to_ctrl,...
    'joint_name_trgt',joint_name_trgt,...
    'T_trgt_goal',T_trgt_goal,'IK_P',IK_P,'IK_R',IK_R,...
    'p_err_weight',1.0,'w_err_weight',0.1,'ik_err_th',0.5);
```

Compute the Jacobian

```
% Compute dq from 'ik_err' and 'J_use'
dq = damped_ls(J_use,ik_err,...
    'lambda_rate',0.01,...
    'lambda_min',1e-6,...
    'step_size',0.1,...
    'dq_th',20*D2R);
```

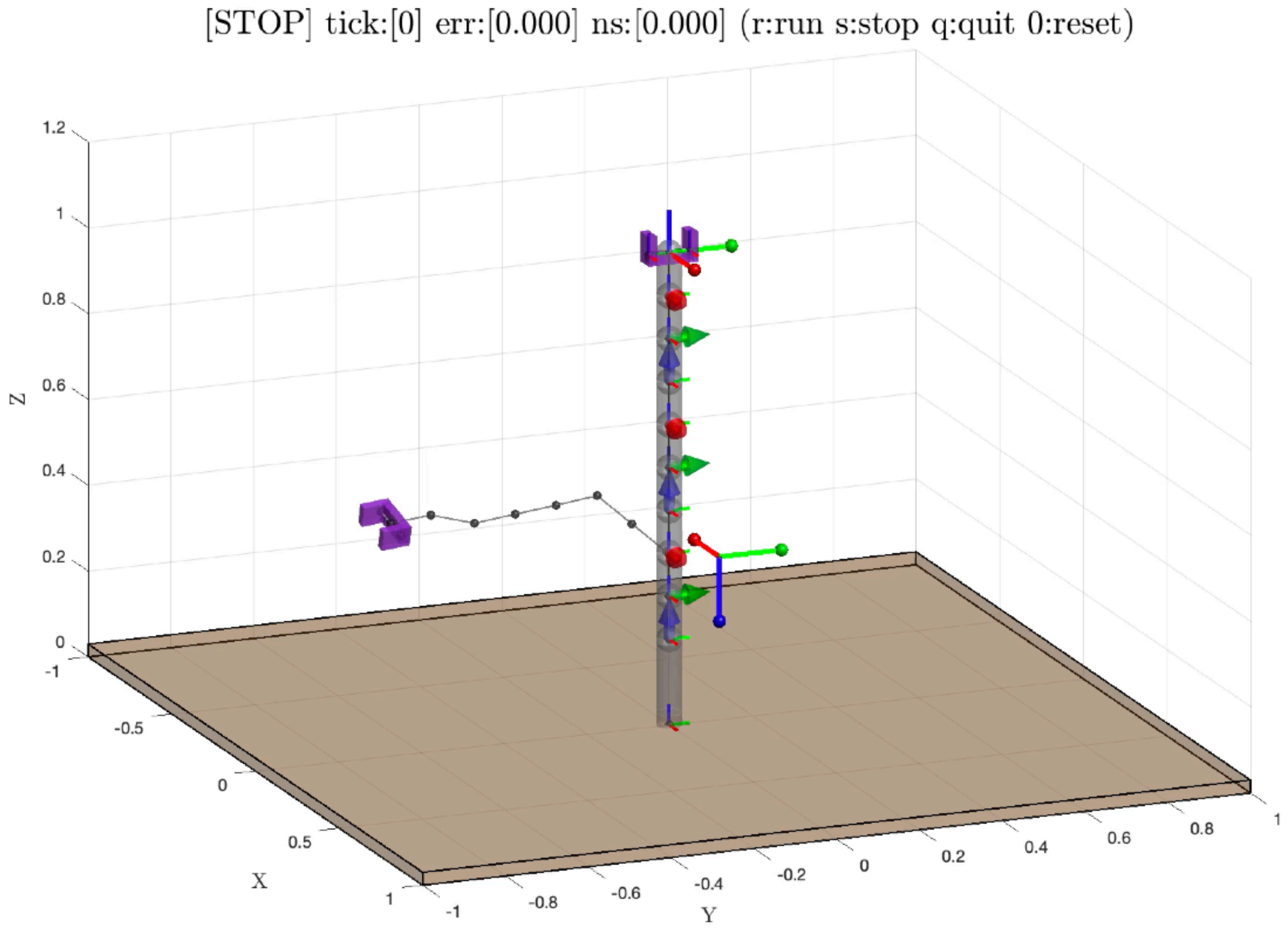
Original Task

```
% Once the error is small enough, do nullspace control
ik_err_avg = mean(abs(ik_err));
if (ik_err_avg < 1e-3)
    err_ns = (q_ns_des - q);
    ik_err_ns_avg = mean(abs(err_ns));
    nullspace_proj = (eye(n_ctrl,n_ctrl) - pinv(J_use)*J_use);
    dq_ns = nullspace_proj * err_ns;
    dq = dq + dq_ns;
    step_size = 0.1;
    dq = trim_scale(step_size*dq,10*D2R);
end
```

Null Space Task

```
% Update
q = q + dq;
chain = update_chain_q(chain,joint_names_to_ctrl,q);
```

Null Space IK in Joint Space



Null Space IK in Task Space

- Given current joint position $\mathbf{q}_{\text{curr}}$, the goal position in the **task space** $\mathbf{x}_{\text{goal}}$, and the **null space goal task position** $\mathbf{x}_{\text{ns}}$ the next joint position is computed as:

$$\mathbf{q}_{\text{next}} \leftarrow \mathbf{q}_{\text{curr}} + \alpha\delta\mathbf{q}_1 + \beta\delta\mathbf{q}_2$$

$$\delta\mathbf{q}_1 = (J_1^T J_1 + \lambda I)^{-1} J_1^T (\mathbf{x}_{\text{goal}} - \mathbf{x}_{\text{curr}})$$

$$\delta\mathbf{q}_2 = (I - J_1^{\dagger} J_1)(J_2^T J_2 + \lambda I)^{-1} J_2^T (\mathbf{x}_{\text{ns}} - \mathbf{x}_{\text{curr}})$$

where J_1 is the Jacobian for computing $\mathbf{x}_{\text{goal}}$ and J_2 is the Jacobian for computing $\mathbf{x}_{\text{ns}}$.

Null Space IK in Task Space

ROBOT INTELLIGENCE LAB

```
% Get IK ingredients
[J_use,ik_err] = get_ik_ingredients(chain,...
    'joint_names_to_ctrl',joint_names_to_ctrl,...
    'joint_idxs_to_ctrl',joint_idxs_to_ctrl,...
    'joint_name_trgt',joint_name_trgt,...
    'T_trgt_goal',T_trgt_goal,'IK_P',IK_P,'IK_R',IK_R,...
    'p_err_weight',1.0,'w_err_weight',0.1,'ik_err_th',0.5);
dq = damped_ls(J_use,ik_err,...
    'lambda_rate',0.01,'lambda_min',1e-6,...
    'step_size',0.1,'dq_th',20*D2R);
```

Original Task

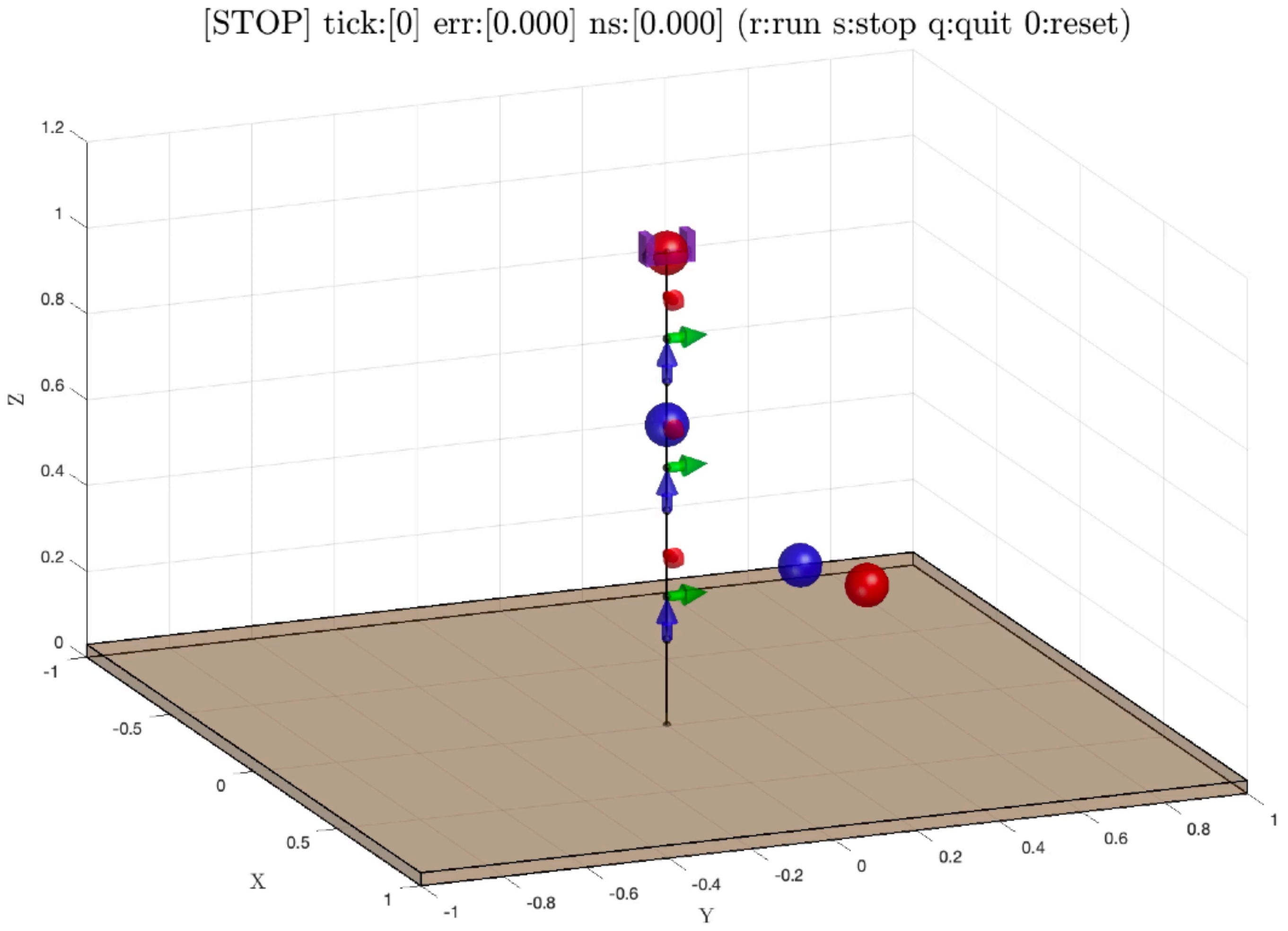
```
% Get IK ingredients for nullspace
[J_use_ns,ik_err_ns] = get_ik_ingredients(chain,...
    'joint_names_to_ctrl',joint_names_to_ctrl,...
    'joint_idxs_to_ctrl',joint_idxs_to_ctrl,...
    'joint_name_trgt',joint_name_trgt_ns,...
    'T_trgt_goal',T_trgt_goal_ns,'IK_P',IK_P_ns,'IK_R',IK_R_ns,...
    'p_err_weight',1.0,'w_err_weight',0.1,'ik_err_th',0.5);
dq_ns = damped_ls(J_use_ns,ik_err_ns,...
    'lambda_rate',0.01,'lambda_min',1e-6,...
    'step_size',0.1,'dq_th',10*D2R);
```

Secondary Task

```
% Once the error is small enough, do nullspace control
ik_err_avg = mean(abs(ik_err));
if (ik_err_avg < 1e-1)
    ik_err_ns_avg = mean(abs(ik_err_ns));
    dq = dq + (eye(n_ctrl,n_ctrl)-pinv(J_use)*J_use)*dq_ns;
    step_size = 1.0;
    dq = trim_scale(step_size*dq,10*D2R);
end
```

Null Space Projection

Null Space IK in Task Space



Thank You

ROBOT INTELLIGENCE LAB