Intelligent Robotics

Kinematics

Sungjoon Choi, Korea University

Differentiation of Rotation Matrix

ROBOT INTELLIGENCE LAB

$$\mathbf{p} = \mathbf{R}\bar{\mathbf{p}}$$

$$\dot{\mathbf{p}} = \dot{\mathbf{R}}\bar{\mathbf{p}}$$

$$\dot{\mathbf{p}} = \dot{\mathbf{R}}\mathbf{R}^T\mathbf{p}$$

$$\downarrow$$

$$\dot{\mathbf{p}} = \omega \times \mathbf{p}$$

$$\omega \times \mathbf{p} = \dot{\mathbf{R}}\mathbf{R}^T\mathbf{p}$$

$$\downarrow$$

$$\omega \times \mathbf{p} = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \times \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \begin{bmatrix} \omega_y p_z - \omega_z p_y \\ \omega_z p_x - \omega_x p_z \\ \omega_x p_y - \omega_y p_x \end{bmatrix}$$

Differentiation of Rotation Matrix

ROBOT INTELLIGENCE LAB

$$\omega \times \mathbf{p} = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \times \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \begin{bmatrix} \omega_y p_z - \omega_z p_y \\ \omega_z p_x - \omega_x p_z \\ \omega_x p_y - \omega_y p_x \end{bmatrix}$$

$$\omega \times \mathbf{p} = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \equiv \mathbf{S}\mathbf{p}$$

$\mathbf{S}^T = -\mathbf{S}$ skew symmetric matrix

$$\omega \times \mathbf{p} = \dot{\mathbf{R}}\mathbf{R}^T\mathbf{p}$$

$(\dot{\mathbf{R}}\mathbf{R}^T)^T = -\dot{\mathbf{R}}\mathbf{R}^T$ skew symmetric matrix

Differentiation of Rotation Matrix

ROBOT INTELLIGENCE LAB

$$\begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}^{\vee} = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$$

unskew: "wedge" ($\wedge$vee$) operation $\vee$

$$\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}^{\wedge} = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}.$$

skew: "hat" ($\wedge$wedge$) operation $\wedge$

$$\omega \times \mathbf{p} = \dot{\mathbf{R}}\mathbf{R}^T\mathbf{p}$$

$$\hat{\omega} = \dot{\mathbf{R}}\mathbf{R}^T \qquad \omega = (\dot{\mathbf{R}}\mathbf{R}^T)^{\vee}$$

Matrix Exponential

ROBOT INTELLIGENCE LAB

$$\hat{\omega} = \dot{\mathbf{R}}\mathbf{R}^T$$

$$\downarrow$$

$$\dot{\mathbf{R}} = \hat{\omega}\mathbf{R}$$ Basic Equation of Rotation

$$\downarrow$$

$$\mathbf{R}(t) = \mathbf{R}_0 e^{\hat{\omega}t}$$ Solution with **matrix exponential**

$$\boldsymbol{\omega} = \mathbf{a}\omega,\ \omega \equiv \|\boldsymbol{\omega}\|,\ \|\mathbf{a}\| = 1$$

$$\downarrow$$

$$e^{\hat{\omega}t} = \mathbf{I} + \hat{\mathbf{a}}\sin(\omega t) + \hat{\mathbf{a}}^2(1 - \cos(\omega t))$$ Rodrigues' formula

Rodrigues' Formula

ROBOT INTELLIGENCE LAB

$$e^{\hat{\omega} t} = \mathbf{I} + \hat{\mathbf{a}} \sin(\omega t) + \hat{\mathbf{a}}^2 (1 - \cos(\omega t))$$

$$\boxed{e^{\hat{\mathbf{a}} \theta}} = \mathbf{I} + \hat{\mathbf{a}} \sin(\theta) + \hat{\mathbf{a}}^2 (1 - \cos(\theta))$$

Rotation matrix

Rodrigues' formula gives us the **rotation matrix** from an angular **velocity vector**.

Matrix Logarithm

ROBOT INTELLIGENCE LAB

Angular velocity

$$\boxed{\boldsymbol{\omega}} = (\ln \boxed{\mathbf{R}})^{\vee}$$

Rotation matrix

$$(\ln \boldsymbol{R})^{\vee} = \begin{cases} [0\ 0\ 0]^T & (\text{if } \boldsymbol{R} = \boldsymbol{E}) \\ \frac{\pi}{2} \begin{bmatrix} r_{11}+1 \\ r_{22}+1 \\ r_{33}+1 \end{bmatrix} & (\text{else if } \boldsymbol{R} \text{ is diagonal}) \\ \theta \frac{\boldsymbol{l}}{\|\boldsymbol{l}\|} & (\text{otherwise}) \end{cases} \tag{2.39}$$

given

$$\boldsymbol{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix},$$

$$\boldsymbol{l} = \begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix},$$

$$\theta = \operatorname{atan2}(\|\boldsymbol{l}\|, r_{11} + r_{22} + r_{33} - 1)$$

Rotation Matrix <-> Angular Velocity

Log map (2.39)

$$\boldsymbol{\omega} = (\ln \mathbf{R})^{\vee}$$

Rotation Matrix

$$\mathbf{R}$$

Angular Velocity

$$\boldsymbol{\omega}$$

Rodrigues' formula (2.37)

$$q = \|\boldsymbol{\omega}\|, \ \mathbf{a} = \boldsymbol{\omega}/\|\boldsymbol{\omega}\|$$

$$\mathbf{R} = \mathbf{I} + \hat{\mathbf{a}}\sin(q) + \hat{\mathbf{a}}^2(1 - \cos(q))$$

Little Terminologies

ROBOT INTELLIGENCE LAB

- **Group**
  - A group is a set equipped with an operation that combines any two elements of the set to produce a third element of the set, in such a way that the operation is associative, an identity element exists, and every element has an inverse.
  - Ex) $\mathbb{Z}$: 1) $\forall a, b, c \in \mathbb{Z}$, $(a + b) + c = a + (b + c)$, 2) $\forall a \in \mathbb{Z}$ $\exists 0 \in \mathbb{Z}$ s.t. $a + 0 = a$, 3) $a + b = b + a = 0$ where b is the inverse element of a.
- **Vector space**
  - A set of which is closed under addition and scaling.
  - Ex) The set of real numbers $\mathbb{R}$ is a vector space.
- **Field**
  - A field is a set of which addition and multiplication are defined.
  - Ex) The set of real numbers $\mathbb{R}$ is also a field.

Little Terminologies

ROBOT INTELLIGENCE LAB

- **Field** vs. **Vector space**
  - The operations on a field $\mathbb{F}$ are
    - $+ : \mathbb{F} \times \mathbb{F} \to \mathbb{F}$
    - $\times : \mathbb{F} \times \mathbb{F} \to \mathbb{F}$
  - The operations on a vector space $\mathbb{V}$ over a field $\mathbb{F}$ are
    - $+ : \mathbb{V} \times \mathbb{V} \to \mathbb{V}$
    - $\times : \mathbb{F} \times \mathbb{V} \to \mathbb{V}$
  - For a field, any nonzero element $c \in \mathbb{F}$ has its inverse $c^{-1} \in \mathbb{F}$ s.t. $c \times c^{-1} = 1$, but there is no corresponding property among the vector space axioms.

Little Terminologies

ROBOT INTELLIGENCE LAB

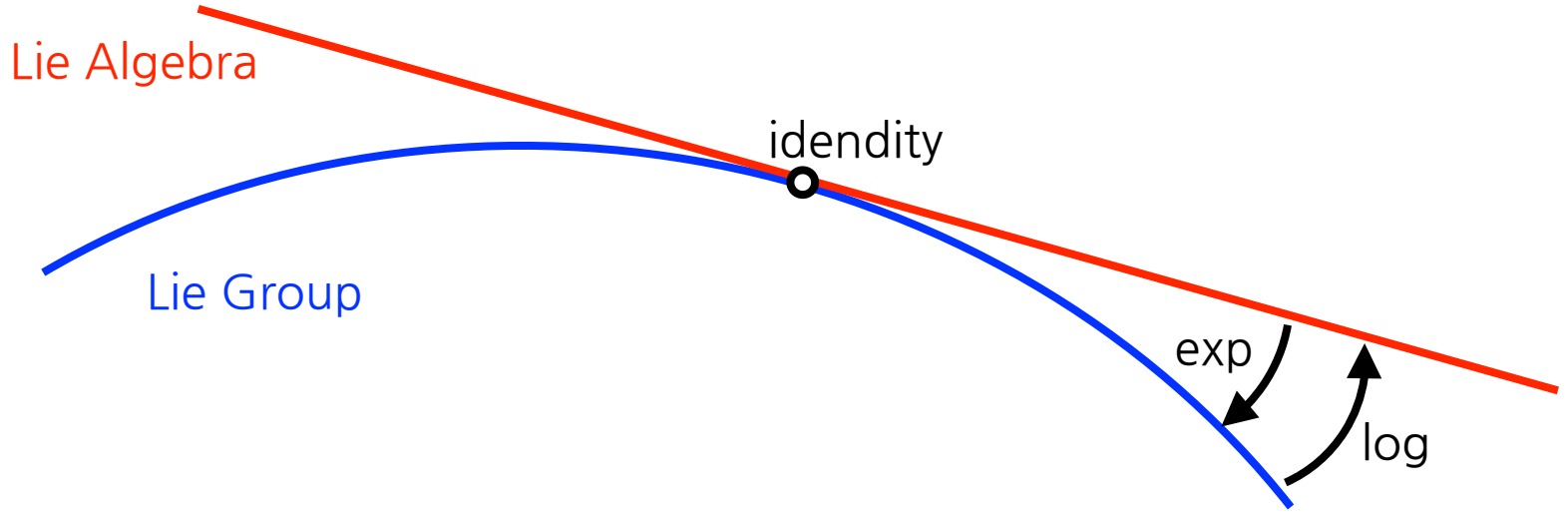
- **Algebra** (over a field)
  - An algebra is a vector space equipped with a bilinear product.

Algebra	vector space	bilinear operator	associativity	commutativity
complex numbers	$\mathbb{R}^2$	product of complex numbers $(a+ib)\cdot(c+id)$	Yes	Yes
cross product of 3D vectors	$\mathbb{R}^3$	cross product $\vec{a}\times\vec{b}$	No	No (anticommutative)
quaternions	$\mathbb{R}^4$	Hamilton product $(a+\vec{v})(b+\vec{w})$	Yes	No

- **Manifold**
  - A manifold is a topological space that locally resembles Euclidean space near each point. The mappings from a manifold the Euclidean space is called charts and the collection of charts is called an atlas.

Lie Group & Lie Algebra

ROBOT INTELLIGENCE LAB



- Definition: A Lie group is a smooth manifold that is also a group, such that the group operations mulitplication and inversion are smooth maps.
- Definition: The tangent space to a Lie group at the identity element is called the associated Lie algebra.
- The mapping from the Lie algebra to the Lie group is called the exponential map. Its inverse is called the logarithm map.

Lie Group & Lie Algebra

ROBOT INTELLIGENCE LAB

- **Lie Algebra,** $so(3)$
  - $\hat{\omega} \in so(3) = \{\hat{\omega} \,|\, \omega \in \mathbb{R}^3\}$
  - An algebra over a field K is a vector space V over K with multiplication on the space V.
  - One can also define the Lie bracket: $[\hat{w}, \hat{v}] = \hat{w}\hat{v} - \hat{v}\hat{w}$
- **Lie Group,** $SO(3)$
  - $R \in SO(3) = \{R \in \mathbb{R}^{3\times3} \,|\, R^T R = RR^T = I,\ \det(R) = 1\}$
  - More precisely, a **Lie Group** is a smooth manifold that is also a group, such that the group operations multiplication and inversion are smooth maps.
- The **Lie algebra** $so(3)$ is the tangent space at the identity of the rotation group $SO(3)$
  - $R(t) = R_0 e^{\hat{\omega}t}$ and for small dt, $R(dt) = R(0) + \hat{\omega}(0)dt$ which shows that $\hat{\omega} \in so(3)$ gives the first order apporximation of a rotation.
  - It also shows that $\hat{\omega}$ is the tangent space at the identity of $SO(3)$ $(R(0) = I)$.

Rotation Matrix <-> Angular Velocity


```
R = rpy2r(360*D2R*rand(3,1)); % random rotation matrix
w = r2w(R); % log map: w = wedge(ln(R))
R2 = rodrigues(uv(w),norm(w)); % exp map
```

```
function w = r2w(R)
%
% Get angular velocity [in radian] vector from a rotation matrix
% - This operation corresponds to w = wedge(ln(R)), a.k.a. a log map.
% - p.36 in 'Introduction to Humanoid Robotics' by Kajita et al.
%
% The reverse can be computed by 'rodrigues'
% ex)
% w = r2w(R);
% R = rodrigues(uv(w),norm(w));
%
el = [R(3,2)-R(2,3); R(1,3)-R(3,1); R(2,1)-R(1,2)];
norm_el = norm(el);
if norm_el > eps
    w = atan2(norm_el, trace(R)-1)/norm_el * el;
elseif R(1,1)>0 && R(2,2)>0 && R(3,3)>0
    w = [0, 0, 0]';
else
    w = pi/2*[R(1,1)+1; R(2,2)+1; R(3,3)+1];
end
```

```
function R = rodrigues(a,q)
%
% Compute the rotation matrix from an angular velocity vector
%  The reverse (R -> w) can be achieve by the w = r2w(R) function where w = a*q
%

if abs(norm(a)) < 1e-6
    R = eye(3,3);
    return;
end
if abs(norm(a)-1) > 1e-6
    warning('[rodrigues] a is not a unit vector (%.4e).\n',norm(a));
end

% Resolve some numerical issues
norm_a = norm(a);
a = a / norm_a;
q = q * norm_a;

% Get R
a_hat = skew(a);
R = eye(3,3) + a_hat*sin(q) + a_hat*a_hat*(1-cos(q));
```

Interpolate Two Rotation Matrices

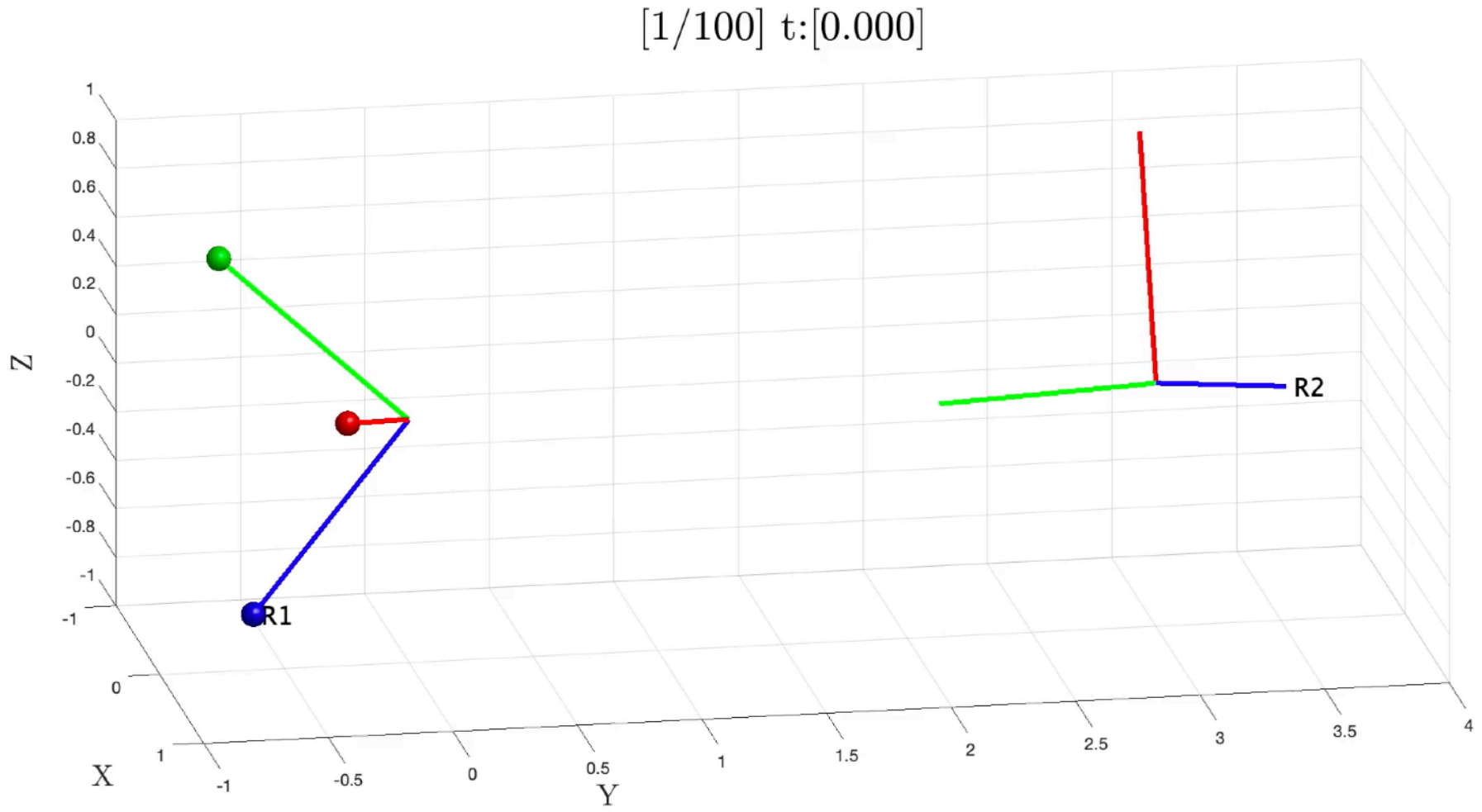
ROBOT INTELLIGENCE LAB

1. Get the rotation matrix which links the two matrices. $\boldsymbol{R} = \boldsymbol{R}_1^T \boldsymbol{R}_2$
2. Get the equivalent angular velocity vector from this rotation matrix. $\boldsymbol{\omega} = (\ln \boldsymbol{R})^\vee$
3. The angular velocity vector in world coordinates is: $\boldsymbol{R}_1 \boldsymbol{\omega}$
4. The interpolation becomes $\boldsymbol{R}(t) = \boldsymbol{R}_1 e^{\hat{\omega} t}, \quad t \in [0, 1].$

We can scale $\hat{\omega} \in so(3)$ (i.e., $0.5\hat{\omega}$), but we cannot scale $R \in SO(3)$.

Interpolate Two Rotation Matrices

ROBOT INTELLIGENCE LAB



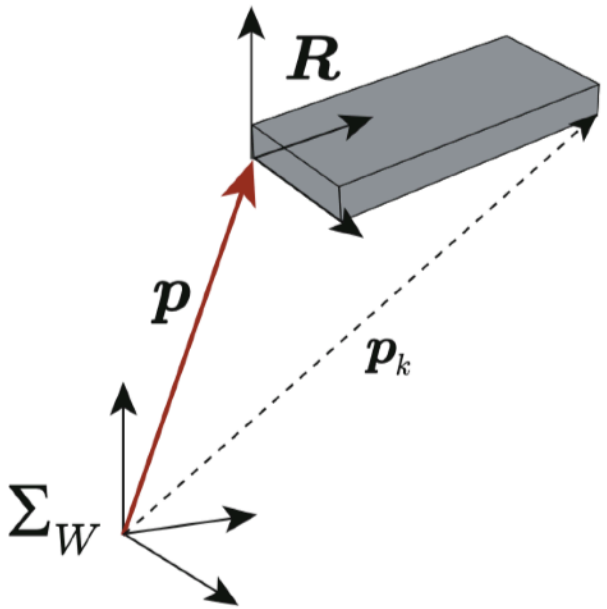
Interpolate Two Rotation Matrices

ROBOT INTELLIGENCE LAB

```
% Get two random rotational matrices
p1 = cv([0,0,0]);
p2 = cv([0,3,0]);
R1 = rpy2r(360*rand(3,1)*D2R);
R2 = rpy2r(360*rand(3,1)*D2R);
R_link = R1'*R2; % rotation matrix which links R1 and R2
w_link = r2w(R_link); % equivalent velocity vector fron the rotation matrix

x_traj = []; y_traj = []; z_traj = [];
all = 1.0; % axis line length
axis_info = [-1,+1,-1,+4,-1,+1];
ts = linspace(0,1,100); % t:[0,1]
for tick = 1:length(ts)

    % Interpolate R1 and R2
    t = ts(tick);
    p_t = (1-t)*p1 + t*p2;
    R_t = R1*rodrigues(w_link/norm(w_link),norm(w_link)*t); % interpolate

    % Append the trajectory
    x_traj = cat(1,x_traj, rv(p_t)+all*rv(R_t(:,1)));
    y_traj = cat(1,y_traj, rv(p_t)+all*rv(R_t(:,2)));
    z_traj = cat(1,z_traj, rv(p_t)+all*rv(R_t(:,3)));

    % Animate
    if (tick==1) || (mod(tick,2)==0) || (tick==length(ts))
        view_info = [80,16];
        fig = set_fig(figure(1),'pos',[0.0,0.5,0.5,0.4],...
            'view_info',view_info,'axis_info',axis_info,...
            'AXIS_EQUAL',1,'GRID_ON',1,'REMOVE_MENUBAR',1,'USE_DRAGZOOM',1,...
            'SET_CAMLIGHT',1,'SET_MATERIAL','METAL','SET_AXISLABEL',1,'afs',18); % make figure
        plot_T(pr2t(p1,R1),'fig_idx',1,'subfig_idx',1,...
            'PLOT_AXIS',1,'all',all,'alw',3,'alc','','PLOT_SPHERE',0,'text_str','R1','TEXT_AT_ZTIP',1); % R1
        plot_T(pr2t(p2,R2),'fig_idx',1,'subfig_idx',2,...
            'PLOT_AXIS',1,'all',all,'alw',3,'alc','','PLOT_SPHERE',0,'text_str','R2','TEXT_AT_ZTIP',1); % R2
        plot_T(pr2t(p_t,R_t),'fig_idx',1,'subfig_idx',3,...
            'PLOT_AXIS',1,'all',all,'alw',2,'alc','','PLOT_SPHERE',0,'PLOT_AXIS_TIP',1); % R_t
        plot_traj(x_traj,'fig_idx',1,'subfig_idx',2,'tlc','r','tlw',1,'tls','--');
        plot_traj(y_traj,'fig_idx',1,'subfig_idx',3,'tlc','g','tlw',1,'tls','--');
        plot_traj(z_traj,'fig_idx',1,'subfig_idx',4,'tlc','b','tlw',1,'tls','--');
        title_str = sprintf('[%d/%d] t:[%.3f]',tick,length(ts),t);
        plot_title(title_str,'tfs',25,'interpreter','latex');
        drawnow; if ~ishandle(fig), break; end
    end

    if tick == 1, pause; end
end
```

Velocity in Three Dimensional Space

ROBOT INTELLIGENCE LAB



$$p_k = p + R\bar{p}_k. \tag{2.40}$$

Let us assume that this object is tumbling through space combining both translational and rotational motion. The velocity of the point p_k can be derived by differentiating the previous equation with respect to time

$$\begin{aligned} \dot{p}_k &= \dot{p} + \dot{R}\bar{p}_k \\ &= v + \hat{\omega}R\bar{p}_k \\ &= v + \omega \times (R\bar{p}_k), \end{aligned} \tag{2.41}$$

Local coordinate

where v, ω have been defined as follows

$$v \equiv \dot{p} \tag{2.42}$$

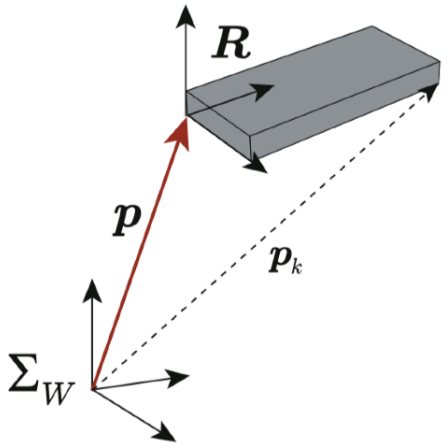
$$\omega \equiv (\dot{R}R^T)^\vee. \tag{2.43}$$

By making substitutions in (2.41) using (2.40) we obtain

$$\dot{p}_k = v + \omega \times (p_k - p). \tag{2.44}$$

Velocity in Three Dimensional Space

ROBOT INTELLIGENCE LAB



Let us assume that this object is tumbling through space combining both translational and rotational motion. The velocity of the point $\boldsymbol{p}_k$ can be derived by differentiating the previous equation with respect to time

$$\begin{aligned}\dot{\boldsymbol{p}}_k &= \dot{\boldsymbol{p}} + \dot{\boldsymbol{R}}\bar{\boldsymbol{p}}_k \\ &= \boldsymbol{v} + \hat{\boldsymbol{\omega}}\boldsymbol{R}\bar{\boldsymbol{p}}_k \\ &= \boldsymbol{v} + \boldsymbol{\omega} \times (\boldsymbol{R}\bar{\boldsymbol{p}}_k),\end{aligned} \tag{2.41}$$

where $\boldsymbol{v}, \boldsymbol{\omega}$ have been defined as follows

$$\boldsymbol{v} \equiv \dot{\boldsymbol{p}} \tag{2.42}$$

$$\boldsymbol{\omega} \equiv (\dot{\boldsymbol{R}}\boldsymbol{R}^T)^\vee. \tag{2.43}$$

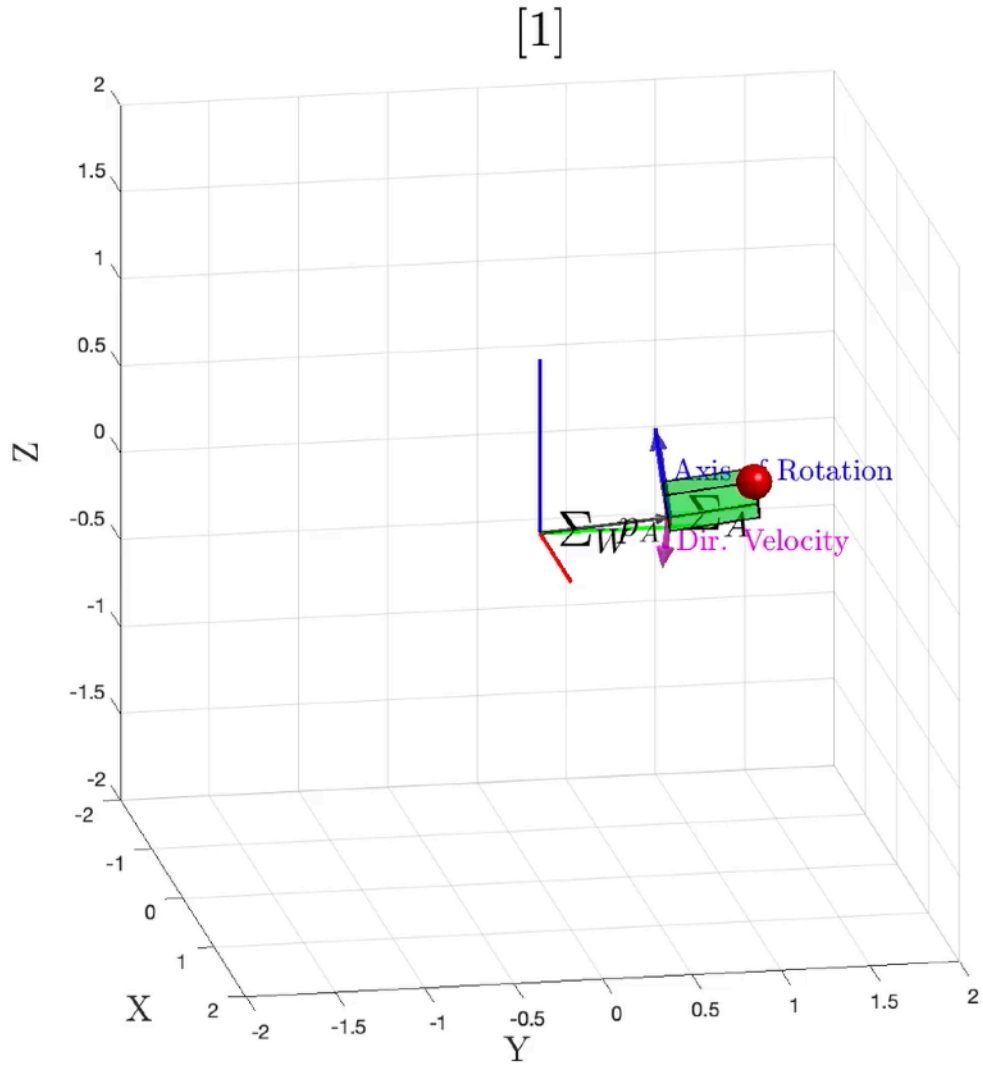
By making substitutions in (2.41) using (2.40) we obtain

$$\dot{\boldsymbol{p}}_k = \boldsymbol{v} + \boldsymbol{\omega} \times (\boldsymbol{p}_k - \boldsymbol{p}). \tag{2.44}$$

```
% Get position and orientation
[p_A_in_W,R_A_in_W] = t2pr(T_A_in_W);
% Update position
p_A_in_W = p_A_in_W + v_A_in_W*dt;
% Update rotaton
a_in_W = R_A_in_W(:,3); % z-axis to be the axis of rotation
R_rot = rodrigues(a_in_W,omega_A_in_W*dt); % rotate w.r.t z-axis of {A}
R_A_in_W = R_rot*R_A_in_W;
% Update coordinate transform
T_A_in_W = pr2t(p_A_in_W,R_A_in_W);

% Point in the world coordinates {W}
p_X_in_W = t2p(T_A_in_W*pr2t(p_X_in_A,''));
% Compute the linear velocity of p_X in {W}
w_A_in_W = r2w(rodrigues(a_in_W,omega_A_in_W)); % angular velocity vector of {A} (2.43)
p_X_dot_in_W = v_A_in_W + cross(w_A_in_W,p_X_in_W-p_A_in_W); % (2.44)

% Concat 'p_X_in_W'
p_X_in_W_traj = cat(1,p_X_in_W_traj,rv(p_X_in_W));
```

Velocity in Three Dimensional Space

ROBOT INTELLIGENCE LAB



Two Object Case

Next we will think about two objects moving in 3D space. Let us assume that we have been given the local coordinates of these objects as follows

$$\boldsymbol{T}_1 = \begin{bmatrix} \boldsymbol{R}_1 & \boldsymbol{p}_1 \\ 0\,0\,0 & 1 \end{bmatrix} \tag{2.45}$$

$$^1\boldsymbol{T}_2 = \begin{bmatrix} \boldsymbol{R}_d & \boldsymbol{p}_d \\ 0\,0\,0 & 1 \end{bmatrix}. \tag{2.46}$$

The position and attitude of the second object is given to you relative to the first object as $^1\boldsymbol{T}_2$. So,

$$\begin{aligned} \boldsymbol{T}_2 &= \boldsymbol{T}_1\, {}^1\boldsymbol{T}_2 \\ &= \begin{bmatrix} \boldsymbol{R}_1 & \boldsymbol{p}_1 \\ 0\,0\,0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{R}_d & \boldsymbol{p}_d \\ 0\,0\,0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} (\boldsymbol{R}_1\boldsymbol{R}_d) & (\boldsymbol{p}_1 + \boldsymbol{R}_1\boldsymbol{p}_d) \\ 0\,0\,0 & 1 \end{bmatrix}. \end{aligned} \tag{2.47}$$

Therefore the position and attitude of the second object in world coordinates is

$$\boldsymbol{p}_2 = \boldsymbol{p}_1 + \boldsymbol{R}_1\boldsymbol{p}_d \tag{2.48}$$

$$\boldsymbol{R}_2 = \boldsymbol{R}_1\boldsymbol{R}_d. \tag{2.49}$$

The linear velocity of the second object can be obtained by taking the time derivative of (2.48).

$$\begin{aligned} \boldsymbol{v}_2 &= \frac{d}{dt}(\boldsymbol{p}_1 + \boldsymbol{R}_1\boldsymbol{p}_d) \\ &= \dot{\boldsymbol{p}}_1 + \dot{\boldsymbol{R}}_1\boldsymbol{p}_d + \boldsymbol{R}_1\dot{\boldsymbol{p}}_d \\ &= \boldsymbol{v}_1 + \widehat{\boldsymbol{\omega}}_1\boldsymbol{R}_1\boldsymbol{p}_d + \boldsymbol{R}_1\boldsymbol{v}_d \\ &= \boldsymbol{v}_1 + \boldsymbol{\omega}_1 \times (\boldsymbol{R}_1\boldsymbol{p}_d) + \boldsymbol{R}_1\boldsymbol{v}_d \end{aligned}$$

where, $\boldsymbol{v}_1 \equiv \dot{\boldsymbol{p}}_1$, $\boldsymbol{v}_d \equiv \dot{\boldsymbol{p}}_d$.

Here, using (2.48) and rearranging its elements we obtain

$$\boldsymbol{v}_2 = \boldsymbol{v}_1 + \boldsymbol{R}_1\boldsymbol{v}_d + \boldsymbol{\omega}_1 \times (\boldsymbol{p}_2 - \boldsymbol{p}_1). \tag{2.50}$$

The angular velocity of the second object can be calculated from (2.49) as follows

$$\begin{aligned} \widehat{\boldsymbol{\omega}}_2 &= \dot{\boldsymbol{R}}_2\boldsymbol{R}_2^T \\ &= \frac{d}{dt}(\boldsymbol{R}_1\boldsymbol{R}_d)\boldsymbol{R}_2^T \\ &= (\dot{\boldsymbol{R}}_1\boldsymbol{R}_d + \boldsymbol{R}_1\dot{\boldsymbol{R}}_d)\boldsymbol{R}_2^T \\ &= (\widehat{\boldsymbol{\omega}}_1\boldsymbol{R}_1\boldsymbol{R}_d + \boldsymbol{R}_1\widehat{\boldsymbol{\omega}}_d\boldsymbol{R}_d)\boldsymbol{R}_2^T \\ &= \widehat{\boldsymbol{\omega}}_1 + \boldsymbol{R}_1\widehat{\boldsymbol{\omega}}_d\boldsymbol{R}_d\boldsymbol{R}_d^T\boldsymbol{R}_1^T \\ &= \widehat{\boldsymbol{\omega}}_1 + \boldsymbol{R}_1\widehat{\boldsymbol{\omega}}_d\boldsymbol{R}_1^T \end{aligned}$$

where $\boldsymbol{\omega}_1 \equiv (\dot{\boldsymbol{R}}_1\boldsymbol{R}_1^T)^\vee$, $\boldsymbol{\omega}_d \equiv (\dot{\boldsymbol{R}}_d\boldsymbol{R}_d^T)^\vee$.

From Fig. 2.12 we can see that $(\boldsymbol{R}\boldsymbol{\omega})^\wedge = \boldsymbol{R}\widehat{\boldsymbol{\omega}}\boldsymbol{R}^T$, so we get

$$\widehat{\boldsymbol{\omega}}_2 = \widehat{\boldsymbol{\omega}}_1 + (\boldsymbol{R}_1\boldsymbol{\omega}_d)^\wedge.$$

By applying $\vee$ to both sides of this equation we get,

$$\boldsymbol{\omega}_2 = \boldsymbol{\omega}_1 + \boldsymbol{R}_1\boldsymbol{\omega}_d. \tag{2.51}$$

Two Object Case

Next we will think about two objects moving in 3D space. Let us assume that we have been given the local coordinates of these objects as follows

$$\boldsymbol{T}_1 = \begin{bmatrix} \boldsymbol{R}_1 & \boldsymbol{p}_1 \\ 0\,0\,0 & 1 \end{bmatrix} \tag{2.45}$$

Let's take a moment to think about (2.22) in Section 2.2.4,

$$\boldsymbol{R}(\boldsymbol{\omega} \times \boldsymbol{p}) = (\boldsymbol{R}\boldsymbol{\omega}) \times (\boldsymbol{R}\boldsymbol{p}) \qquad \ldots (a)$$

The left side of this equation can be rewritten as

$$\begin{aligned} \boldsymbol{R}(\boldsymbol{\omega} \times \boldsymbol{p}) &= \boldsymbol{R}\widehat{\boldsymbol{\omega}}\boldsymbol{p} \\ &= \boldsymbol{R}\widehat{\boldsymbol{\omega}}\boldsymbol{R}^T\boldsymbol{R}\boldsymbol{p} \\ &= (\boldsymbol{R}\widehat{\boldsymbol{\omega}}\boldsymbol{R}^T)(\boldsymbol{R}\boldsymbol{p}). \qquad \ldots (b) \end{aligned}$$

Comparing equation (a) and equation (b) we can see that

$$(\boldsymbol{R}\boldsymbol{\omega})^\wedge = \boldsymbol{R}\widehat{\boldsymbol{\omega}}\boldsymbol{R}^T.$$

Fig. 2.12 Coordinate Transformation and the Angular Velocity Vector

The linear velocity of the second object can be obtained by taking the time derivative of (2.48).

$$\begin{aligned} \boldsymbol{v}_2 &= \frac{d}{dt}(\boldsymbol{p}_1 + \boldsymbol{R}_1\boldsymbol{p}_d) \\ &= \dot{\boldsymbol{p}}_1 + \dot{\boldsymbol{R}}_1\boldsymbol{p}_d + \boldsymbol{R}_1\dot{\boldsymbol{p}}_d \\ &= \boldsymbol{v}_1 + \widehat{\boldsymbol{\omega}}_1\boldsymbol{R}_1\boldsymbol{p}_d + \boldsymbol{R}_1\boldsymbol{v}_d \\ &= \boldsymbol{v}_1 + \boldsymbol{\omega}_1 \times (\boldsymbol{R}_1\boldsymbol{p}_d) + \boldsymbol{R}_1\boldsymbol{v}_d \end{aligned}$$

where, $\boldsymbol{v}_1 \equiv \dot{\boldsymbol{p}}_1$, $\boldsymbol{v}_d \equiv \dot{\boldsymbol{p}}_d$.

Here, using (2.48) and rearranging its elements we obtain

$$\boldsymbol{v}_2 = \boldsymbol{v}_1 + \boldsymbol{R}_1\boldsymbol{v}_d + \boldsymbol{\omega}_1 \times (\boldsymbol{p}_2 - \boldsymbol{p}_1). \tag{2.50}$$

The angular velocity of the second object can be calculated from (2.49) as follows

$$\begin{aligned} \widehat{\boldsymbol{\omega}}_2 &= \dot{\boldsymbol{R}}_2\boldsymbol{R}_2^T \\ &= \frac{d}{dt}(\boldsymbol{R}_1\boldsymbol{R}_d)\boldsymbol{R}_2^T \\ &= (\dot{\boldsymbol{R}}_1\boldsymbol{R}_d + \boldsymbol{R}_1\dot{\boldsymbol{R}}_d)\boldsymbol{R}_2^T \\ &= (\widehat{\boldsymbol{\omega}}_1\boldsymbol{R}_1\boldsymbol{R}_d + \boldsymbol{R}_1\widehat{\boldsymbol{\omega}}_d\boldsymbol{R}_d)\boldsymbol{R}_2^T \\ &= \widehat{\boldsymbol{\omega}}_1 + \boldsymbol{R}_1\widehat{\boldsymbol{\omega}}_d\boldsymbol{R}_d\boldsymbol{R}_d^T\boldsymbol{R}_1^T \\ &= \widehat{\boldsymbol{\omega}}_1 + \boldsymbol{R}_1\widehat{\boldsymbol{\omega}}_d\boldsymbol{R}_1^T \end{aligned}$$

where $\boldsymbol{\omega}_1 \equiv (\dot{\boldsymbol{R}}_1\boldsymbol{R}_1^T)^\vee$, $\boldsymbol{\omega}_d \equiv (\dot{\boldsymbol{R}}_d\boldsymbol{R}_d^T)^\vee$.

From Fig. 2.12 we can see that $(\boldsymbol{R}\boldsymbol{\omega})^\wedge = \boldsymbol{R}\widehat{\boldsymbol{\omega}}\boldsymbol{R}^T$, so we get

$$\widehat{\boldsymbol{\omega}}_2 = \widehat{\boldsymbol{\omega}}_1 + (\boldsymbol{R}_1\boldsymbol{\omega}_d)^\wedge.$$

By applying $\vee$ to both sides of this equation we get,

$$\boldsymbol{\omega}_2 = \boldsymbol{\omega}_1 + \boldsymbol{R}_1\boldsymbol{\omega}_d. \tag{2.51}$$

Two Object Case

ROBOT INTELLIGENCE LAB

In summary, we first defined the position and attitude of object 1 and object 2 as $(\boldsymbol{p}_1, \boldsymbol{R}_1)$, $(\boldsymbol{p}_2, \boldsymbol{R}_2)$. If the velocity of object 1 is $(\boldsymbol{v}_1, \boldsymbol{\omega}_1)$ and the velocity of object 2 relative to object 1 is $(\boldsymbol{v}_d, \boldsymbol{\omega}_d)$, the velocity of object 2 will be given as follows[8]

$$\boldsymbol{v}_2 = \boldsymbol{v}_1 + \boldsymbol{R}_1\boldsymbol{v}_d + \boldsymbol{\omega}_1 \times (\boldsymbol{p}_2 - \boldsymbol{p}_1) \tag{2.52}$$

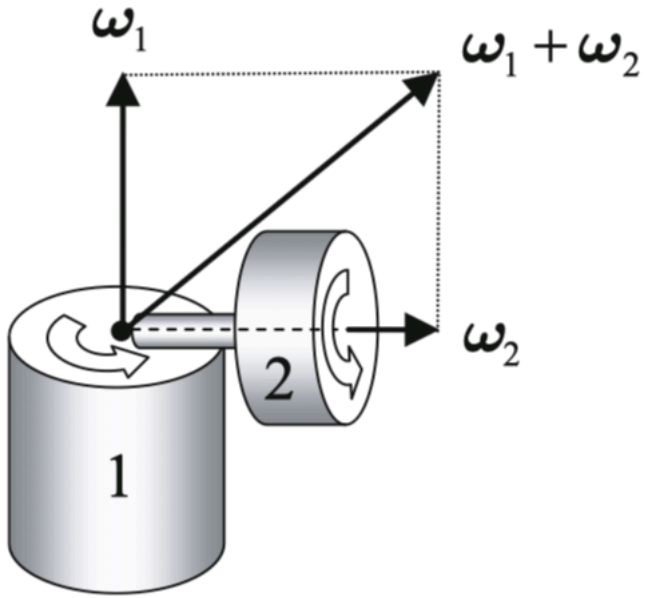
$$\boldsymbol{\omega}_2 = \boldsymbol{\omega}_1 + \boldsymbol{R}_1\boldsymbol{\omega}_d. \tag{2.53}$$

Here $\boldsymbol{R}_1\boldsymbol{v}_d$ and $\boldsymbol{R}_1\boldsymbol{\omega}_d$ describe the relative velocity between the objects in world coordinates. If we replace these using ${}^W\boldsymbol{v}_d, {}^W\boldsymbol{\omega}_d$ we get,

$$\boldsymbol{v}_2 = \boldsymbol{v}_1 + {}^W\boldsymbol{v}_d + \boldsymbol{\omega}_1 \times (\boldsymbol{p}_2 - \boldsymbol{p}_1) \tag{2.54}$$

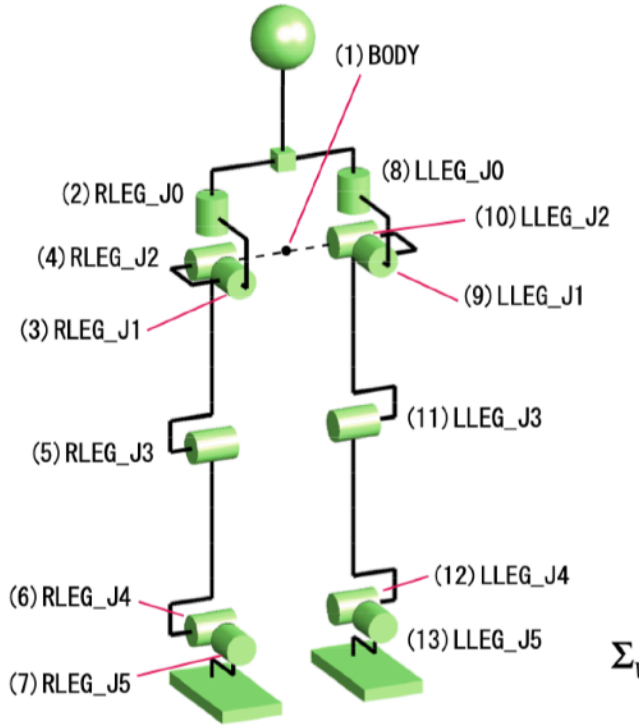
$$\boldsymbol{\omega}_2 = \boldsymbol{\omega}_1 + {}^W\boldsymbol{\omega}_d. \tag{2.55}$$

Therefore barring the fact that we get an influence of the rotational velocity the third element of (2.54), the velocity of the object in world coordinates is simply a sum.

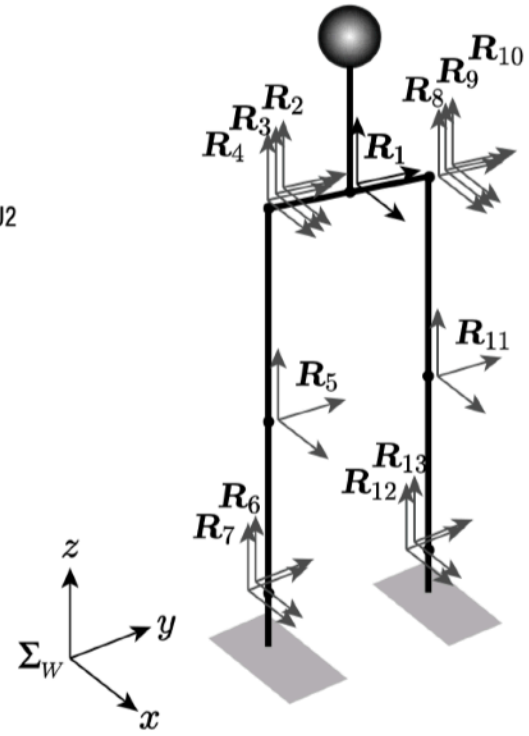


Kinematics of a Humanoid Robot

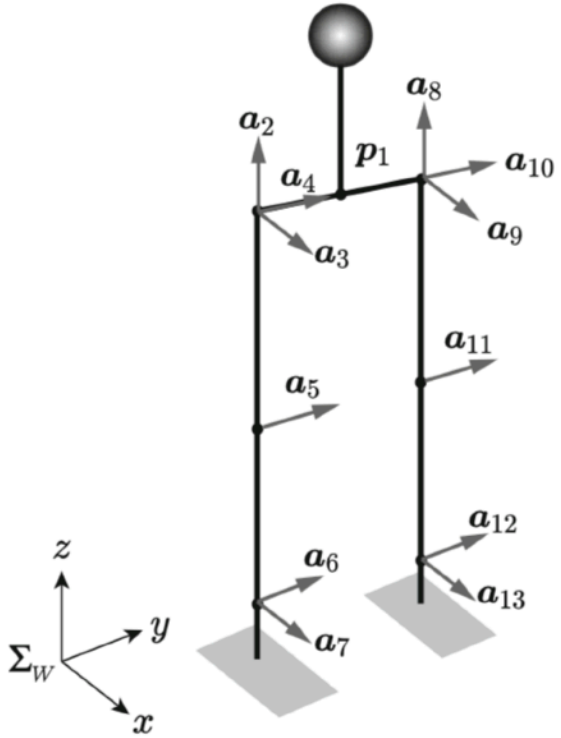
ROBOT INTELLIGENCE LAB



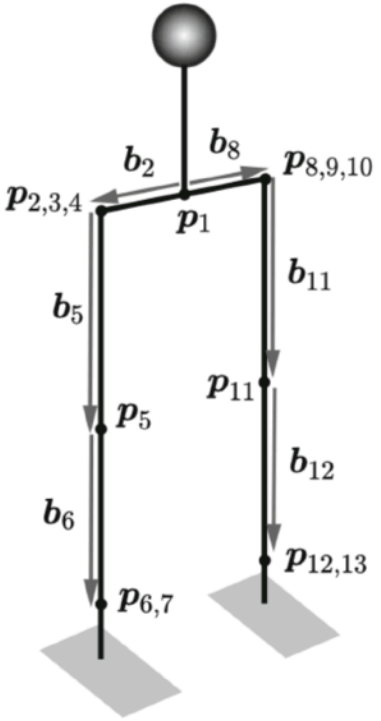
Structure of a robot



Rotation matrices



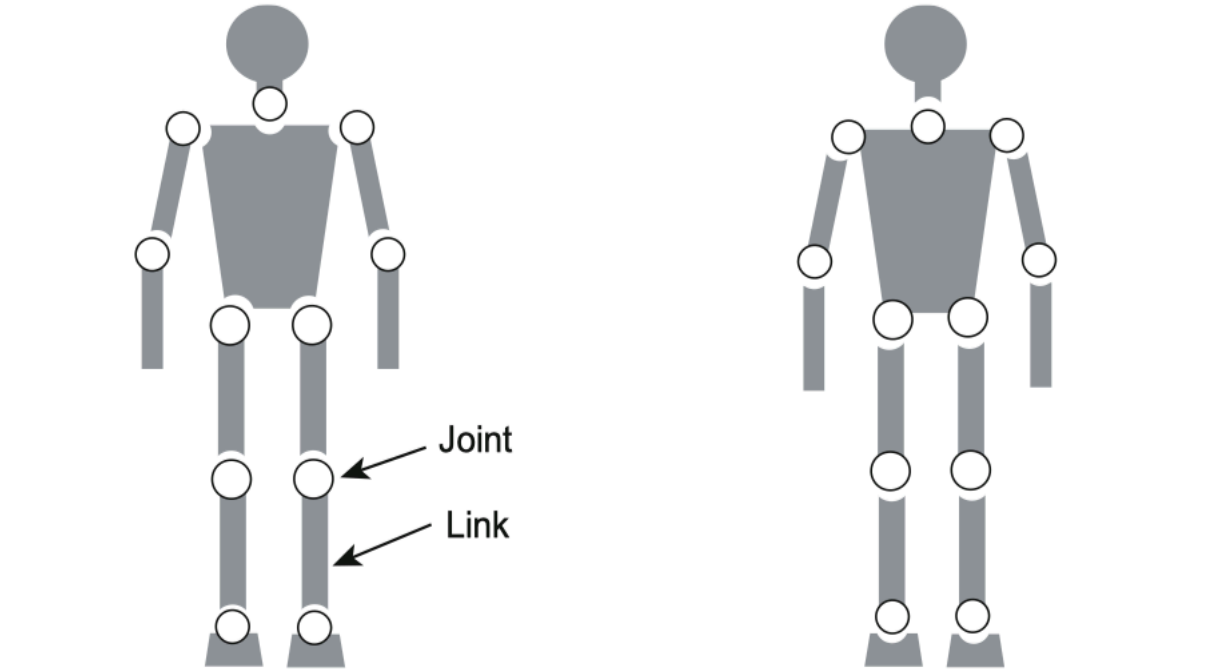
Joint axis vectors



Relative position and local coordinates.

Data Structures

ROBOT INTELLIGENCE LAB



(a) Separate into joint+link units (b) Separate into link+joint units

What is a **proper data structure** for humanoid robots?

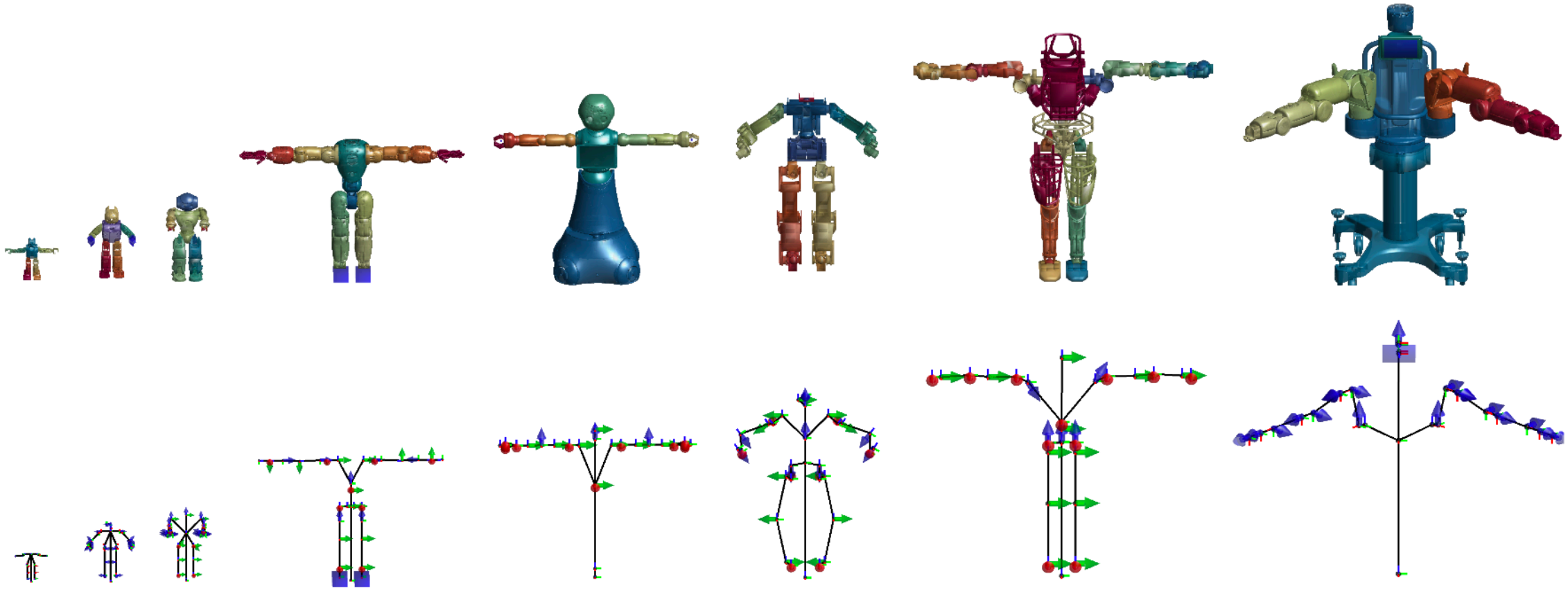
Data Structures

ROBOT INTELLIGENCE LAB

Joints vs. Joints + Links?

Kinematic Chain

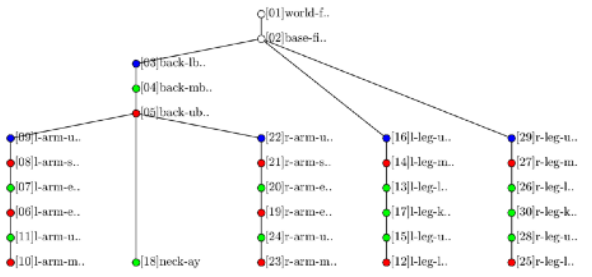
ROBOT INTELLIGENCE LAB



Kinematic Chain

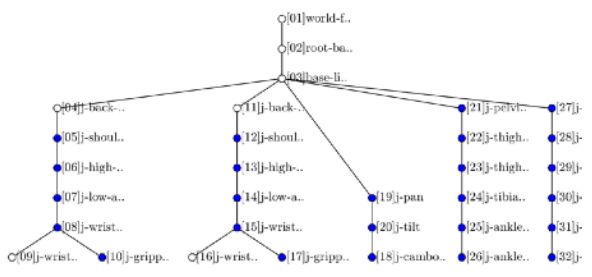
ROBOT INTELLIGENCE LAB

Atlas

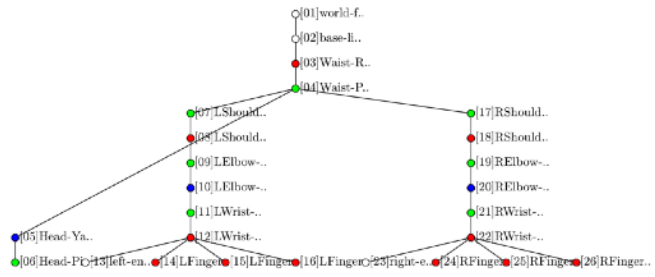


Coman

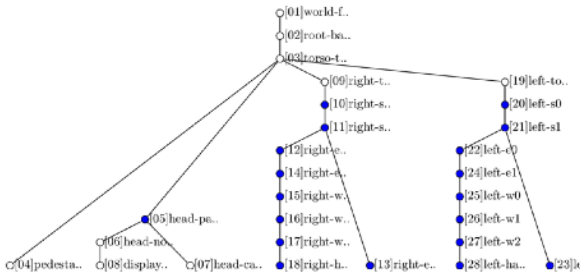
Darwin



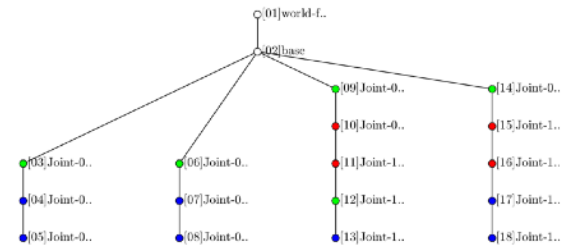
Social-robot



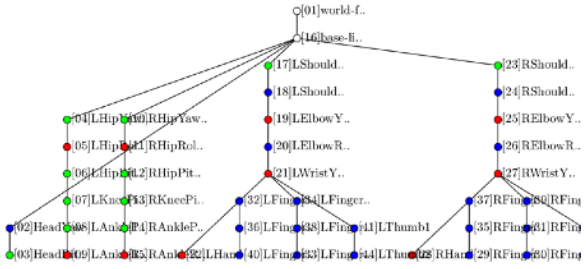
Baxter



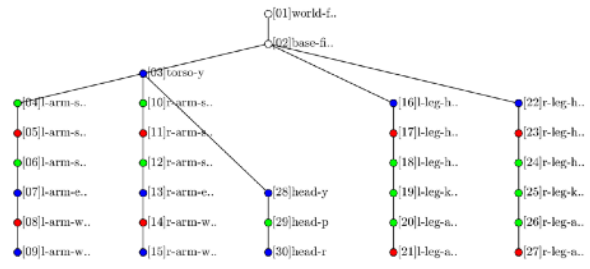
Mini



Nao

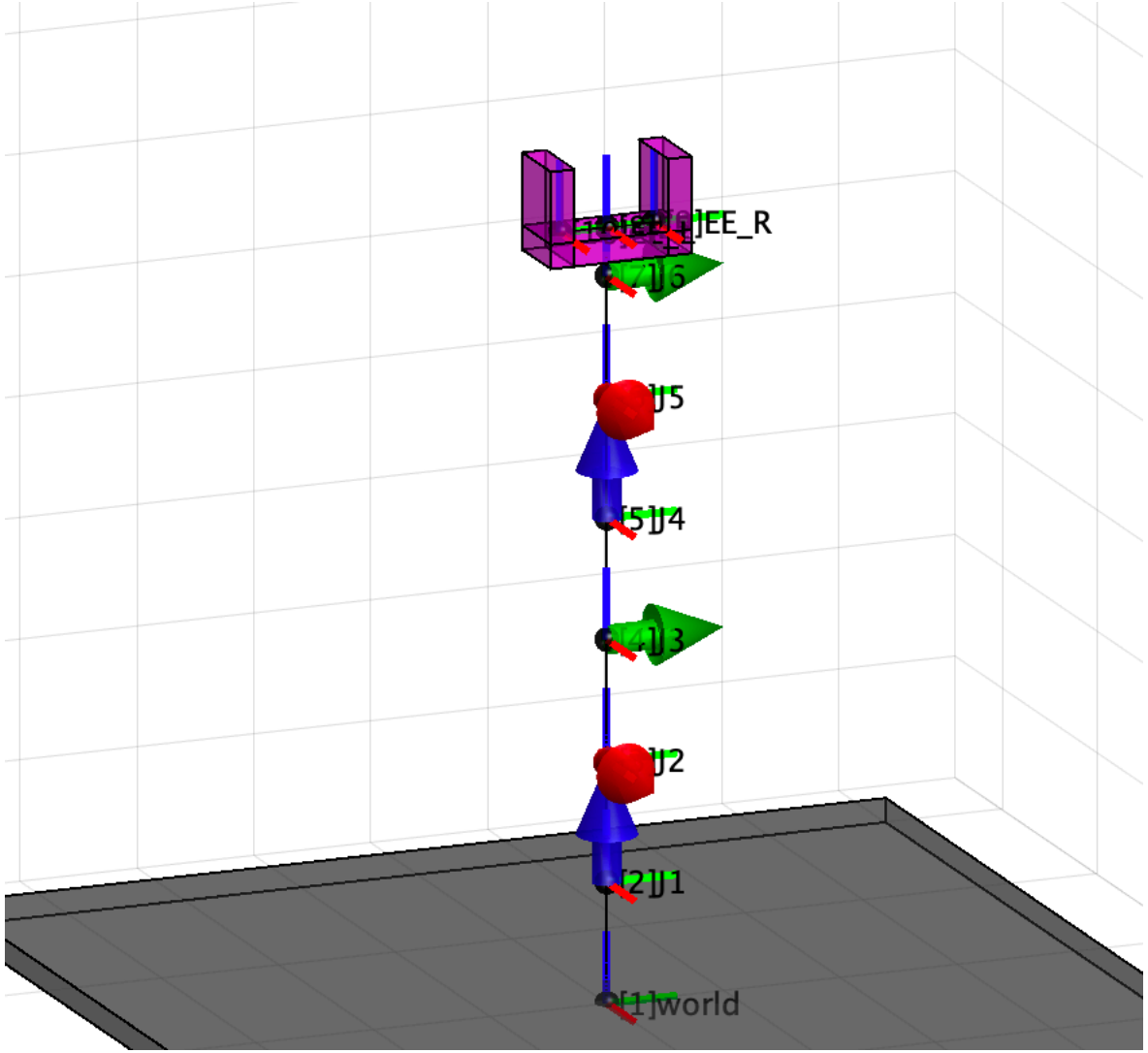


Thormang



Kinematics of a Humanoid Robot

ROBOT INTELLIGENCE LAB



Kinematic Chain

ROBOT INTELLIGENCE LAB

```
% Initialize a kinematic chain
chain = init_chain('name','kinematic_chain');

% Add joint to the chain
chain = add_joint_to_chain(chain,'name','world');
chain = add_joint_to_chain(chain,'name','J1','parent_name','world',...
    'p_offset',cv([0,0,0.5]),'a',cv([0,0,1]));
chain = add_joint_to_chain(chain,'name','J2','parent_name','J1',...
    'p_offset',cv([0,0,0.5]),'a',cv([1,0,0]));
chain = add_joint_to_chain(chain,'name','J3','parent_name','J2',...
    'p_offset',cv([0,0,0.5]),'a',cv([0,1,0]));
chain = add_joint_to_chain(chain,'name','J4','parent_name','J3',...
    'p_offset',cv([0,0,0.5]),'a',cv([0,0,1]));
chain = add_joint_to_chain(chain,'name','J5','parent_name','J4',...
    'p_offset',cv([0,0,0.5]),'a',cv([1,0,0]));
chain = add_joint_to_chain(chain,'name','J6','parent_name','J5',...
    'p_offset',cv([0,0,0.5]),'a',cv([0,1,0]));
chain = add_joint_to_chain(chain,'name','EE','parent_name','J6',...
    'p_offset',cv([0,0,0.2]),'a',cv([0,0,0]));
chain = add_joint_to_chain(chain,'name','EE_R','parent_name','EE',...
    'p_offset',cv([0,0.2,0]),'a',cv([0,0,0]));
chain = add_joint_to_chain(chain,'name','EE_L','parent_name','EE',...
    'p_offset',cv([0,-0.2,0]),'a',cv([0,0,0]));

% Add link to the chain
box_added = struct('xyz_min',[-2,-2,0],'xyz_len',[4,4,0.1],...
    'p_offset',cv([0,0,0]),'R_offset',rpy2r([0,0,0]*D2R),...
    'color',0.3*[1,1,1],'alpha',0.5,'ec','k');
chain = add_link_to_chain(chain,'name','base_link','joint_name','world','box_added',box_added);
box_added = struct('xyz_min',[-0.15,-0.3,-0.1],'xyz_len',[0.3,0.6,0.1],...
    'p_offset',cv([0,0,0]),'R_offset',rpy2r([0,0,0]*D2R),...
    'color','m','alpha',0.5,'ec','k');
chain = add_link_to_chain(chain,'name','EE_link','joint_name','EE','box_added',box_added);
box_added = struct('xyz_min',[-0.15,0,-0.1],'xyz_len',[0.3,0.1,0.4],...
    'p_offset',cv([0,0,0]),'R_offset',rpy2r([0,0,0]*D2R),...
    'color','m','alpha',0.5,'ec','k');
chain = add_link_to_chain(chain,'name','EE_R_link','joint_name','EE_R','box_added',box_added);
box_added = struct('xyz_min',[-0.15,-0.1,-0.1],'xyz_len',[0.3,0.1,0.4],...
    'p_offset',cv([0,0,0]),'R_offset',rpy2r([0,0,0]*D2R),...
    'color','m','alpha',0.5,'ec','k');
chain = add_link_to_chain(chain,'name','EE_L_link','joint_name','EE_L','box_added',box_added);
```

Kinematic Chain

ROBOT INTELLIGENCE LAB

```
chain =

  struct with fields:

              name: 'kinematic_chain'
                dt: 0.0100
             joint: [1×10 struct]
       joint_names: {1×10 cell}
           n_joint: 10
     rev_joint_names: {'J1'  'J2'  'J3'  'J4'  'J5'  'J6'}
       n_rev_joint: 6
              link: [1×4 struct]
        link_names: {1×4 cell}
            n_link: 4
```

Kinematic Chain

ROBOT INTELLIGENCE LAB

```
>> chain.joint

ans =

  1×10 struct array with fields:

    name
    p
    R
    a
    type
    p_offset
    R_offset
    q
    dq
    ddq
    q_diff
    q_prev
    v
    vo
    w
    dvo
    dw
    u
    ext_f
    parent
    childs
    link_idx
    limit
```

```
>> chain.link

ans =

  1×4 struct array with fields:

    name
    joint_idx
    p_offset
    R_offset
    mesh_path
    scale
    fv
    box
    capsule
    box_added
    m
    I_bar
    com_bar
    v
    vo
    w
```

```
% Initialize a kinematic chain
chain = init_chain('name','kinematic_chain');

% Add joint to the chain
chain = add_joint_to_chain(chain,'name','world');
chain = add_joint_to_chain(chain,'name','J1','parent_name','world',...
    'p_offset',cv([0,0,0.5]),'a',cv([0,0,1]));
chain = add_joint_to_chain(chain,'name','J2','parent_name','J1',...
    'p_offset',cv([0,0,0.5]),'a',cv([1,0,0]));
chain = add_joint_to_chain(chain,'name','J3','parent_name','J2',...
    'p_offset',cv([0,0,0.5]),'a',cv([0,1,0]));
chain = add_joint_to_chain(chain,'name','J4','parent_name','J3',...
    'p_offset',cv([0,0,0.5]),'a',cv([0,0,1]));
chain = add_joint_to_chain(chain,'name','J5','parent_name','J4',...
    'p_offset',cv([0,0,0.5]),'a',cv([1,0,0]));
chain = add_joint_to_chain(chain,'name','J6','parent_name','J5',...
    'p_offset',cv([0,0,0.5]),'a',cv([0,1,0]));
chain = add_joint_to_chain(chain,'name','EE','parent_name','J6',...
    'p_offset',cv([0,0,0.2]),'a',cv([0,0,0]));
chain = add_joint_to_chain(chain,'name','EE_R','parent_name','EE',...
    'p_offset',cv([0,0.2,0]),'a',cv([0,0,0]));
chain = add_joint_to_chain(chain,'name','EE_L','parent_name','EE',...
    'p_offset',cv([0,-0.2,0]),'a',cv([0,0,0]));

% Add link to the chain
box_added = struct('xyz_min',[-2,-2,0],'xyz_len',[4,4,0.1],...
    'p_offset',cv([0,0,0]),'R_offset',rpy2r([0,0,0]*D2R),...
    'color',0.3*[1,1,1],'alpha',0.5,'ec','k');
chain = add_link_to_chain(chain,'name','base_link','joint_name','world','box_added',box_added);
box_added = struct('xyz_min',[-0.15,-0.3,-0.1],'xyz_len',[0.3,0.6,0.1],...
    'p_offset',cv([0,0,0]),'R_offset',rpy2r([0,0,0]*D2R),...
    'color','m','alpha',0.5,'ec','k');
chain = add_link_to_chain(chain,'name','EE_link','joint_name','EE','box_added',box_added);
box_added = struct('xyz_min',[-0.15,0,-0.1],'xyz_len',[0.3,0.1,0.4],...
    'p_offset',cv([0,0,0]),'R_offset',rpy2r([0,0,0]*D2R),...
    'color','m','alpha',0.5,'ec','k');
chain = add_link_to_chain(chain,'name','EE_R_link','joint_name','EE_R','box_added',box_added);
box_added = struct('xyz_min',[-0.15,-0.1,-0.1],'xyz_len',[0.3,0.1,0.4],...
    'p_offset',cv([0,0,0]),'R_offset',rpy2r([0,0,0]*D2R),...
    'color','m','alpha',0.5,'ec','k');
chain = add_link_to_chain(chain,'name','EE_L_link','joint_name','EE_L','box_added',box_added);

% Update chain mass, inertia, and com
chain = update_chain_mass_inertia_com(chain,'density',500);
```

ROBOT INTELLIGENCE LAB

Thank You