Intelligent Robotics

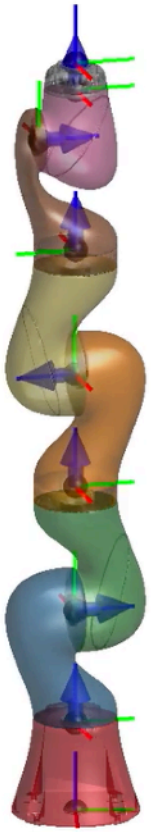
Kinematics

Sungjoon Choi, Korea University

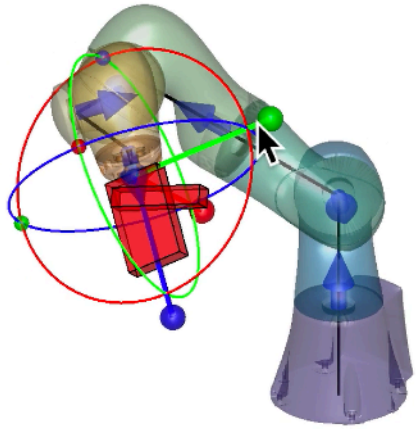
Kinematics

ROBOT INTELLIGENCE LAB

- Kinematics relates the position and orientation of a link and the joint angles of a mechanism.
  - It is the **basis** on which robotics is formed.



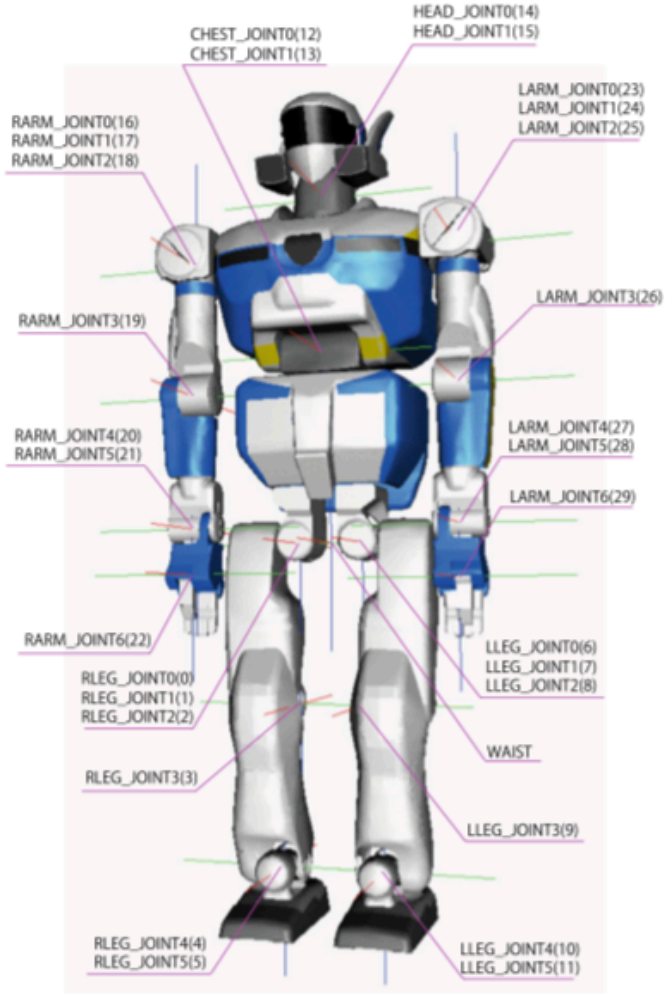
Forward Kinematics



Inverse Kinematics

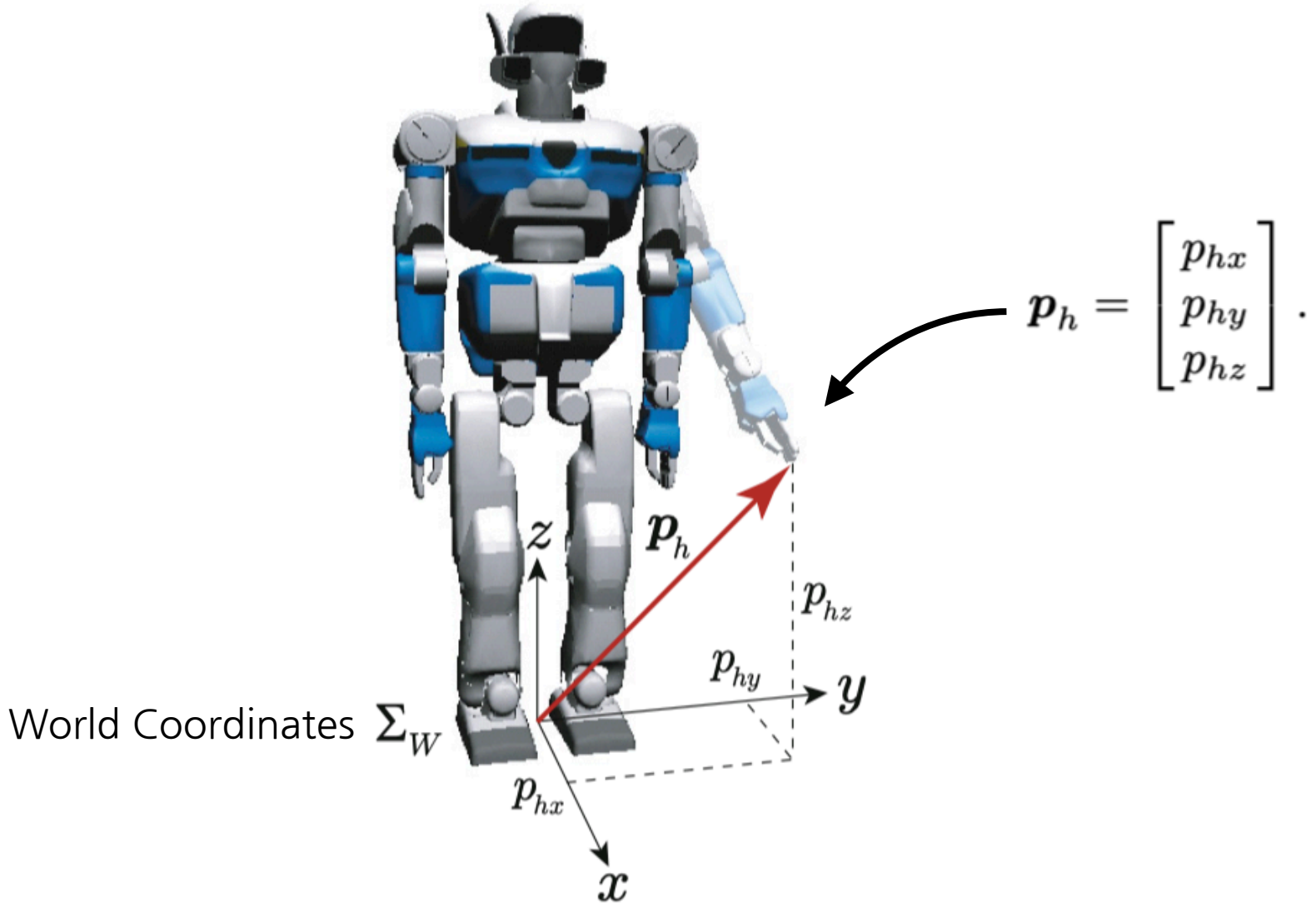
Coordinate Transforms

ROBOT INTELLIGENCE LAB



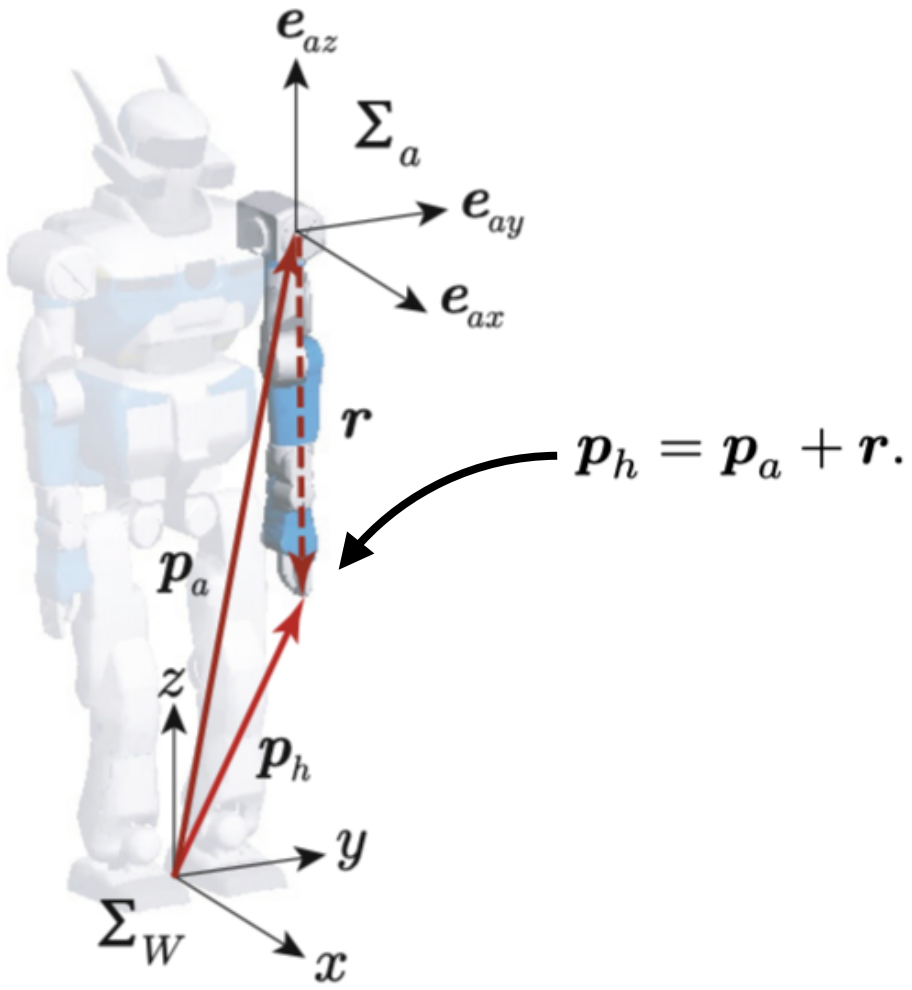
World Coordinates

ROBOT INTELLIGENCE LAB



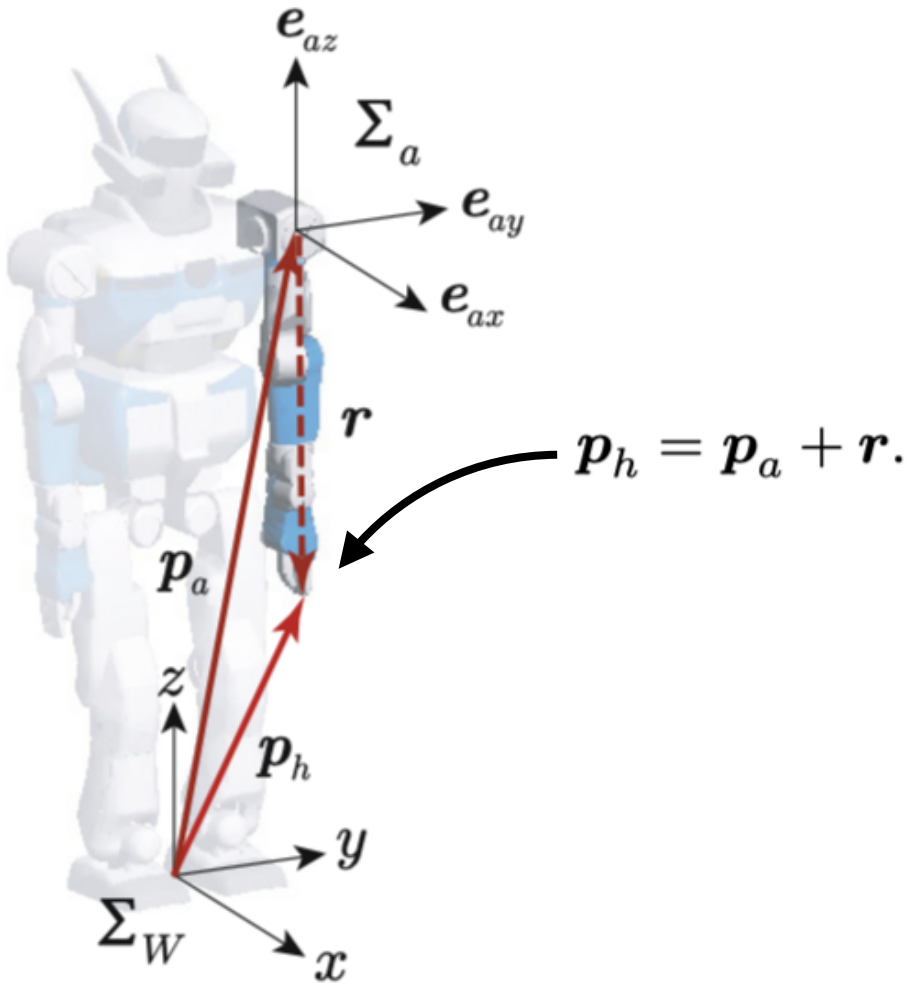
Local Coordinates

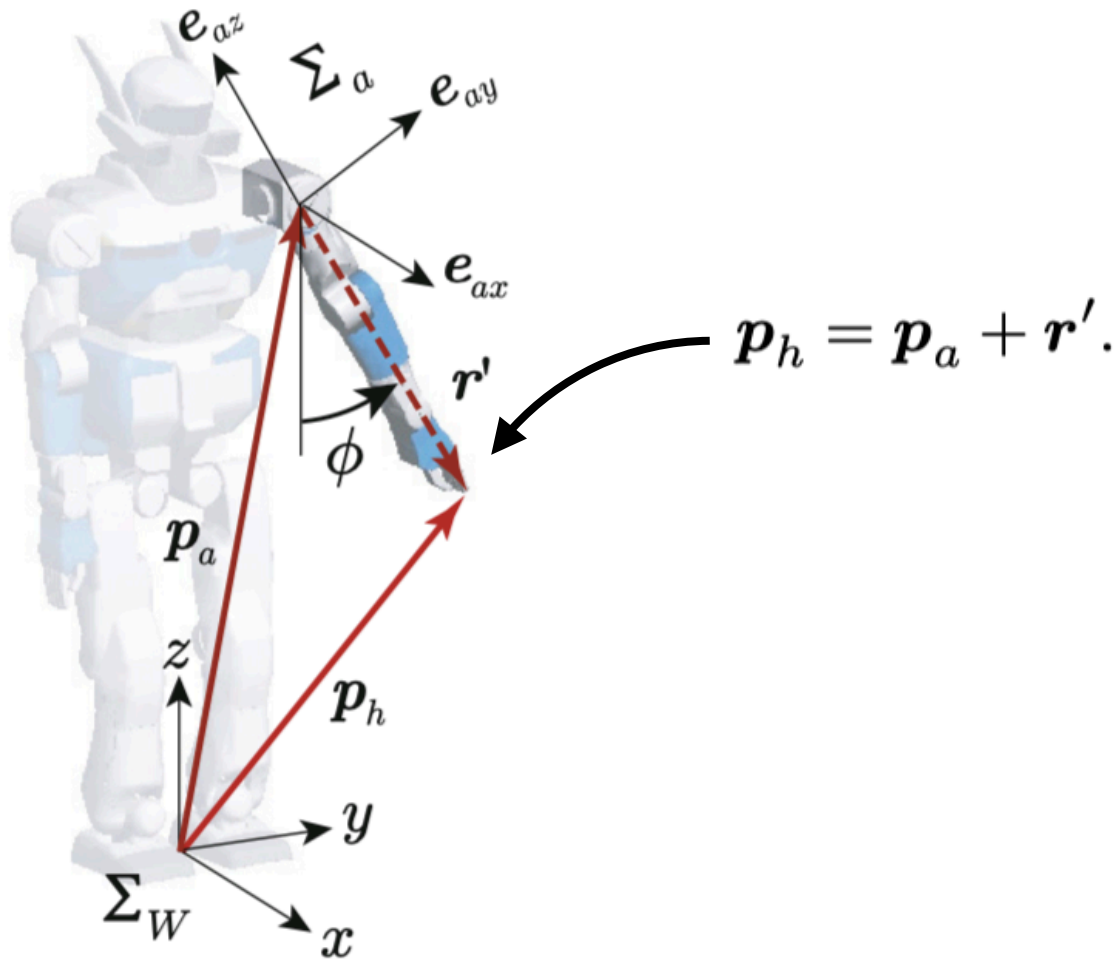
ROBOT INTELLIGENCE LAB



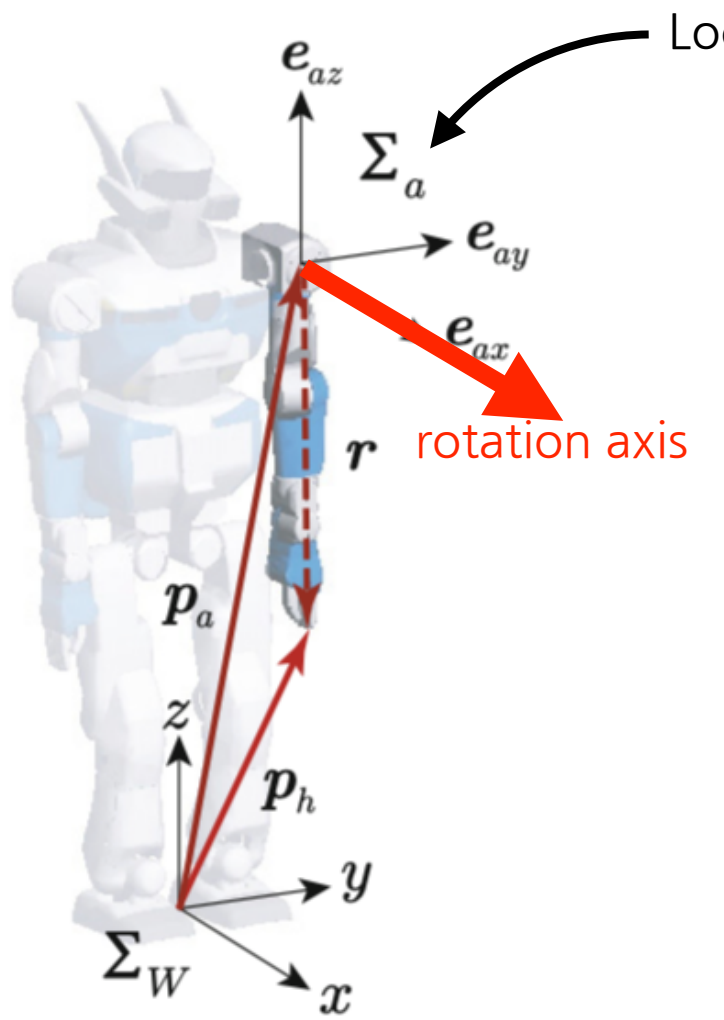
Local Coordinates

ROBOT INTELLIGENCE LAB





Local Coordinates



$$e_{ax} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad e_{ay} = \begin{bmatrix} 0 \\ \cos\phi \\ \sin\phi \end{bmatrix}, \quad e_{az} = \begin{bmatrix} 0 \\ -\sin\phi \\ \cos\phi \end{bmatrix}.$$

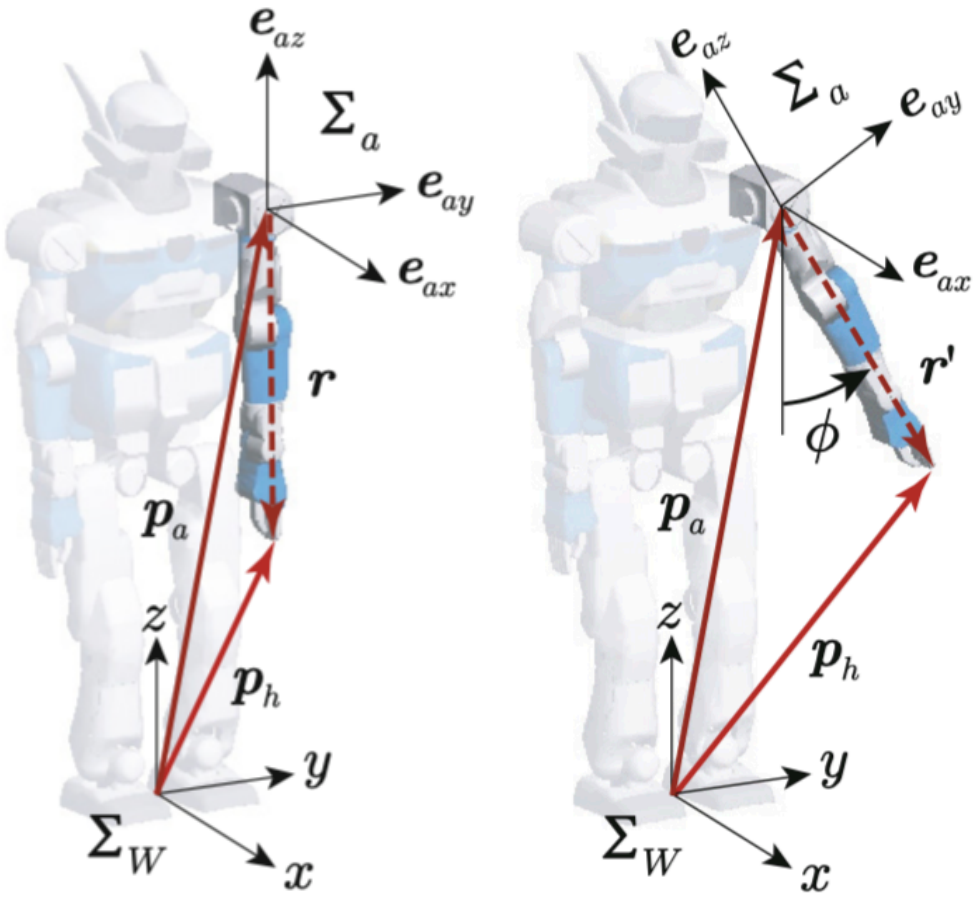
Since it is a rotation around the x axis, only $\boldsymbol{e}_{ay}$ and $\boldsymbol{e}_{az}$ change by ϕ. Let us define a 3×3 matrix, $\boldsymbol{R}_a$ as follows:

$$\boldsymbol{R}_a \equiv [\boldsymbol{e}_{ax}\, \boldsymbol{e}_{ay}\, \boldsymbol{e}_{az}]. \tag{2.3}$$

Using the matrix $\boldsymbol{R}_a$ we can describe the relationship between $\boldsymbol{r}$ and $\boldsymbol{r}'$ in Fig. 2.3 as follows:

$$\boldsymbol{r}' = \boldsymbol{R}_a \boldsymbol{r}. \tag{2.4}$$

Local Coordinates



The relationship between the two are as follows. From (2.1), (2.4) and (2.5), we get,

$$\boldsymbol{p}_h = \boldsymbol{p}_a + \boldsymbol{R}_a \, {}^a\boldsymbol{p}_h. \tag{2.6}$$

Equation (2.6) can also be rewritten as,

$$\begin{bmatrix} \boldsymbol{p}_h \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_a & \boldsymbol{p}_a \\ 0\,0\,0 & 1 \end{bmatrix} \begin{bmatrix} {}^a\boldsymbol{p}_h \\ 1 \end{bmatrix}. \tag{2.7}$$

Here we added 0 and 1 to the matrices to match the dimensions and make it match (2.6). The 4×4 matrix on the right hand side of the equation comes from combining the position vector $\boldsymbol{p}_a$ and the rotation matrix $\boldsymbol{R}_a$. This can be written as,

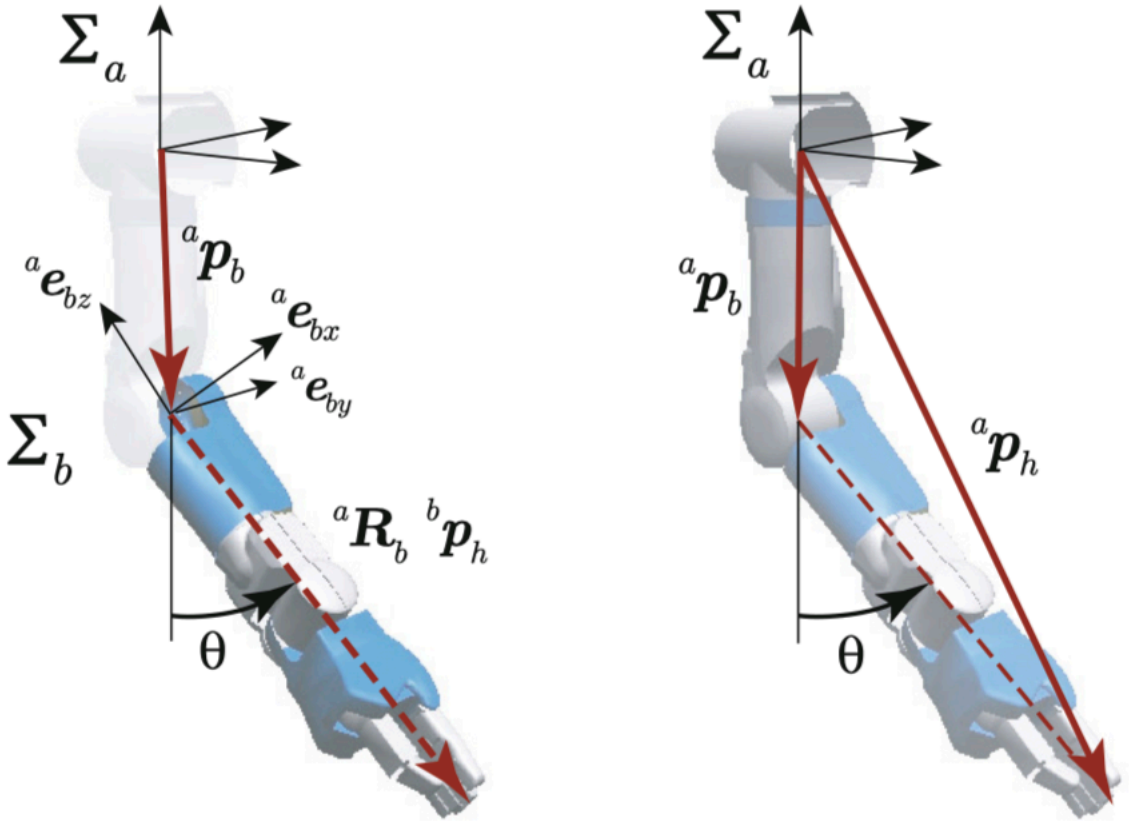
$$\boldsymbol{T}_a \equiv \begin{bmatrix} \boldsymbol{R}_a & \boldsymbol{p}_a \\ 0\,0\,0 & 1 \end{bmatrix}.$$

Matrices like this are called **Homogeneous transformation matrices**[2]. The Homogeneous Transformation $\boldsymbol{T}_a$ converts points described in arm local coordinates to world coordinates

$$\begin{bmatrix} \boldsymbol{p} \\ 1 \end{bmatrix} = \boldsymbol{T}_a \begin{bmatrix} {}^a\boldsymbol{p} \\ 1 \end{bmatrix}.$$

Local to Local Coordinate Systems

ROBOT INTELLIGENCE LAB



$$ {}^a\boldsymbol{e}_{bx} = \begin{bmatrix} \cos\theta \\ 0 \\ \sin\theta \end{bmatrix}, \quad {}^a\boldsymbol{e}_{by} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \quad {}^a\boldsymbol{e}_{bz} = \begin{bmatrix} -\sin\theta \\ 0 \\ \cos\theta \end{bmatrix}. \tag{2.8} $$

Since the elbow rotates around the y axis, only the vectors ${}^a\boldsymbol{e}_{bx}$ and ${}^a\boldsymbol{e}_{bz}$ change by θ. The vectors are defined in Σ_a space and thus they have the indicator a on the top left. Let us define a matrix ${}^a\boldsymbol{R}_b$ as a combination of the three unit vectors.

$$ {}^a\boldsymbol{R}_b \equiv [{}^a\boldsymbol{e}_{bx}\ {}^a\boldsymbol{e}_{by}\ {}^a\boldsymbol{e}_{bz}] \tag{2.9} $$

The conversion of ${}^b\boldsymbol{p}_h$ (in Fig. 2.4(a)) which is in Σ_b space, to ${}^a\boldsymbol{p}_h$ (in Fig. 2.4 (b)) in Σ_a space becomes as follows

$$ \begin{bmatrix} {}^a\boldsymbol{p}_h \\ 1 \end{bmatrix} = {}^a\boldsymbol{T}_b \begin{bmatrix} {}^b\boldsymbol{p}_h \\ 1 \end{bmatrix}. \tag{2.10} $$

Here we defined a homogeneous transformation ${}^a\boldsymbol{T}_b$ as follows

$$ {}^a\boldsymbol{T}_b \equiv \begin{bmatrix} {}^a\boldsymbol{R}_b & {}^a\boldsymbol{p}_b \\ 0\,0\,0 & 1 \end{bmatrix}. $$

Note that ${}^a\boldsymbol{p}_b$ is the origin of Σ_b viewed from Σ_a.

By combining the results of (2.10) with (2.7) we get the equation which converts a point defined in Σ_b space to world coordinates. The example below converts the manipulator end point in Σ_b

$$ \begin{bmatrix} \boldsymbol{p}_h \\ 1 \end{bmatrix} = \boldsymbol{T}_a\ {}^a\boldsymbol{T}_b \begin{bmatrix} {}^b\boldsymbol{p}_h \\ 1 \end{bmatrix}. \tag{2.11} $$

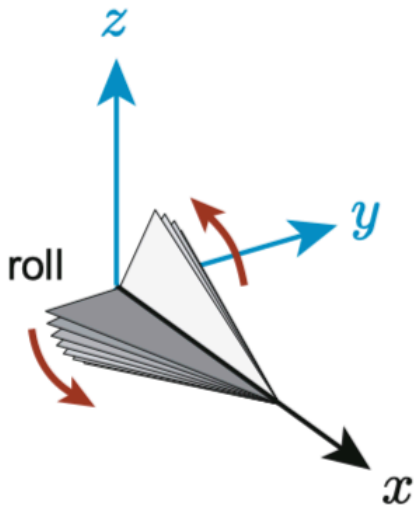
Homogeneous Transformation

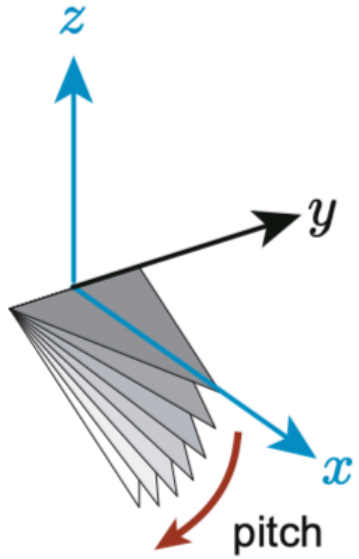
ROBOT INTELLIGENCE LAB

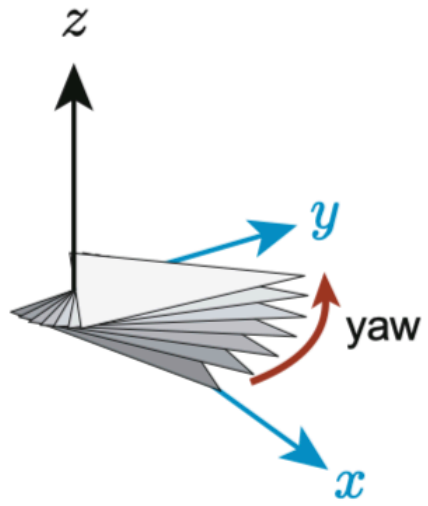
$$\boldsymbol{T}_N = \boldsymbol{T}_1{}^1\boldsymbol{T}_2{}^2\boldsymbol{T}_3 \ldots {}^{N-1}\boldsymbol{T}_N.$$

Rotation Matrix

ROBOT INTELLIGENCE LAB







Rotation Axis	Name	Notation
x axis	Roll	ϕ
y axis	Pitch	θ
z axis	Yaw	ψ

Rotation Matrix

ROBOT INTELLIGENCE LAB

$$\boldsymbol{R}_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}$$

$$\boldsymbol{R}_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

$$\boldsymbol{R}_z(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

If we Roll, Pitch and then Yaw a point $\boldsymbol{p}$ around the origin, it will move to the point,

$$\boldsymbol{p}' = \boldsymbol{R}_z(\psi)\boldsymbol{R}_y(\theta)\boldsymbol{R}_x(\phi)\,\boldsymbol{p}.$$

Rotation Matrix

ROBOT INTELLIGENCE LAB

We can rewrite this as

$$\boldsymbol{p}' = \boldsymbol{R}_{rpy}(\phi, \theta, \psi)\, \boldsymbol{p}.$$

Here we have introduced the following notation

$$\begin{aligned} \boldsymbol{R}_{rpy}(\phi, \theta, \psi) &\equiv \boldsymbol{R}_z(\psi)\boldsymbol{R}_y(\theta)\boldsymbol{R}_x(\phi) \\ &= \begin{bmatrix} c_\psi c_\theta & -s_\psi c_\phi + c_\psi s_\theta s_\phi & s_\psi s_\phi + c_\psi s_\theta c_\phi \\ s_\psi c_\theta & c_\psi c_\phi + s_\psi s_\theta s_\phi & -c_\psi s_\phi + s_\psi s_\theta c_\phi \\ -s_\theta & c_\theta s_\phi & c_\theta c_\phi \end{bmatrix}, \end{aligned} \tag{2.13}$$

where $c_\psi \equiv \cos\psi$, $s_\psi \equiv \sin\psi$, $c_\theta \equiv \cos\theta$, and so on.

Note that order matters. Why?

Rotation Matrix

ROBOT INTELLIGENCE LAB

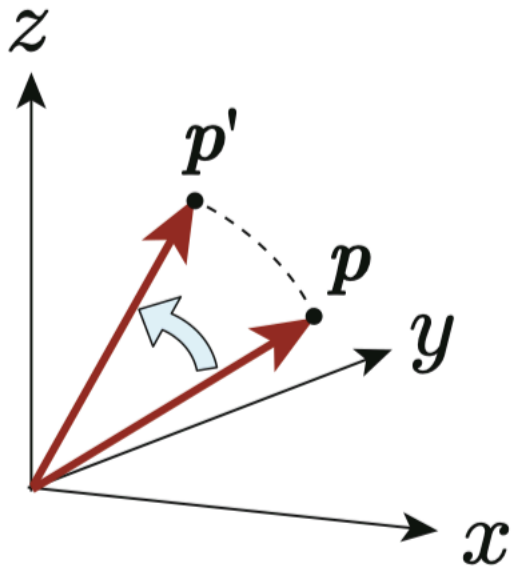
```
function R = rpy2r(rpy_rad)
%
% Euler angles (roll, pitch, and yaw) in radian to a rotation matrix
%
r_rad = rpy_rad(1);
p_rad = rpy_rad(2);
y_rad = rpy_rad(3);

cos_r = cos(r_rad); sin_r = sin(r_rad);
cos_p = cos(p_rad); sin_p = sin(p_rad);
cos_y = cos(y_rad); sin_y = sin(y_rad);

R = [...
    cos_y*cos_p,   -sin_y*cos_r+cos_y*sin_p*sin_r, sin_y*sin_r+cos_y*sin_p*cos_r ; ...
    sin_y*cos_p,    cos_y*cos_r+sin_y*sin_p*sin_r,  -cos_y*sin_r+sin_y*sin_p*cos_r ; ...
    -sin_p,         cos_p*sin_r,                    cos_p*cos_r...
    ];
```

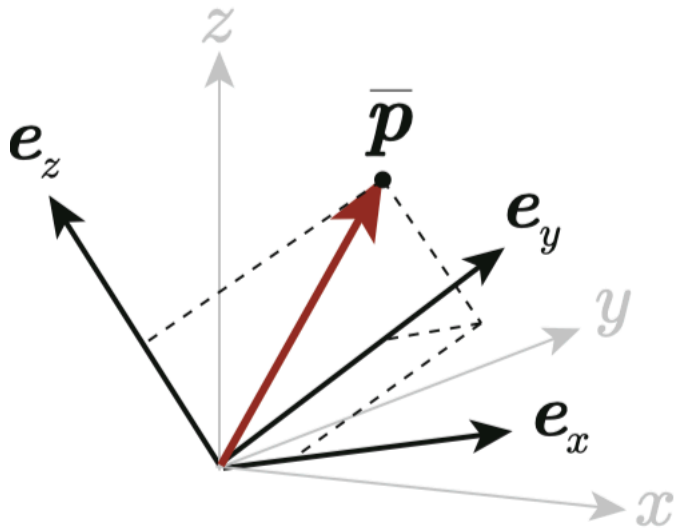
The meaning of Rotation Matrices

ROBOT INTELLIGENCE LAB



(a) Operator to rotate a vector

$$p' = Rp$$



(b) Attitude of a local frame

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

Inverse of a Rotation Matrix

ROBOT INTELLIGENCE LAB

$$\boldsymbol{R}^T = \boldsymbol{R}^{-1}.$$

Why?

Little more about a Rotation Matrix

- R is an orthogonal matrix.
  - Applying rotation on two different vectors on a rigid frame should not change its inner product: $\mathbf{a}^T\mathbf{b} = (R\mathbf{a})^T(R\mathbf{b}) = \mathbf{a}^T R^T R\mathbf{b}$. This should hold for all $\mathbf{a}$ and $\mathbf{b}$, hence, $R^T R = I$.

- $|\det(R)| = 1$
  - Since $\|R\mathbf{x}\| = \|\mathbf{x}\|$ and if we set $\mathbf{x}$ to be eigenvectors of R, we get all eigenvalues of R is 1 or -1. Then $|\det(R)|$ is the absolute of the product of all eigenvalues, it is 1.

- Combining these two, we get $\mathrm{SO}(3)$ a group of 3×3 matrices which are orthogonal and have $\det() = 1$.

Inverse of a Homogeneous Transformation Matrix

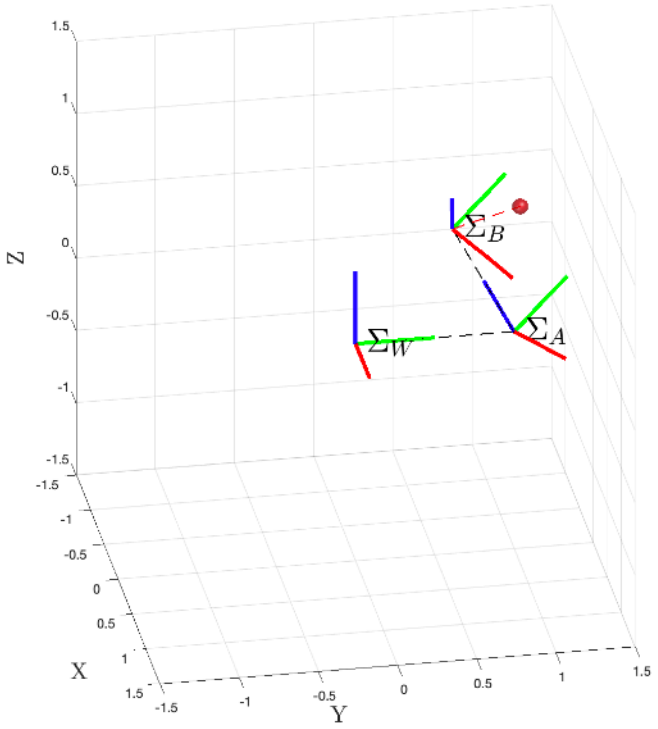
$$T = \begin{bmatrix} R & \mathbf{p} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \qquad T^{-1} = \begin{bmatrix} R^T & -R^T\mathbf{p} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix}$$

$$TT^{-1} = \begin{bmatrix} R & \mathbf{p} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \begin{bmatrix} R^T & -R^T\mathbf{p} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} = \begin{bmatrix} RR^T & -RR^T\mathbf{p}+\mathbf{p} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} = \begin{bmatrix} I & \mathbf{0}_{3\times 1} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix}$$

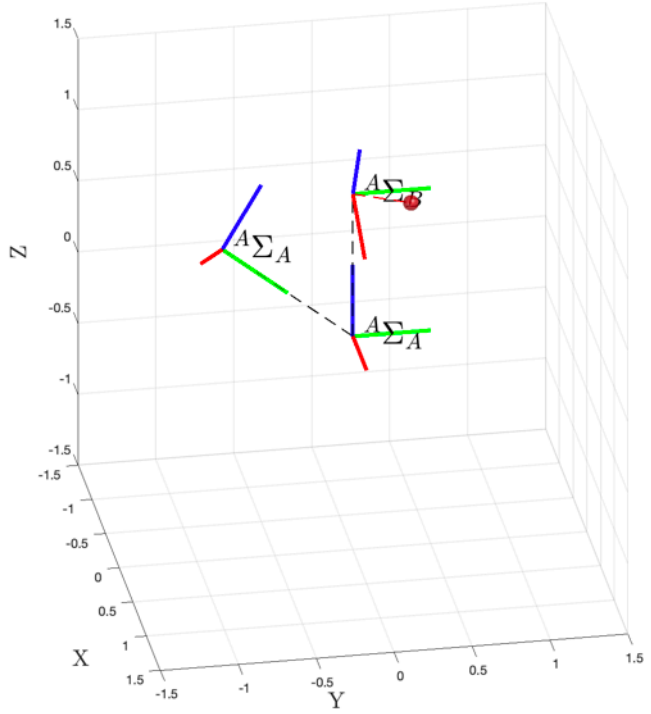
Coordinate Transform in action

ROBOT INTELLIGENCE LAB

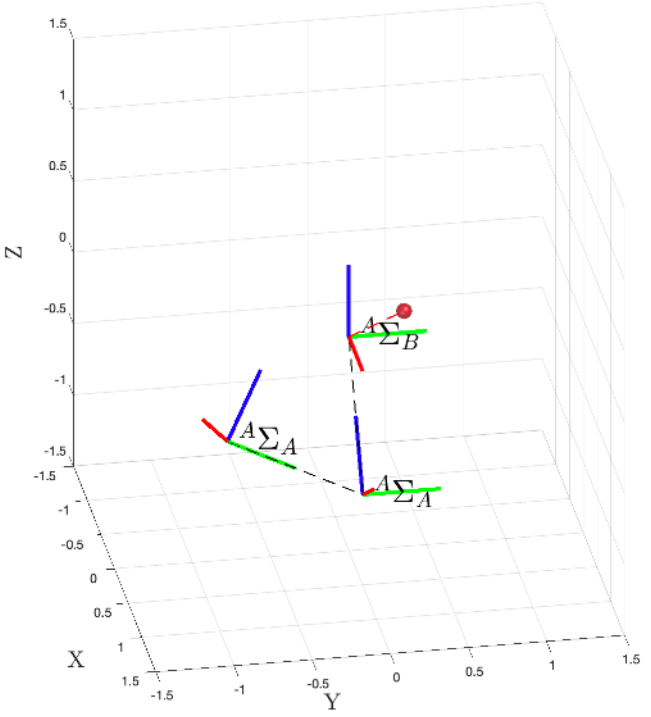
In World Coordinate System



In Local Coordinate System A



In Local Coordinate System B



Coordinate Transform in action

ROBOT INTELLIGENCE LAB

```
%
% T_W -> T_A -> T_B
%

% Initialize World Coordinate {W} and local coordinates, {A}, {B}, and {C}
T_W = pr2t('','');  % world coordinates {W}
T_A_in_W = pr2t(cv([0,1,0]),rpy2r([30,0,30]*D2R));  % Local coordinates {A} in {W}
T_B_in_A = pr2t(cv([0,0,1]),rpy2r([0,30,0]*D2R));  % Local coordinates {B} in {A}

% Coordinate transfrom: Pre-multiply
%
%       T_{BinW} = T_{W} * T_{AinW} * T_{BinA}
%
% Coordinates in {W}
T_A_in_W = T_W*T_A_in_W;  % {A} in {W}
T_B_in_W = T_W*T_A_in_W*T_B_in_A;  % {B} in {W}
% Coordinates in {A}
T_W_in_A = inv_T(T_A_in_W);
T_A_in_A = pr2t('','');
% Coordinates in {B}
T_W_in_B = inv_T(T_B_in_W);
T_A_in_B = inv_T(T_B_in_A);
T_B_in_B = pr2t('','');

% Point transform: Pre-multiply
%
%       p_X_in_{A} = T_{B}in{A} * p_X_in{B}
%       p_X_in_{W} = T_{A}in{W} * T_{B}in{A} * p_X_in{B}
%
% Point X in {B}
p_X_in_B = cv([0.3,0.3,0.3]);  % point in {B}
T_X_in_B = pr2t(p_X_in_B,'');
% Point X in {A}
T_X_in_A = T_B_in_A*T_X_in_B;
% Point X in {W}
T_X_in_W = T_A_in_W*T_B_in_A*T_X_in_B;
```

Coordinate Transform in action

ROBOT INTELLIGENCE LAB

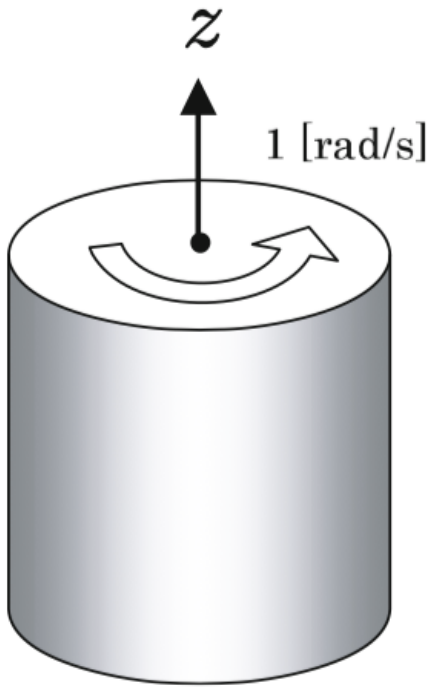
```
% Plot things in {W}
axis_info = 1.5*[-1,+1,-1,+1,-1,+1];
fig_idx = 1;
set_fig(figure(fig_idx),'pos',[0.0,0.5,0.3,0.5],...
    'view_info',[80,26],'axis_info',axis_info,'AXIS_EQUAL',1,'GRID_ON',1,...
    'REMOVE_MENUBAR',1,'USE_DRAGZOOM',1,...
    'SET_CAMLIGHT',1,'SET_MATERIAL','METAL','SET_AXISLABEL',1,'afs',18);
plot_T(T_W,'fig_idx',fig_idx,'subfig_idx',1,...
    'all',0.5,'alw',3,'text_str','~$\Sigma_W$','text_interp','latex','text_fs',25);
plot_line(t2p(T_W),t2p(T_A_in_W),'fig_idx',fig_idx,'subfig_idx',1,...
    'lc','k','lw',1,'ls','--');
plot_T(T_A_in_W,'fig_idx',fig_idx,'subfig_idx',2,...
    'all',0.5,'alw',3,'text_str','~$\Sigma_A$','text_interp','latex','text_fs',25);
plot_line(t2p(T_A_in_W),t2p(T_B_in_W),'fig_idx',fig_idx,'subfig_idx',2,...
    'lc','k','lw',1,'ls','--');
plot_T(T_B_in_W,'fig_idx',fig_idx,'subfig_idx',3,...
    'all',0.5,'alw',3,'text_str','~$\Sigma_B$','text_interp','latex','text_fs',25);
plot_line(t2p(T_B_in_W),t2p(T_X_in_W),'fig_idx',fig_idx,'subfig_idx',3,...
    'lc','r','lw',1,'ls','--');
plot_T(T_X_in_W,'fig_idx',fig_idx,'subfig_idx',4,...
    'PLOT_AXIS',0,'PLOT_SPHERE',1,'sr',0.05,'sfc','r','sfa',0.5);
plot_title('In World Coordinate System','fig_idx',fig_idx,'interpreter','latex','tfs',30);

% Plot things in {A}
fig_idx = 2;
set_fig(figure(fig_idx),'pos',[0.3,0.5,0.3,0.5],...
    'view_info',[80,26],'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);
plot_T(T_W_in_A,'fig_idx',fig_idx,'subfig_idx',1,...
    'all',0.5,'alw',3,'text_str','~$^A\Sigma_A$','text_interp','latex','text_fs',25);
plot_line(t2p(T_W_in_A),t2p(T_A_in_A),'fig_idx',fig_idx,'subfig_idx',1,...
    'lc','k','lw',1,'ls','--');
plot_T(T_A_in_A,'fig_idx',fig_idx,'subfig_idx',2,...
    'all',0.5,'alw',3,'text_str','~$^A\Sigma_A$','text_interp','latex','text_fs',25);
plot_line(t2p(T_A_in_A),t2p(T_B_in_A),'fig_idx',fig_idx,'subfig_idx',2,...
    'lc','k','lw',1,'ls','--');
plot_T(T_B_in_A,'fig_idx',fig_idx,'subfig_idx',3,...
    'all',0.5,'alw',3,'text_str','~$^A\Sigma_B$','text_interp','latex','text_fs',25);
plot_line(t2p(T_B_in_A),t2p(T_X_in_A),'fig_idx',fig_idx,'subfig_idx',3,...
    'lc','r','lw',1,'ls','--');
plot_T(T_X_in_A,'fig_idx',fig_idx,'subfig_idx',4,...
    'PLOT_AXIS',0,'PLOT_SPHERE',1,'sr',0.05,'sfc','r','sfa',0.5);
plot_title('In Local Coordinate System A','fig_idx',fig_idx,'interpreter','latex','tfs',30);

% Plot things in {B}
fig_idx = 3;
set_fig(figure(fig_idx),'pos',[0.6,0.5,0.3,0.5],...
    'view_info',[80,26],'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);
plot_T(T_W_in_B,'fig_idx',fig_idx,'subfig_idx',1,...
    'all',0.5,'alw',3,'text_str','~$^A\Sigma_A$','text_interp','latex','text_fs',25);
plot_line(t2p(T_W_in_B),t2p(T_A_in_B),'fig_idx',fig_idx,'subfig_idx',1,...
    'lc','k','lw',1,'ls','--');
plot_T(T_A_in_B,'fig_idx',fig_idx,'subfig_idx',2,...
    'all',0.5,'alw',3,'text_str','~$^A\Sigma_A$','text_interp','latex','text_fs',25);
plot_line(t2p(T_A_in_B),t2p(T_B_in_B),'fig_idx',fig_idx,'subfig_idx',2,...
    'lc','k','lw',1,'ls','--');
plot_T(T_B_in_B,'fig_idx',fig_idx,'subfig_idx',3,...
    'all',0.5,'alw',3,'text_str','~$^A\Sigma_B$','text_interp','latex','text_fs',25);
plot_line(t2p(T_B_in_B),t2p(T_X_in_B),'fig_idx',fig_idx,'subfig_idx',3,...
    'lc','r','lw',1,'ls','--');
plot_T(T_X_in_B,'fig_idx',fig_idx,'subfig_idx',4,...
    'PLOT_AXIS',0,'PLOT_SPHERE',1,'sr',0.05,'sfc','r','sfa',0.5);
plot_title('In Local Coordinate System B','fig_idx',fig_idx,'interpreter','latex','tfs',30);
```

Angular Velocity Vector

ROBOT INTELLIGENCE LAB



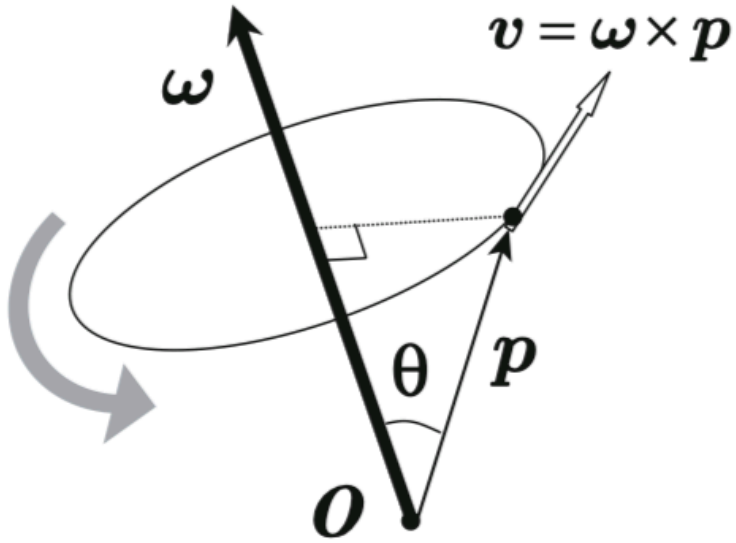
1 **ω is defined as a unit vector $\times$ scalar value**

Let $\boldsymbol{a}$ be a unit vector on the rotating axis and $\dot{q}$ be the rotational speed (a scalar value). Then the angular velocity vector of the object is their product

$$\boldsymbol{\omega} = \boldsymbol{a}\dot{q}. \tag{2.17}$$

Angular Velocity Vector

ROBOT INTELLIGENCE LAB



2 ω describes the velocity of all the points on the rotating object

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

Angular Velocity

ROBOT INTELLIGENCE LAB

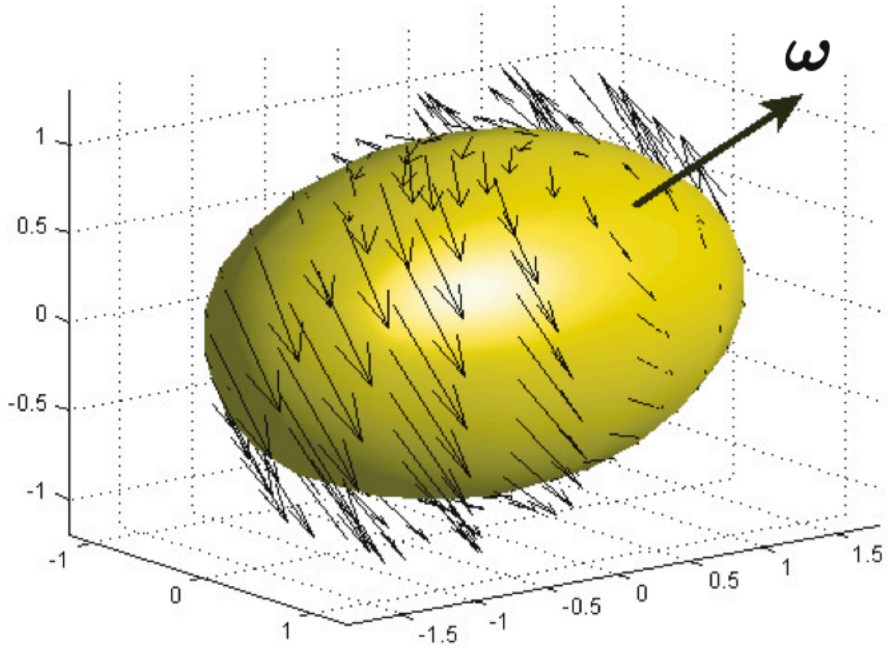


Fig. 2.9 Surface Velocity on an Ellipsoid. The object rotates according to angular velocity vector $\boldsymbol{\omega}$ (Thick Arrow). The resulting surface velocity is represented in small arrows.

Angular Velocity

ROBOT INTELLIGENCE LAB

3 **ω can be rotated too**

Let us multiply both sides of (2.17) using some rotation matrix $\boldsymbol{R}$

$$\boldsymbol{R}\boldsymbol{\omega} = \boldsymbol{R}\boldsymbol{a}\dot{q}. \tag{2.20}$$

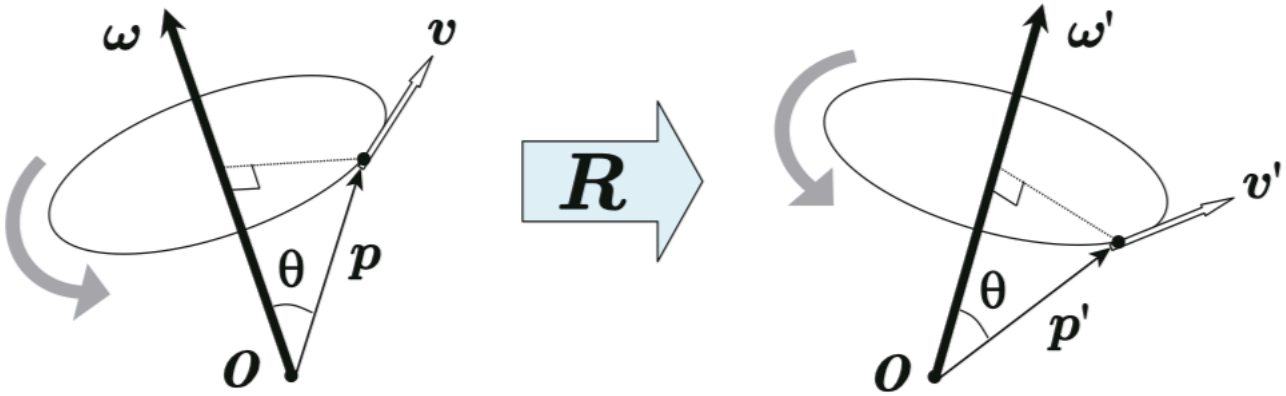
If we introduce following new vectors,

$$\boldsymbol{\omega}' = \boldsymbol{R}\boldsymbol{\omega}, \quad \boldsymbol{a}' = \boldsymbol{R}\boldsymbol{a}$$

then we can rewrite (2.20) as

$$\boldsymbol{\omega}' = \boldsymbol{a}'\dot{q}.$$

Angular Velocity



Next let us try to rotate an angular velocity vector, a position vector, and the corresponding velocity vector using the same rotation matrix $\boldsymbol{R}$ as shown in Fig. 2.10 which gives

$$\boldsymbol{\omega}' = \boldsymbol{R}\boldsymbol{\omega}, \quad \boldsymbol{p}' = \boldsymbol{R}\boldsymbol{p}, \quad \boldsymbol{v}' = \boldsymbol{R}\boldsymbol{v}.$$

Since the spatial relation between these three vectors does not change by this rotation, the definition of the cross product must be preserved, so

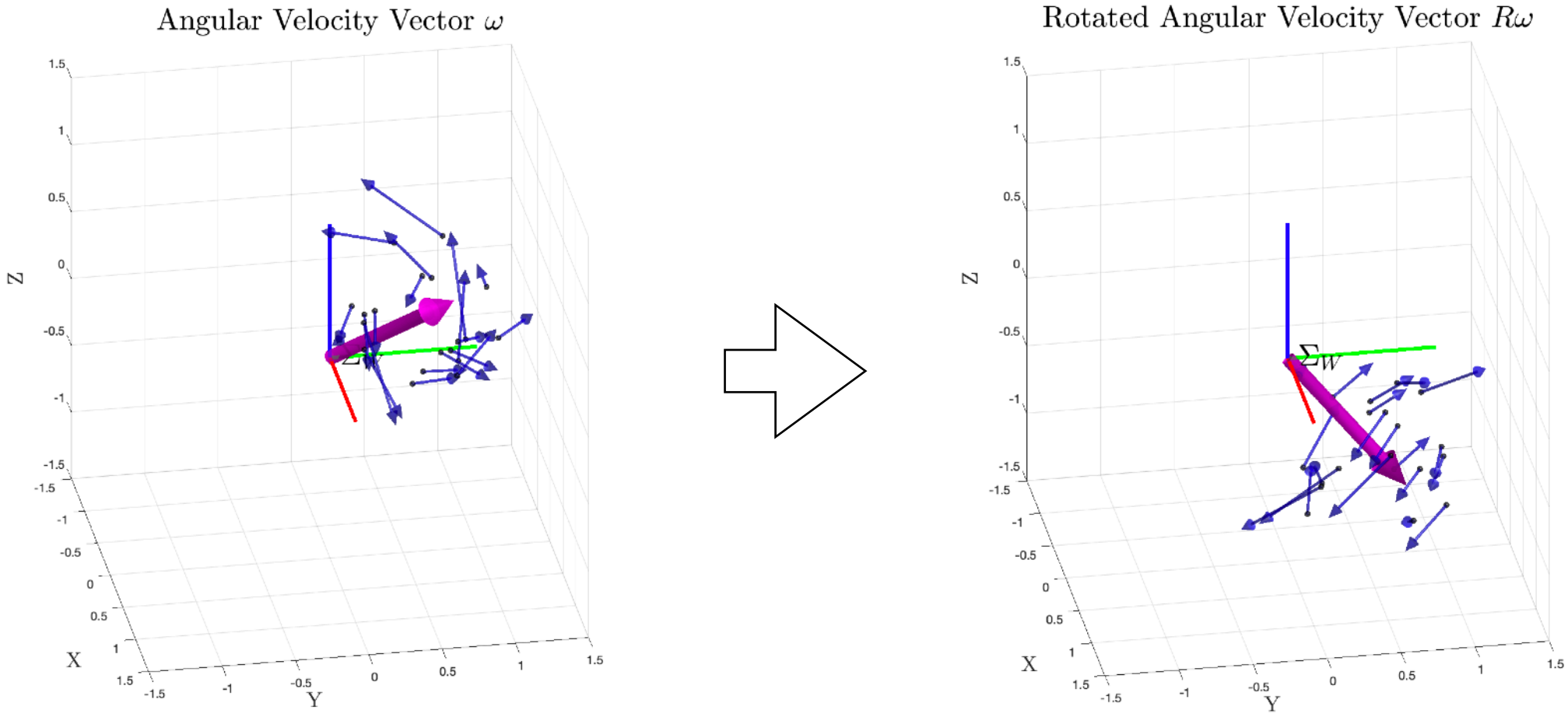
$$\boldsymbol{v}' = \boldsymbol{\omega}' \times \boldsymbol{p}'. \tag{2.21}$$

Substituting the rotated vectors $\boldsymbol{v}', \boldsymbol{\omega}'$ and $\boldsymbol{p}'$ by their originals $\boldsymbol{R}\boldsymbol{v}, \boldsymbol{R}\boldsymbol{\omega}$ and $\boldsymbol{R}\boldsymbol{p}$, we have

$$\boldsymbol{R}(\boldsymbol{\omega} \times \boldsymbol{p}) = (\boldsymbol{R}\boldsymbol{\omega}) \times (\boldsymbol{R}\boldsymbol{p}). \tag{2.22}$$

Rotated Angular Velocity in Action

ROBOT INTELLIGENCE LAB



Rotated Angular Velocity in Action

ROBOT INTELLIGENCE LAB

```
% Angular velocity vector
w = 0.5 + 0.5*rand(3,1);

% Plot the original angular velocity vector and directional velocity at point
fig_idx = 1; axis_info = 1.5*[-1,+1,-1,+1,-1,+1];
set_fig(figure(fig_idx),'pos',[0.0,0.5,0.3,0.5],...
    'view_info',[80,26],'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);
plot_T(pr2t(cv([0,0,0]),eye(3,3)),'fig_idx',fig_idx,'subfig_idx',1,...
    'PLOT_AXIS',1,'all',1.0,'alw',3,'PLOT_SPHERE',0,...
    'text_str','~$\Sigma_W$','text_interp','latex','text_fs',25); % world coordinate
sw = 0.05; tw = 0.1; % stem width and tip width
plot_arrow_3d(zeros(3,1),w,'fig_idx',fig_idx,'subfig_idx',1,'alpha',0.7,'color','m',...
    'sw',sw,'tw',tw); % angular velocity vector
ps = cell(1,20); vs = cell(1,20);
for i_idx = 1:20
    p = rand(3,1);          % random point
    v = cross(w,p);         % the directional velocity of the point
    ps{i_idx} = p; vs{i_idx} = v; % append point and directional velocity
    sw = 0.01; tw = 0.04;
    plot_T(p2t(p),'fig_idx',fig_idx,'subfig_idx',1+i_idx,...
        'PLOT_AXIS',0,'PLOT_SPHERE',1,'sr',0.02,'sfc','k');
    plot_arrow_3d(p,p+v,'fig_idx',fig_idx,'subfig_idx',1+i_idx,...
        'alpha',0.5,'color','b','sw',sw,'tw',tw); % directional velocity vector
end
plot_title('Angular Velocity Vector $\omega$',...
    'fig_idx',1,'tfs',25,'interpreter','latex');

% Get a random rotation matrix to rotate the angular velocity vector
R = rpy2r(360*rand(3,1));

% Plot the rotated angular velocity vector and directional velocity at point
fig_idx = 2;
set_fig(figure(fig_idx),'pos',[0.3,0.5,0.3,0.5],...
    'view_info',[80,26],'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);
plot_T(pr2t(cv([0,0,0]),eye(3,3)),'fig_idx',fig_idx,'subfig_idx',1,...
    'PLOT_AXIS',1,'all',1.0,'alw',3,'PLOT_SPHERE',0,...
    'text_str','~$\Sigma_W$','text_interp','latex','text_fs',25); % world coordinate
sw = 0.05; tw = 0.1; % stem width and tip width
plot_arrow_3d(zeros(3,1),R*w,'fig_idx',fig_idx,'subfig_idx',1,'alpha',0.7,'color','m',...
    'sw',sw,'tw',tw); % rotated angular velocity vector
for i_idx = 1:20
    p = R*ps{i_idx}; % rotate position
    v = R*vs{i_idx}; % rotate velocity
    sw = 0.01; tw = 0.04;
    plot_T(p2t(p),'fig_idx',fig_idx,'subfig_idx',1+i_idx,...
        'PLOT_AXIS',0,'PLOT_SPHERE',1,'sr',0.02,'sfc','k');
    plot_arrow_3d(p,p+v,'fig_idx',fig_idx,'subfig_idx',1+i_idx,...
        'alpha',0.5,'color','b','sw',sw,'tw',tw); % directional velocity vector
end
plot_title('Rotated Angular Velocity Vector $R\omega$',...
    'fig_idx',fig_idx,'tfs',25,'interpreter','latex');
```

Thank You