MCT344/MCT342/CSE373/CSE471

Industrial Robotics

Lecture 2: Rigid Motions for Robotics

Ain Shams University
Faculty of Engineering

Presented By: Dr. Diaa Emad

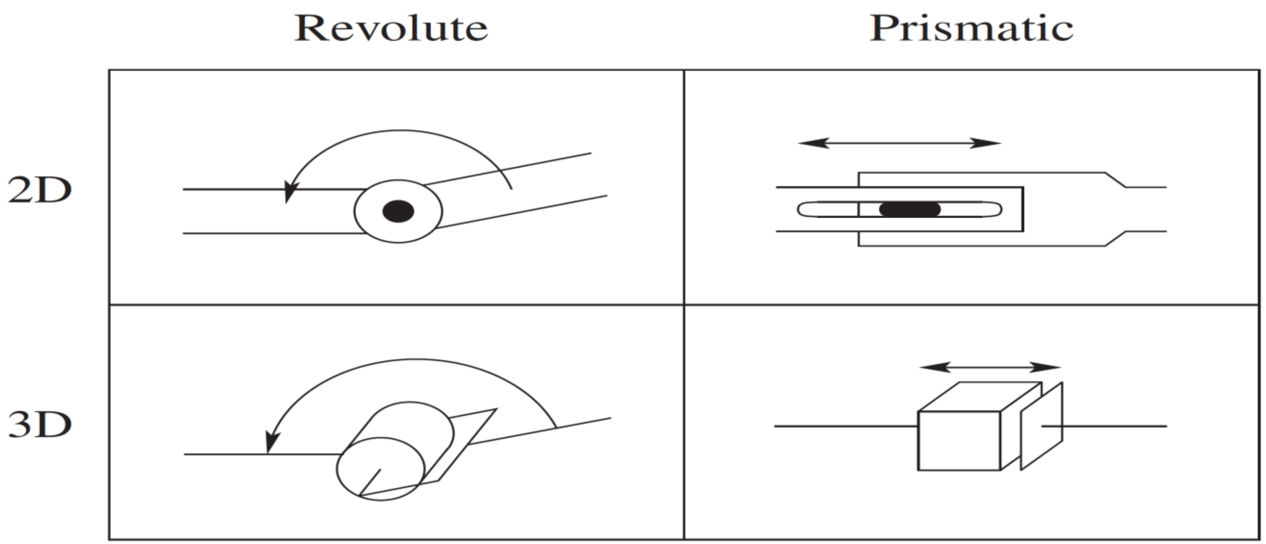
Today’s Agenda

- *Representing Positions.*
- *Representing Rotations.*
- *Composition of Rotations*
- *Parameterizations of Rotations.*
- *Homogeneous Transformations*

Introduction

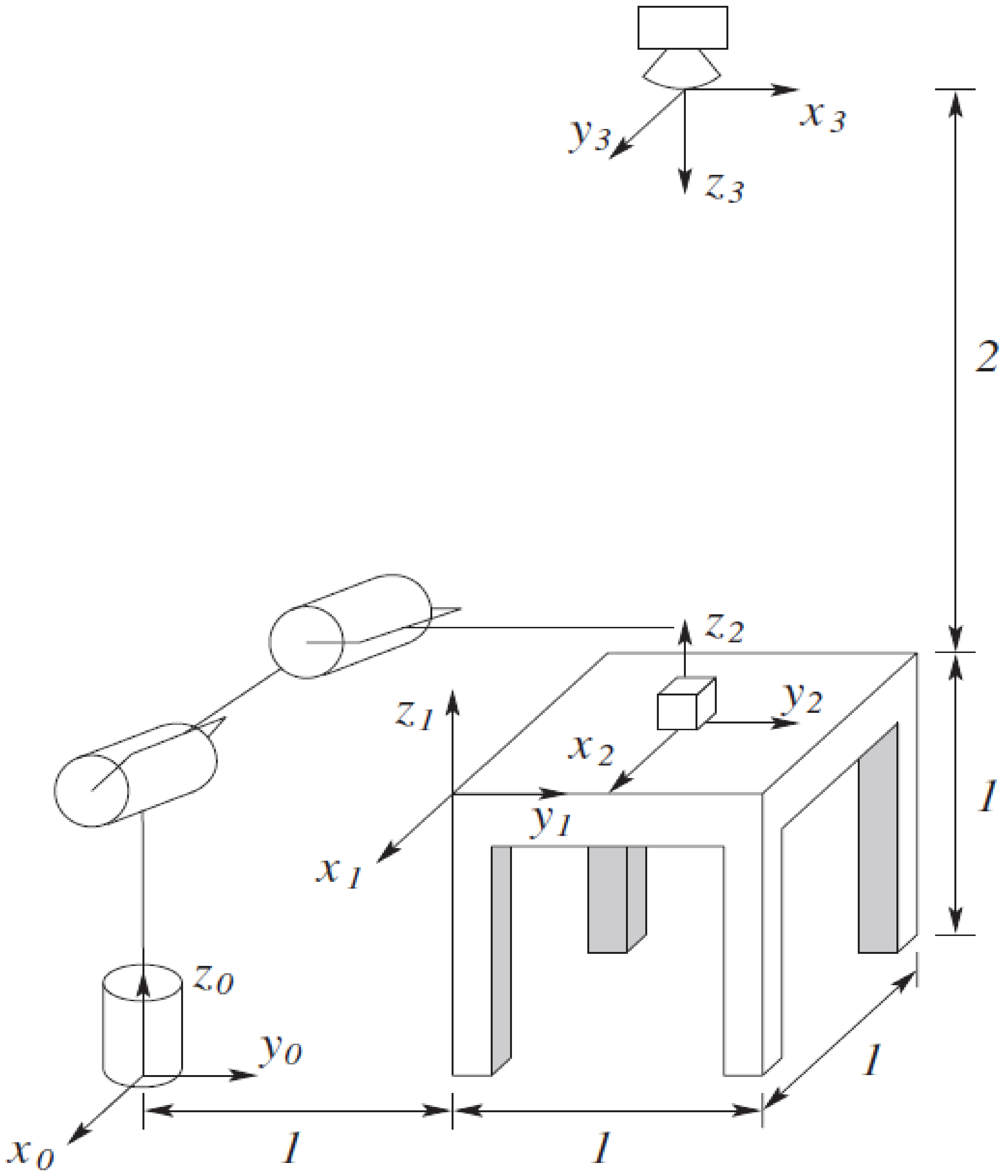
Symbolic Representation of Robot Manipulators

- Robot manipulators are composed of **links** connected by **joints** to form a kinematic chain.
- Joints are typically rotary (**revolute**) or linear (**prismatic**).
- We denote revolute joints by **R** and prismatic joints by **P.**
- The joint variables, denoted by $\boldsymbol{\theta}$ for a revolute joint and $\boldsymbol{d}$ for the prismatic joint



Symbolic Representation of robot joints

Introduction



The Questions here:

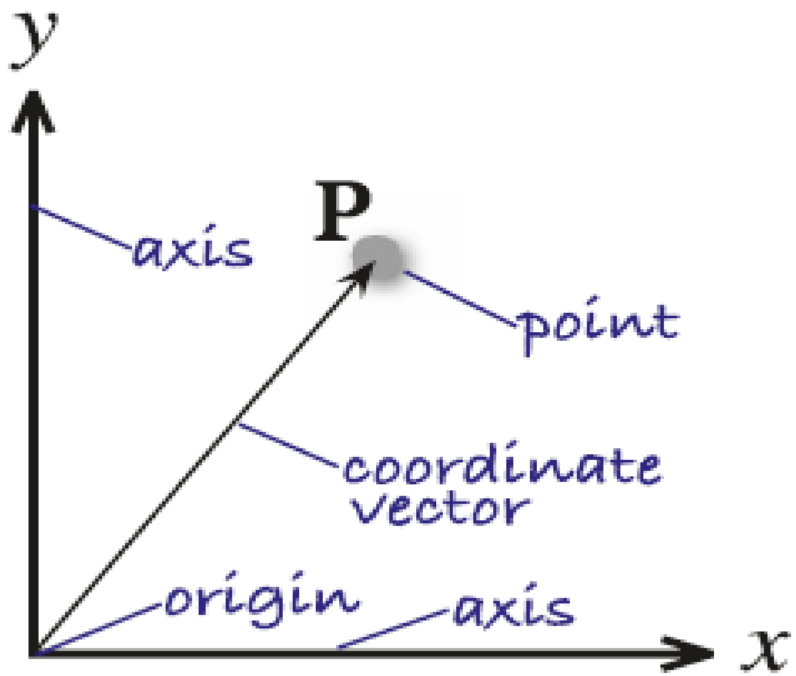
1- How to relate the motion of endeffector to robot base?

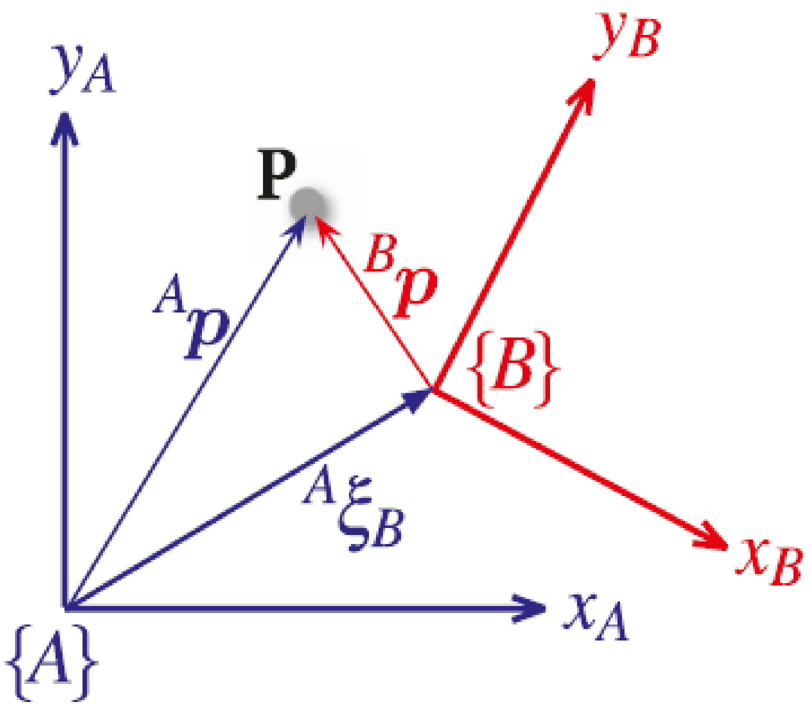
2- How to relate the locations of table, cube block and camera to the robot base?

3- How to relate the location of cube block to the camera frame?

Representing Positions

- **Analytic Geometric Reasoning:** represents entities using coordinates or equations, and reasoning is performed via algebraic manipulations.
- A **coordinate frame** consists of an origin (a single point in space), and two or three orthogonal coordinate axes for two- and three-dimensional spaces, respectively.





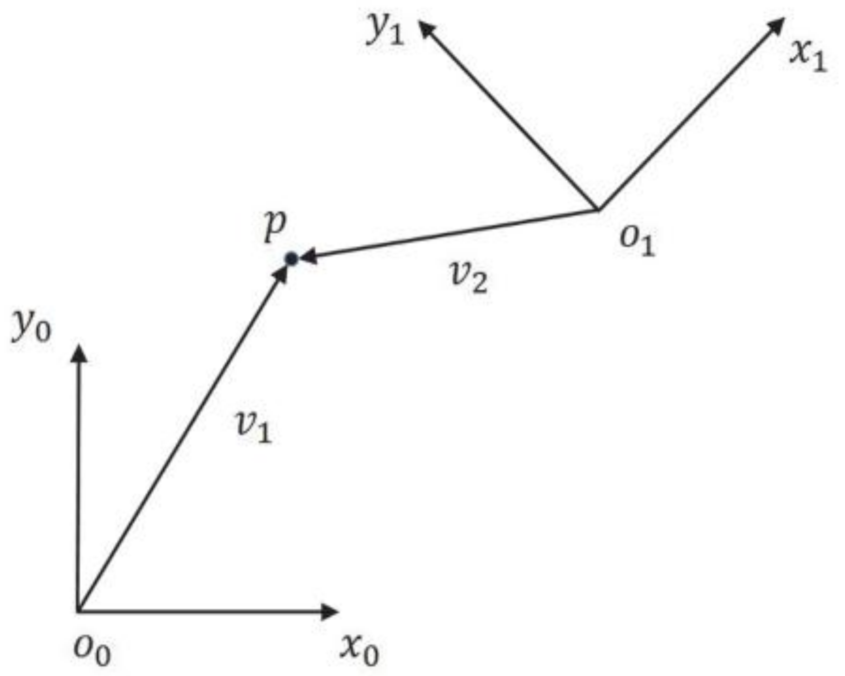
?

$$^A p = {}^A\xi_B \cdot {}^B p$$

A point is described by a bound coordinate vector that represents its displacement from the origin of a reference coordinate system.

Representing Positions

- Two coordinate frames, a point $\boldsymbol{p}$, and two vectors $\boldsymbol{v_1}$ and $\boldsymbol{v_2}$.
- Superscript is used to denote the **reference frame**.



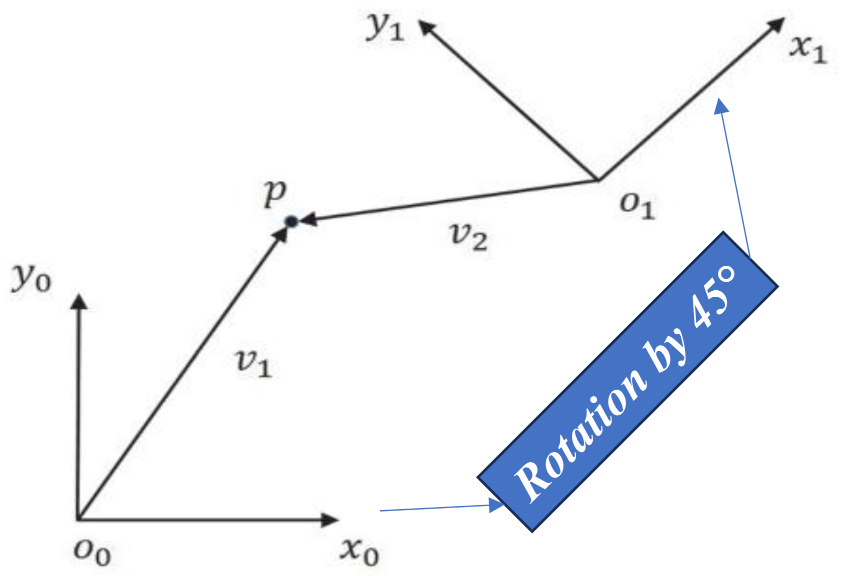
$$p^0 = \begin{bmatrix} 5 \\ 6 \end{bmatrix}, \qquad P^1 = \begin{bmatrix} -3 \\ 2 \end{bmatrix}$$

$$o_1^0 = \begin{bmatrix} 12 \\ 8 \end{bmatrix}, \qquad o_0^1 = \begin{bmatrix} -14 \\ 2.8 \end{bmatrix}$$

- $\boldsymbol{p^0}$ and $\boldsymbol{p^1}$ are coordinate vectors that represent the location of this point in space with respect to coordinate frames $\boldsymbol{o_0x_0y_0}$ (***Frame 0***) and $\boldsymbol{o_1x_1y_1}$ ***(Frame 1)***, respectively.
- $\boldsymbol{O_1^0}$ species the coordinates of the point $\boldsymbol{o_1}$ relative to frame 0 and $\boldsymbol{O_0^1}$ species the coordinates of the point $\boldsymbol{0_0}$ relative to frame 1.

Representing Positions

- While a point corresponds to a specific location in space, a vector species a direction and a magnitude.
- Vectors can be used to represent displacements or forces.



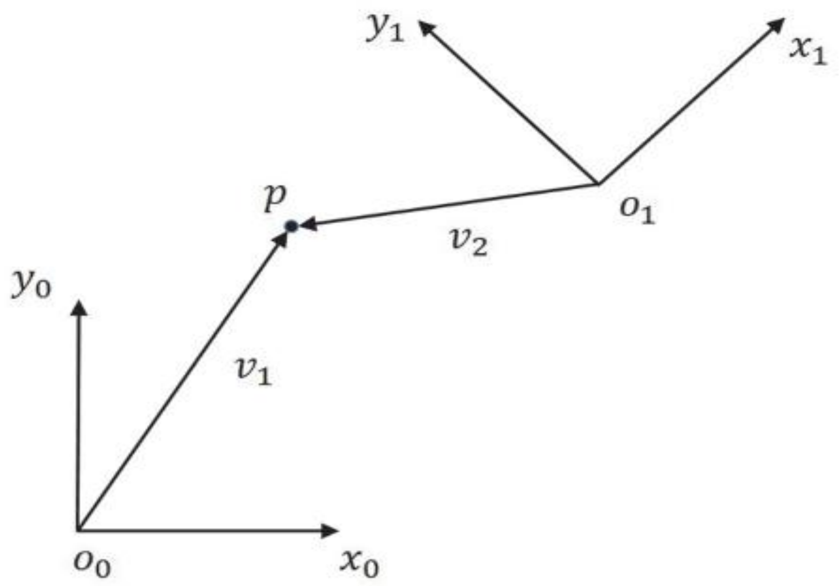
$$v_1^0 = \begin{bmatrix} 5 \\ 6 \end{bmatrix} \qquad v_1^1 = \begin{bmatrix} 7.8 \\ 0.7 \end{bmatrix}$$

$$v_2^0 = \begin{bmatrix} -3.5 \\ -0.7 \end{bmatrix} \qquad v_2^1 = \begin{bmatrix} -3 \\ 2 \end{bmatrix}$$

- While the point ***p*** is not equivalent to the vector $\boldsymbol{v_1}$, the displacement from the origin $\boldsymbol{o_0}$ to the point ***p*** is given by the vector $\boldsymbol{v_1}$.
- **Free vectors** are not constrained to be located at a particular point in space.
- Points refer to specific locations in space, but a free vector can be moved to any location in space.

Representing Positions

- It is essential that all coordinate vectors be defined with respect to the same coordinate frame.
- In the case of free vectors, it is enough that they be defined with respect to **"parallel"** coordinate frames, that is, frames whose respective coordinate axes are parallel, since only their magnitude and direction are specified and not their absolute locations in space.



$$\boldsymbol{v}_1^0 = \begin{bmatrix} 5 \\ 6 \end{bmatrix} \qquad \boldsymbol{v}_1^1 = \begin{bmatrix} 7.8 \\ 0.7 \end{bmatrix}$$

$$\boldsymbol{v}_2^0 = \begin{bmatrix} -3.5 \\ -0.7 \end{bmatrix} \qquad \boldsymbol{v}_2^1 = \begin{bmatrix} -3 \\ 2 \end{bmatrix}$$

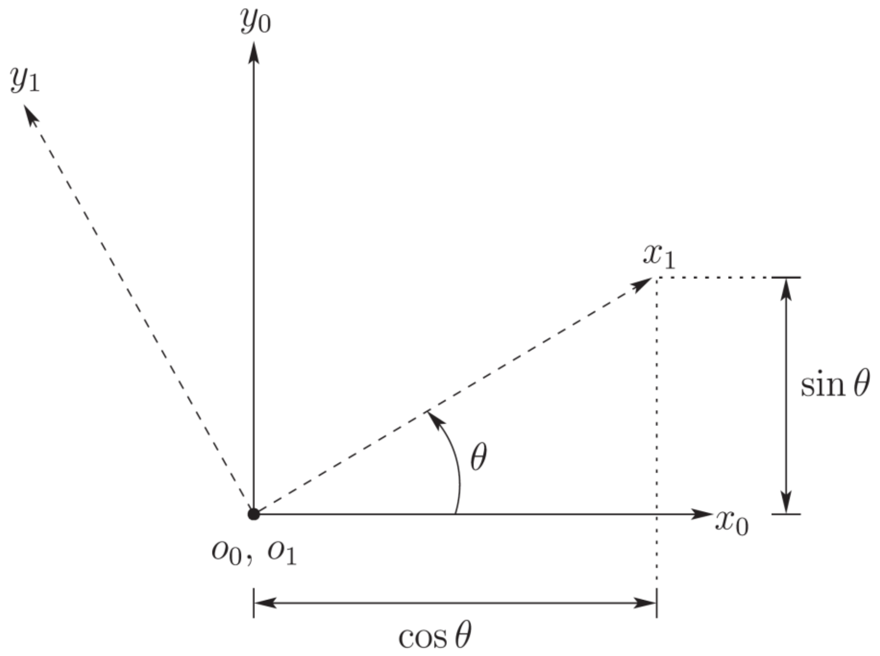
- Frames 0 and 1 are not parallel.
- Thus, we see a clear need not only for a representation system that allows points to be expressed with respect to various coordinate frames, but also for a mechanism that allows us to transform the coordinates of points from one coordinate frame to another.

Representing Rotations (2D)

- In order to represent the relative position and orientation of one rigid body with respect to another, we attach coordinate frames to each body, and then specify the geometric relationship between these coordinate frames.
- x_1^0 and y_1^0 are the coordinates in frame 0 of unit vectors $\boldsymbol{x_1}$ and $\boldsymbol{y_1}$, respectively.

R_1^0 is a matrix whose column vectors are the coordinates of the unit vectors along the axes of frame 1 expressed relative to frame 0.

$$R_1^0 = \left[x_1^0 \mid y_1^0\right]$$



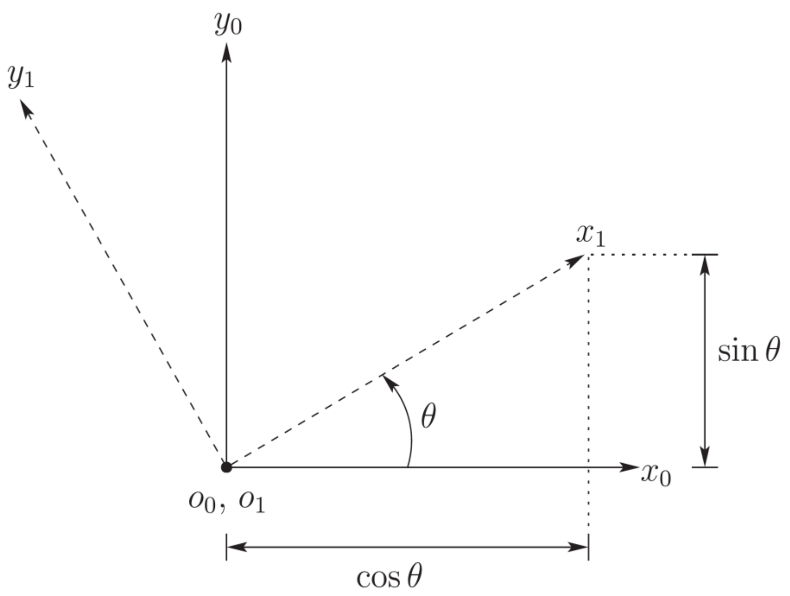
$$x_1^0 = \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix}, \qquad y_1^0 = \begin{bmatrix} -\sin\theta \\ \cos\theta \end{bmatrix}$$

$$R_1^0 = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

Rotation Matrix

Representing Rotations (2D)

Alternative Approach



$$R_1^0 = [x_1^0 \mid y_1^0]$$

$$x_1^0 = \begin{bmatrix} x_1 \cdot x_0 \\ x_1 \cdot y_0 \end{bmatrix}, \qquad y_1^0 = \begin{bmatrix} y_1 \cdot x_0 \\ y_1 \cdot y_0 \end{bmatrix}$$

$$R_1^0 = \begin{bmatrix} x_1 \cdot x_0 & y_1 \cdot x_0 \\ x_1 \cdot y_0 & y_1 \cdot y_0 \end{bmatrix}$$

$$R_0^1 = \begin{bmatrix} x_0 \cdot x_1 & y_0 \cdot x_1 \\ x_0 \cdot y_1 & y_0 \cdot y_1 \end{bmatrix} \longrightarrow R_0^1 = (R_1^0)^T \longrightarrow (R_1^0)^T = (R_1^0)^{-1} \longrightarrow (R_1^0)^T R_1^0 = I$$

$$x_i \cdot y_j = y_j \cdot x_i$$

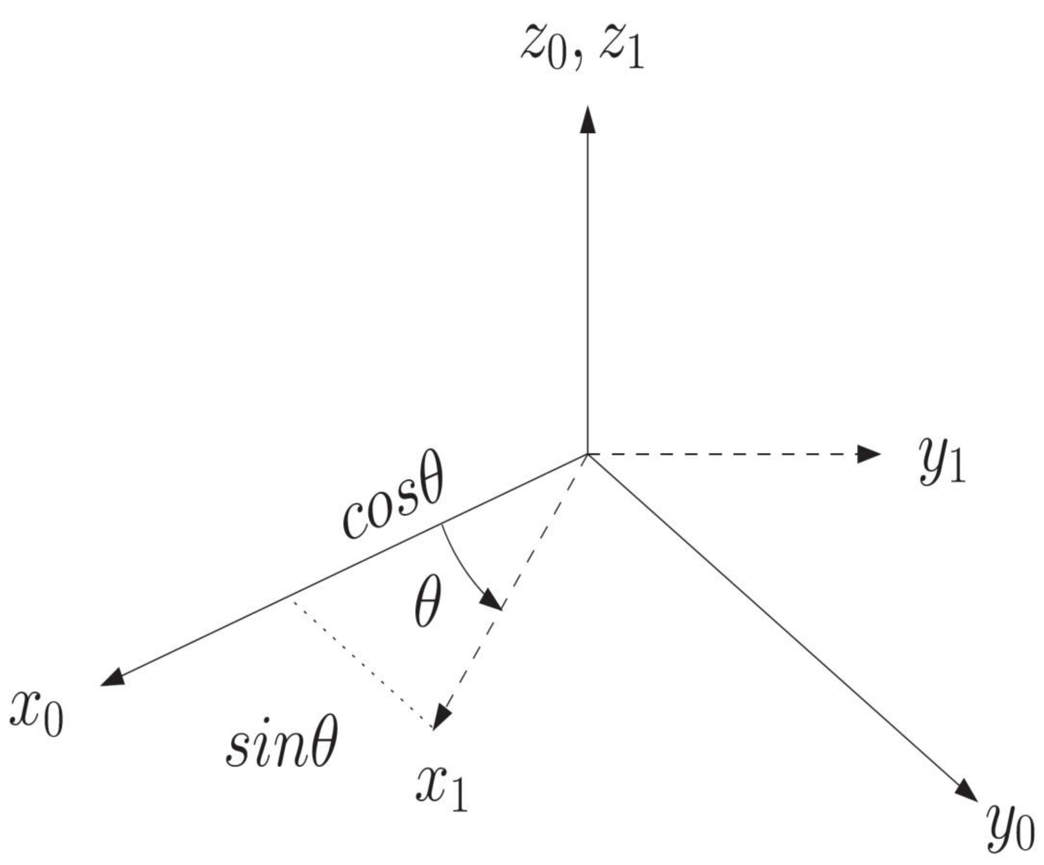
Coordinate axes are mutually orthogonal

Representing Rotations (2D)

Note that in the two-dimensional case, the inverse of the rotation matrix corresponding to a rotation by angle θ can also be easily computed simply by constructing the rotation matrix for a rotation by the angle −θ.

$$\begin{bmatrix} \cos(-\theta) & -\sin(-\theta) \\ \sin(-\theta) & \cos(-\theta) \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}^T$$

Representing Rotations (3D)



$$R_1^0 = \begin{bmatrix} x_1 \cdot x_0 & y_1 \cdot x_0 & z_1 \cdot x_0 \\ x_1 \cdot y_0 & y_1 \cdot y_0 & z_1 \cdot y_0 \\ x_1 \cdot z_0 & y_1 \cdot z_0 & z_1 \cdot z_0 \end{bmatrix}$$

$$x_1 \cdot x_0 = \cos\theta, \qquad y_1 \cdot x_0 = -\sin\theta,$$

$$x_1 \cdot y_0 = \sin\theta, \qquad y_1 \cdot y_0 = \cos\theta$$

$$z_0 \cdot z_1 = 1$$

Basic Rotation Matrix →

$$R_1^0 = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} = R_{z,\theta}$$

Representing Rotations (3D)

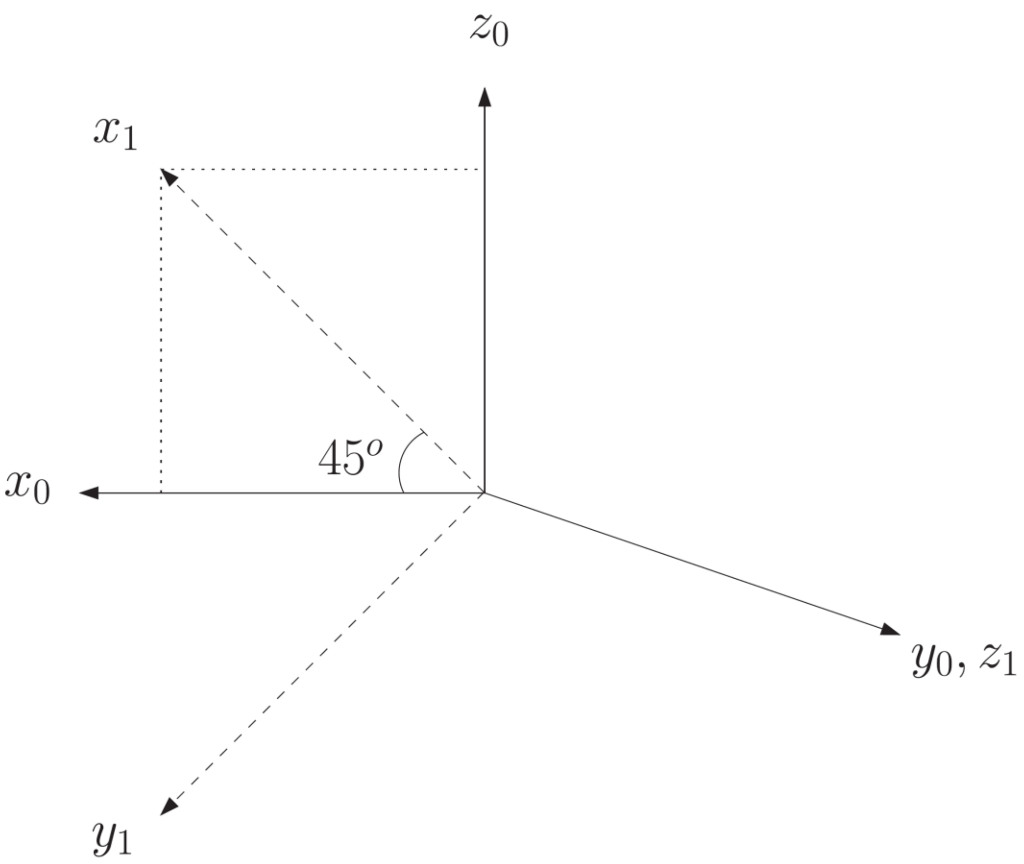
Similarly, the basic rotation matrices representing rotations about the x and y-axes:

$$R_{x,\theta} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$$

$$R_{y,\theta} = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

Representing Rotations (3D)

Example



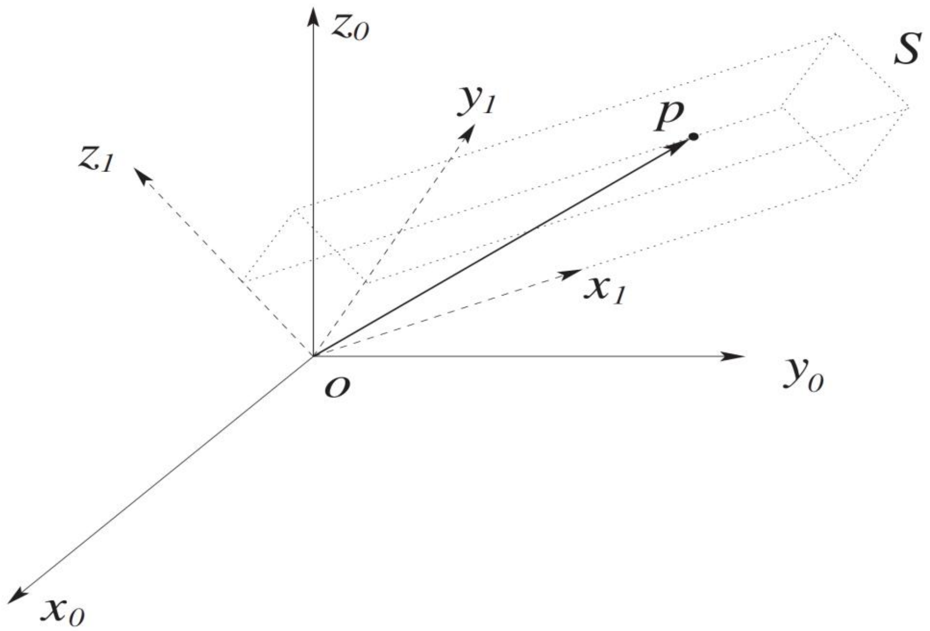
$$x_1^0 = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ \frac{1}{\sqrt{2}} \end{bmatrix}, \quad y_1^0 = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ \frac{-1}{\sqrt{2}} \end{bmatrix}, \quad z_1^0 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

$$R_1^0 = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 1 \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} & 0 \end{bmatrix}$$

Rotational Transformations

Coordinate frame attached to a rigid body

$$p^1 = (u, v, w) \longrightarrow p^1 = ux_1 + vy_1 + wz_1$$



$$p^0 = \begin{bmatrix} p \cdot x_0 \\ p \cdot y_0 \\ p \cdot z_0 \end{bmatrix}$$

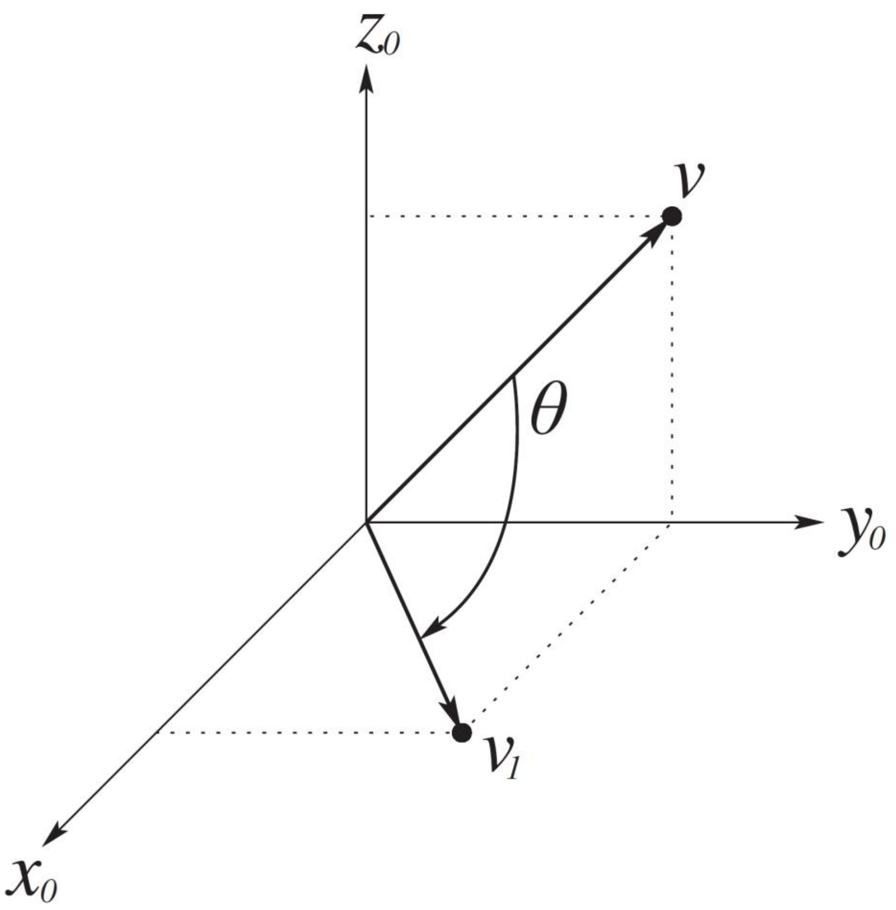
$$p^0 = \begin{bmatrix} (ux_1 + vy_1 + wz_1) \cdot x_0 \\ (ux_1 + vy_1 + wz_1) \cdot y_0 \\ (ux_1 + vy_1 + wz_1) \cdot z_0 \end{bmatrix} = \begin{bmatrix} ux_1 \cdot x_0 + vy_1 \cdot x_0 + wz_1 \cdot x_0 \\ ux_1 \cdot y_0 + vy_1 \cdot y_0 + wz_1 \cdot y_0 \\ ux_1 \cdot z_0 + vy_1 \cdot z_0 + wz_1 \cdot z_0 \end{bmatrix} = \begin{bmatrix} x_1 \cdot x_0 & y_1 \cdot x_0 & z_1 \cdot x_0 \\ x_1 \cdot y_0 & y_1 \cdot y_0 & z_1 \cdot y_0 \\ x_1 \cdot z_0 & y_1 \cdot z_0 & z_1 \cdot z_0 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix}$$

Thus, the rotation matrix R_1^0 can be used not only to represent the orientation of coordinate frame 1 with respect to frame 0, but also to transform the coordinates of a point from one frame to another.

$$p^0 = R_1^0 p^1$$

Rotational Transformations

Example



The vector v with coordinates $v_0 = (0; 1; 1)$ is rotated about y_0 by $\frac{\pi}{2}$

$$
\begin{aligned}
v_1^0 &= R_{y,\frac{\pi}{2}} v^0 \\
&= \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}
\end{aligned}
$$

Composition of Rotations

Rotation with Respect to the Current Frame

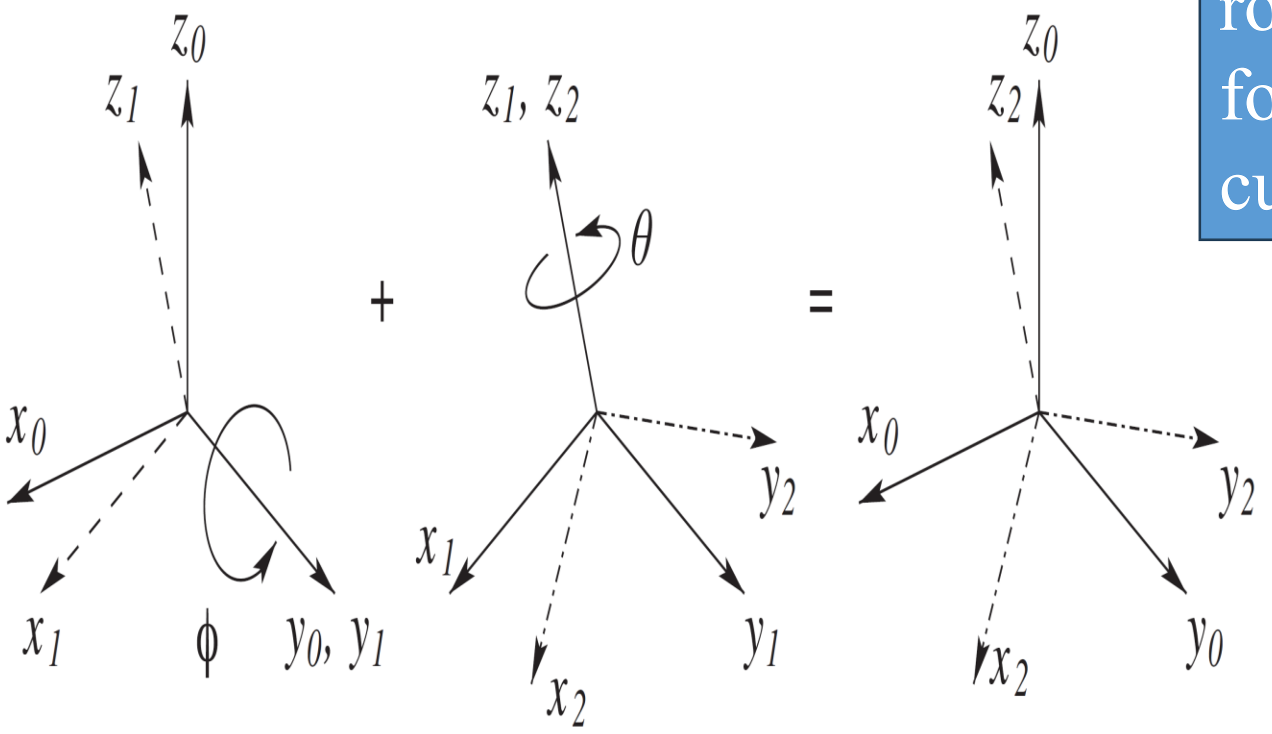
$$
\begin{aligned}
p^0 &= R_1^0 p^1 \\
p^1 &= R_2^1 p^2 \\
p^0 &= R_2^0 p^2
\end{aligned}
$$

$$p^0 = R_1^0 R_2^1 p^2$$

$$R_2^0 = R_1^0 R_2^1$$

Composition of Rotations

Example



Post-multiply

Suppose a rotation matrix ***R*** represents a rotation of angle ϕ about the current y-axis followed by a rotation of angle θ about the current z-axis.

$$
\begin{aligned}
R &= R_{y,\phi} R_{z,\theta} \\
&= \begin{bmatrix} c_\phi & 0 & s_\phi \\ 0 & 1 & 0 \\ -s_\phi & 0 & c_\phi \end{bmatrix} \begin{bmatrix} c_\theta & -s_\theta & 0 \\ s_\theta & c_\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} c_\phi c_\theta & -c_\phi s_\theta & s_\phi \\ s_\theta & c_\theta & 0 \\ -s_\phi c_\theta & s_\phi s_\theta & c_\phi \end{bmatrix}
\end{aligned}
$$

Composition of Rotations

Example

Suppose that the previous rotations are performed in the reverse order, that is, first a rotation about the current z-axis followed by a rotation about the current y-axis. Then the resulting rotation matrix is given by

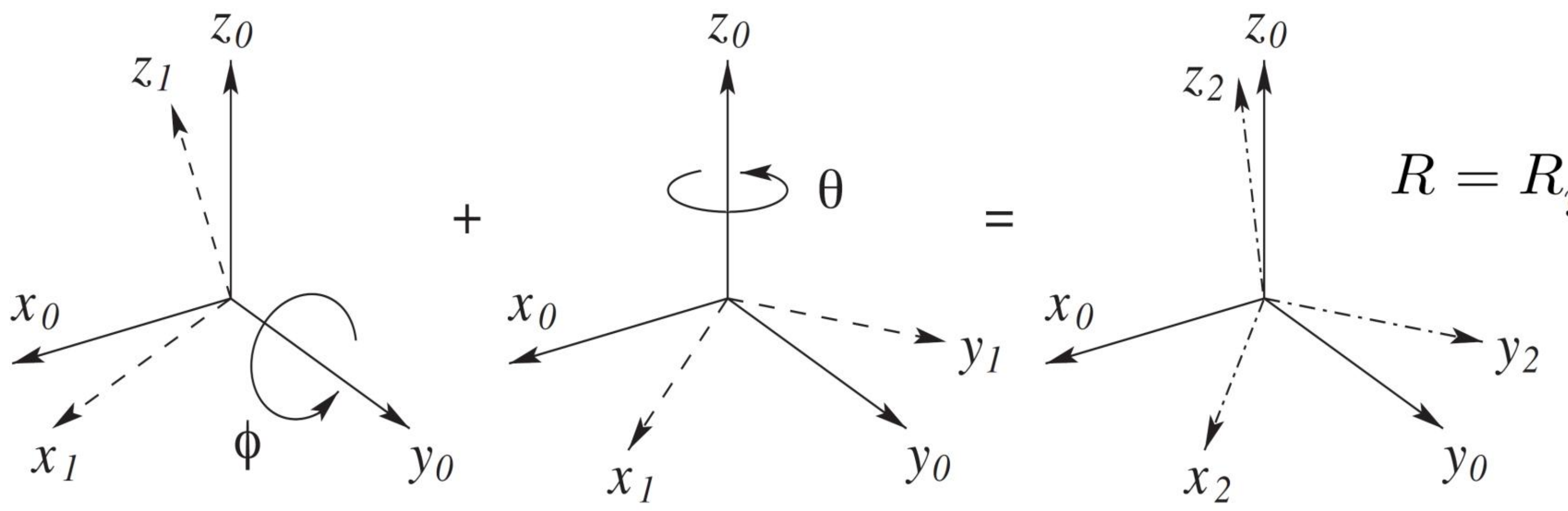
$$
\begin{aligned}
R' &= R_{z,\theta}R_{y,\phi} \\
&= \begin{bmatrix} c_\theta & -s_\theta & 0 \\ s_\theta & c_\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_\phi & 0 & s_\phi \\ 0 & 1 & 0 \\ -s_\phi & 0 & c_\phi \end{bmatrix} \\
&= \begin{bmatrix} c_\theta c_\phi & -s_\theta & c_\theta s_\phi \\ s_\theta c_\phi & c_\theta & s_\theta s_\phi \\ -s_\phi & 0 & c_\phi \end{bmatrix}
\end{aligned}
$$

we see that $R \neq R'$

Composition of Rotations

Rotation with Respect to the Fixed Frame

Example



$$R = R_{y,\phi}\left[R_{y,-\phi}R_{z,\theta}R_{y,\phi}\right] = R_{z,\theta}R_{y,\phi}$$

Thus, when a rotation R is performed with respect to the world coordinate frame, the current rotation matrix is **pre-multiplied** by R to obtain the desired rotation matrix.

Composition of Rotations

Example

Suppose ***R*** is defined by the following sequence of basic rotations in the order specified:

1. *A rotation of θ about the current x-axis*
2. *A rotation of ϕ about the current z-axis*
3. *A rotation of α about the fixed z-axis*
4. *A rotation of β about the current y-axis*
5. *A rotation of δ about the fixed x-axis*

→ $R = R_{x,\delta} R_{z,\alpha} R_{x,\theta} R_{z,\phi} R_{y,\beta}$

Parameterizations of Rotations

Euler Angles

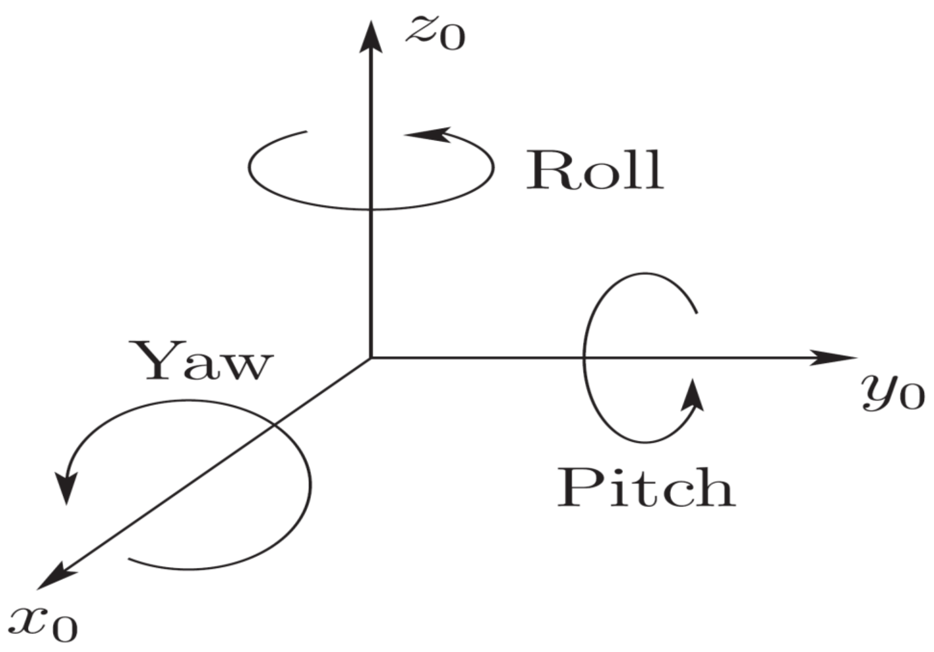
First rotate about the z-axis by the angle ϕ.
Next rotate about the current y-axis by the angle θ.
Finally rotate about the current z-axis by the angle ψ.

$$
\begin{aligned}
R_{ZYZ} &= R_{z,\phi} R_{y,\theta} R_{z,\psi} \\
&= \begin{bmatrix} c_\phi & -s_\phi & 0 \\ s_\phi & c_\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_\theta & 0 & s_\theta \\ 0 & 1 & 0 \\ -s_\theta & 0 & c_\theta \end{bmatrix} \begin{bmatrix} c_\psi & -s_\psi & 0 \\ s_\psi & c_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} c_\phi c_\theta c_\psi - s_\phi s_\psi & -c_\phi c_\theta s_\psi - s_\phi c_\psi & c_\phi s_\theta \\ s_\phi c_\theta c_\psi + c_\phi s_\psi & -s_\phi c_\theta s_\psi + c_\phi c_\psi & s_\phi s_\theta \\ -s_\theta c_\psi & s_\theta s_\psi & c_\theta \end{bmatrix}
\end{aligned}
$$

Parameterizations of Rotations

Roll, Pitch, Yaw Angles

We specify the order of rotation as x-y-z:
first a yaw about x_0 through an angle ψ
then pitch about the y_0 by an angle θ
finally roll about the z_0 by an angle ϕ



$$
\begin{aligned}
R &= R_{z,\phi}R_{y,\theta}R_{x,\psi} \\
&= \begin{bmatrix} c_\phi & -s_\phi & 0 \\ s_\phi & c_\phi & 0 \\ 0 & 0 & 1 \end{bmatrix}
\begin{bmatrix} c_\theta & 0 & s_\theta \\ 0 & 1 & 0 \\ -s_\theta & 0 & c_\theta \end{bmatrix}
\begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\psi & -s_\psi \\ 0 & s_\psi & c_\psi \end{bmatrix} \\
&= \begin{bmatrix} c_\phi c_\theta & -s_\phi c_\psi + c_\phi s_\theta s_\psi & s_\phi s_\psi + c_\phi s_\theta c_\psi \\ s_\phi c_\theta & c_\phi c_\psi + s_\phi s_\theta s_\psi & -c_\phi s_\psi + s_\phi s_\theta c_\psi \\ -s_\theta & c_\theta s_\psi & c_\theta c_\psi \end{bmatrix}
\end{aligned}
$$

Homogeneous Transformations

A homogeneous transformation is a matrix representation of a rigid motion.

$$p^0 = R_1^0 p^1 + d^0$$

$$H = \begin{bmatrix} R & d \\ 0 & 1 \end{bmatrix} \longrightarrow H^{-1} = \begin{bmatrix} R^T & -R^T d \\ 0 & 1 \end{bmatrix}$$

$$P^0 = H_1^0 P^1$$

Homogeneous Transformations

$$\text{Trans}_{x,a} = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \text{Rot}_{x,\alpha} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_\alpha & -s_\alpha & 0 \\ 0 & s_\alpha & c_\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\text{Trans}_{y,b} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \text{Rot}_{y,\beta} = \begin{bmatrix} c_\beta & 0 & s_\beta & 0 \\ 0 & 1 & 0 & 0 \\ -s_\beta & 0 & c_\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\text{Trans}_{z,c} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \text{Rot}_{z,\gamma} = \begin{bmatrix} c_\gamma & -s_\gamma & 0 & 0 \\ s_\gamma & c_\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Homogeneous Transformations

The most general homogeneous transformation that we will consider may be written

$$H_1^0 = \begin{bmatrix} n_x & s_x & a_x & d_x \\ n_y & s_y & a_y & d_y \\ n_z & s_z & a_z & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} n & s & a & d \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- $n = (n_x; n_y; n_z)$ is a vector representing the direction of x_1 in the frame 0.
- $s = (s_x; s_y; s_z)$ represents the direction of y_1 in the frame 0.
- $a = (a_x; a_y; a_z)$ represents the direction of z_1 in the frame 0.
- The vector $d = (d_x; d_y; d_z)$ represents the vector from the origin o_0 to the origin o_1 expressed in the frame 0.

- The same interpretation regarding composition and ordering of transformations holds for 4x4 homogeneous transformations as for 3x3 rotations.

Homogeneous Transformations

Example

The homogeneous transformation matrix ***H*** that represents:

- a rotation by angle ***α*** about the current x-axis followed by
- a translation of ***b*** units along the current x-axis,
- followed by a translation of ***d*** units along the current z-axis,
- followed by a rotation by angle ***θ*** about the current z-axis,

$$H = Rot_{x,\alpha} Trans_{x,b} Trans_{z,d} Rot_{z,\theta}$$

$$= \begin{bmatrix} c_\theta & -s_\theta & 0 & b \\ c_\alpha s_\theta & c_\alpha c_\theta & -s_\alpha & -d s_\alpha \\ s_\alpha s_\theta & s_\alpha c_\theta & c_\alpha & d c_\alpha \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

MATLAB: Robotics System Toolbox

Coordinate Transformations

Axis-Angle

- A rotation in 3-D space described by a scalar rotation around a fixed axis defined by a vector.
- 1-by-3-unit vector and a scalar angle combined as a 1-by-4 vector.

Example

axang = [0 1 0 pi/2] ***a rotation of pi/2 radians around the y-axis***

MATLAB: Robotics System Toolbox

Coordinate Transformations

Euler Angles

- Euler angles are three angles that describe the orientation of a rigid body.
- Each angle is a scalar rotation around a given coordinate frame axis.
- The Robotics System Toolbox supports two rotation orders.
- The 'ZYZ' axis order is commonly used for robotics applications.
- It also support the '**ZYX**' axis order which is also denoted as **"Roll Pitch Yaw (rpy)".**
- 1-by-3 vector of scalar angles

Example

eul = [0 pi 0] ***a rotation of pi around the y -axis***

MATLAB: Robotics System Toolbox

Coordinate Transformations

Rotation Matrix

- A rotation matrix describes a rotation in 3-D space.
- It is a square, orthonormal matrix with a determinant of 1.
- 3-by-3 matrix

Example

rotm =

$$\begin{matrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{matrix}$$

a rotation of α degrees around the x-axis

MATLAB: Robotics System Toolbox

Coordinate Transformations

Translation Vector

- A translation vector is represented in 3-D Euclidean space as Cartesian coordinates.
- It only involves coordinate translation applied equally to all points.
- There is no rotation involved.
- 1-by-3 vector

Example

trvec = [3 0 2.5]

a translation by 3 units along the x -axis and 2.5 units along the z -axis

MATLAB: Robotics System Toolbox

Coordinate Transformations

Homogeneous Transformation Matrix

- A homogeneous transformation matrix combines a translation and rotation into one matrix.
- 4-by-4 matrix.

Example

tform =

$$\begin{matrix} \cos\alpha & 0 & \sin\alpha & 0 \\ 0 & 1 & 0 & 4 \\ -\sin\alpha & 0 & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$

a rotation of angle α around the y -axis and a translation of 4 units along the y -axis

MATLAB: Robotics System Toolbox

Coordinate Transformations

axang2rotm

- Convert axis-angle rotation (in rad) to rotation matrix

Example

```
axang = [0 1 0 pi/2];    %% Rotation around y-axis by 90°
rotm = axang2rotm(axang)
```

```
rotm = 3×3

    0.0000         0    1.0000
         0    1.0000         0
   -1.0000         0    0.0000
```

MATLAB: Robotics System Toolbox

Coordinate Transformations

axang2tform

- Convert axis-angle rotation (in rad) to homogeneous transformation

Example

```
axang = [1 0 0 pi/2];    %% Rotation around x-axis by 90°
tform = axang2tform(axang)


tform = 4×4
   1.0000        0         0         0
        0   0.0000   -1.0000         0
        0   1.0000    0.0000         0
        0        0         0    1.0000
```

MATLAB: Robotics System Toolbox

Coordinate Transformations

eul2rotm

- Convert Euler angles to rotation matrix
- rotm = eul2rotm(eul,sequence) %% Sequence "ZYX”, "ZYZ", "XYZ"

Example

```
eul = [0 pi/2 pi/2];
rotmZYZ = eul2rotm(eul,'ZYZ')

rotmZYZ = 3×3
    0.0000   -0.0000    1.0000
    1.0000    0.0000         0
   -0.0000    1.0000    0.0000
```

MATLAB: Robotics System Toolbox

Coordinate Transformations

eul2tform

- Convert Euler angles to homogeneous transformation
- tform = eul2tform(eul,sequence) %% Sequence "ZYX”, "ZYZ", "XYZ"

Example

```
eul = [0 pi/2 pi/2];
tformZYZ = eul2tform(eul,'ZYZ')

tformZYZ = 4×4
    0.0000   -0.0000    1.0000         0
    1.0000    0.0000         0         0
   -0.0000    1.0000    0.0000         0
         0         0         0    1.0000
```

MATLAB: Robotics System Toolbox

Coordinate Transformations

Example

The homogeneous transformation matrix ***H*** that represents:

- a rotation by angle α about the current x-axis followed by
- a translation of ***b*** units along the current x-axis,
- followed by a translation of ***d*** units along the current z-axis,
- followed by a rotation by angle θ about the current z-axis,

$$
\begin{aligned}
H &= Rot_{x,\alpha} Trans_{x,b} Trans_{z,d} Rot_{z,\theta} \\
&= \begin{bmatrix} c_\theta & -s_\theta & 0 & b \\ c_\alpha s_\theta & c_\alpha c_\theta & -s_\alpha & -d s_\alpha \\ s_\alpha s_\theta & s_\alpha c_\theta & c_\alpha & d c_\alpha \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned}
$$

```
alpha=pi/2; theta=pi/3; d=0.5; b=0.2;
H1=axang2tform([1 0 0 alpha])
H2=trvec2tform([b 0 0])
H3=trvec2tform([0 0 d])
H4=axang2tform([0 0 1 theta])
H=H1*H2*H3*H4
```

MATLAB: Robotics System Toolbox

Coordinate Transformations

Additional Commands

rotm2axang	Convert rotation matrix to axis-angle rotation
rotm2eul	Convert rotation matrix to Euler angles
rotm2tform	Convert rotation matrix to homogeneous transformation
tform2axang	Convert homogeneous transformation to axis-angle rotation
tform2eul	Extract Euler angles from homogeneous transformation
tform2rotm	Extract rotation matrix from homogeneous transformation
tform2trvec	Extract translation vector from homogeneous transformation
trvec2tform	Convert translation vector to homogeneous transformation

Thank You

Any Questions ??