# CS 4610/5335 – Lecture 4
# Forward and inverse kinematics

Lawson L.S. Wong
Northeastern University
1/31/22

Material adapted from:
1. Robert Platt, CS 4610/5335
2. Peter Corke, Robotics, Vision and Control
3. Oussama Khatib, Stanford CS 223A

# Announcements

- Ex1 posted, due 2/11

- First project “session” will be during lecture on 2/2 (Wed)

# Announcements

- Ex1 posted, due 2/11

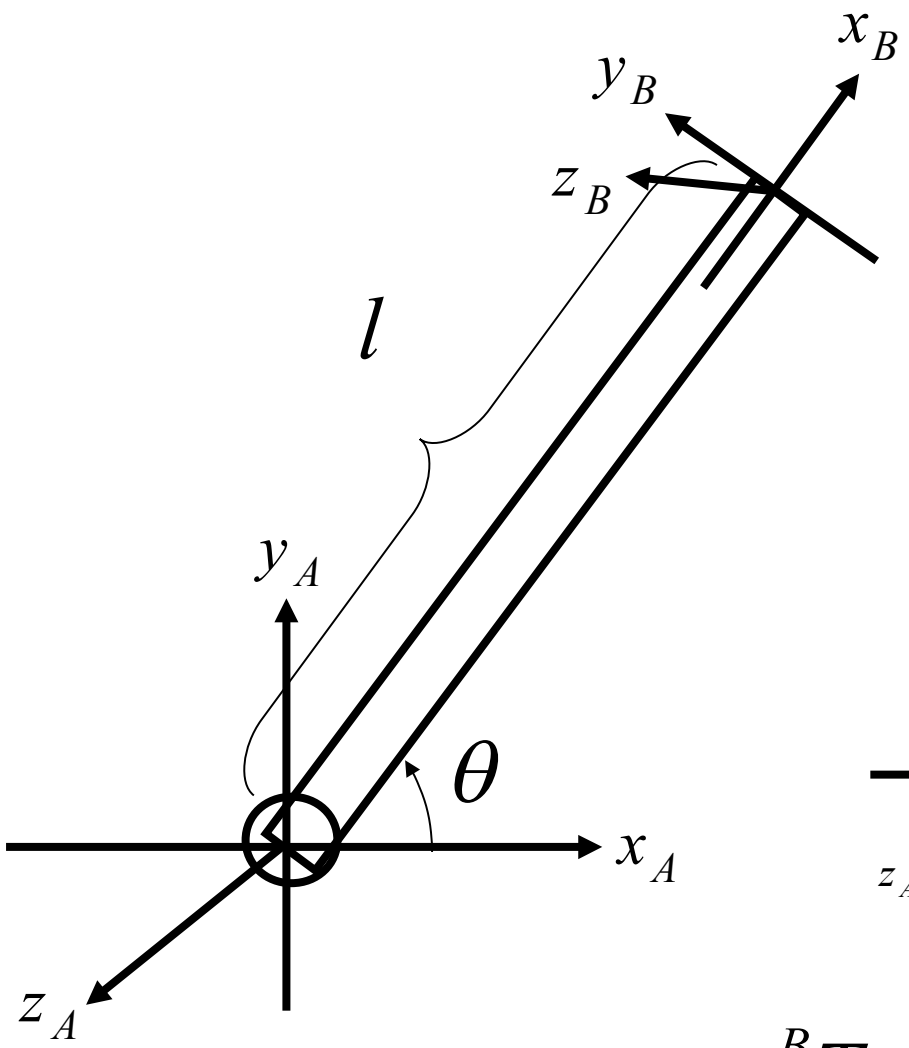
- First project “session” will be during lecture on 2/2 (Wed)

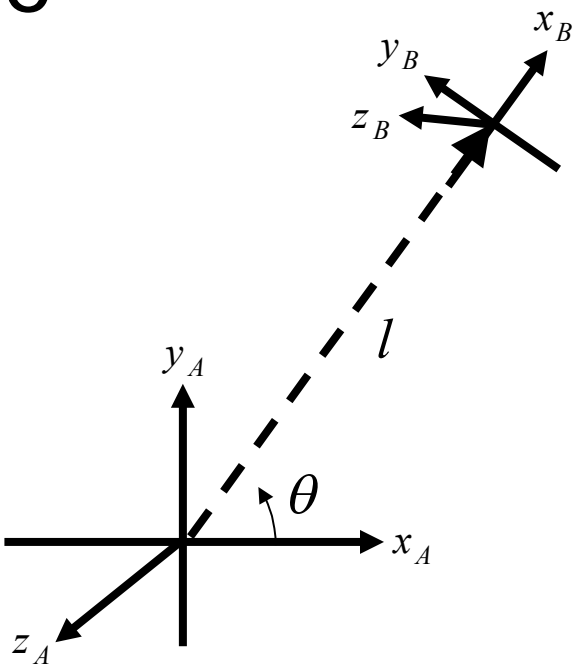
    - Think about what you want to work on
        - What task(s), requiring what type of robot
        - Arm-only, wheeled-only, wheeled+arm, etc.
        - Also, level of sophistication in platform

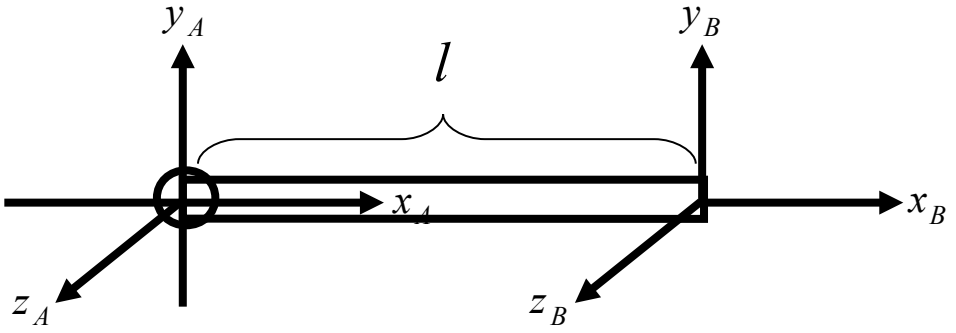
    - Task(s) should involve perception, reasoning, motion
    - There should be ways to make task(s) easier/harder

    - Any of the following situations are can work:
        - You have a great idea: Looking for members
        - You don't have concrete ideas yet: Joining a team
        - You have already formed a team

# Recap: Example



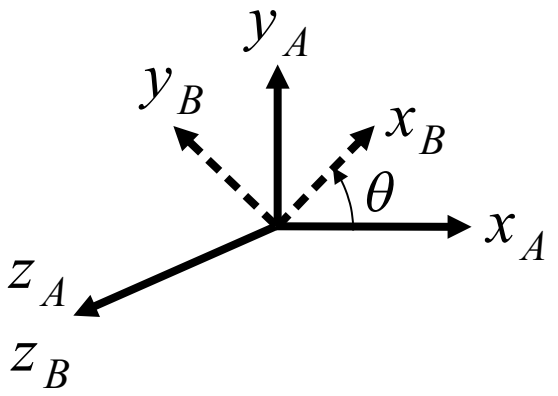




Find ${}^{B}T_{A}$

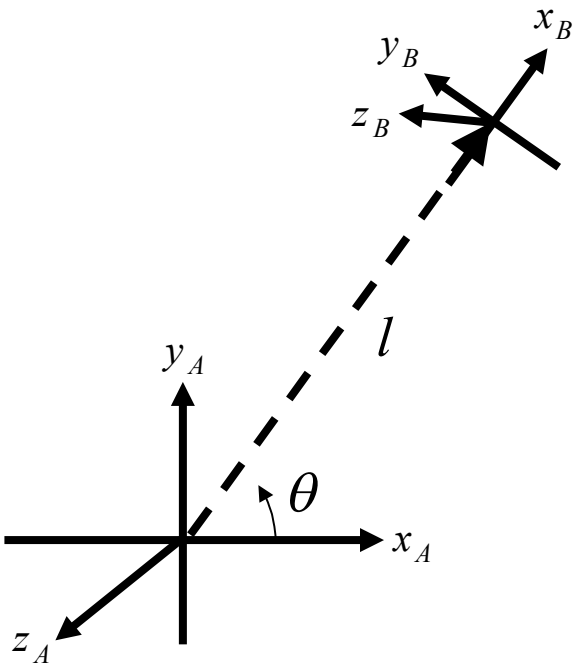
# Recap: Example

Find ${}^{B}T_{A}$



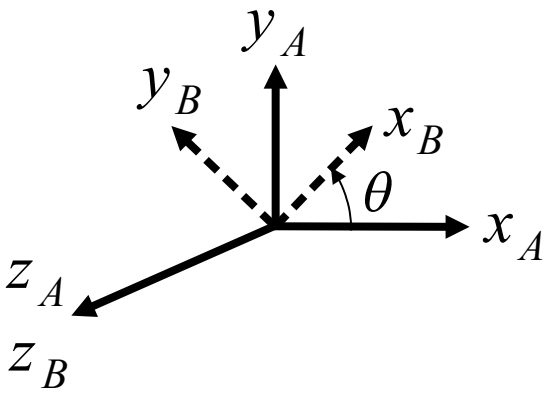

$${}^{A}R_{B} = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$${}^{B}d_{A} = \begin{pmatrix} -l \\ 0 \\ 0 \end{pmatrix}$$



# Recap: Example

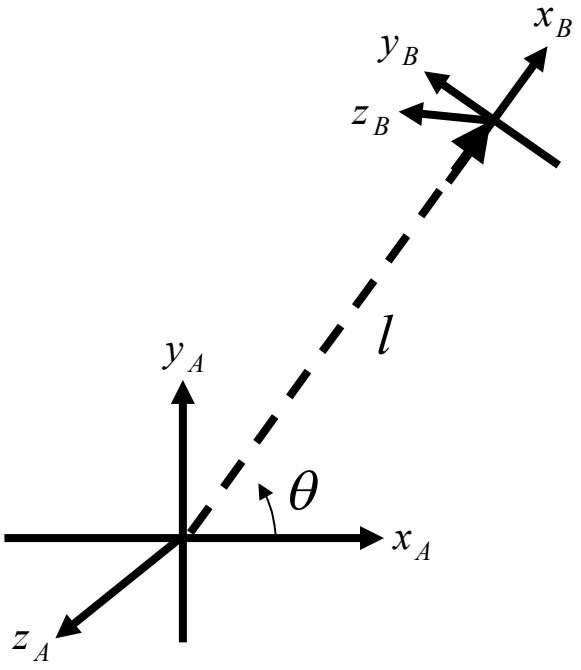
Find ${}^{B}T_{A}$



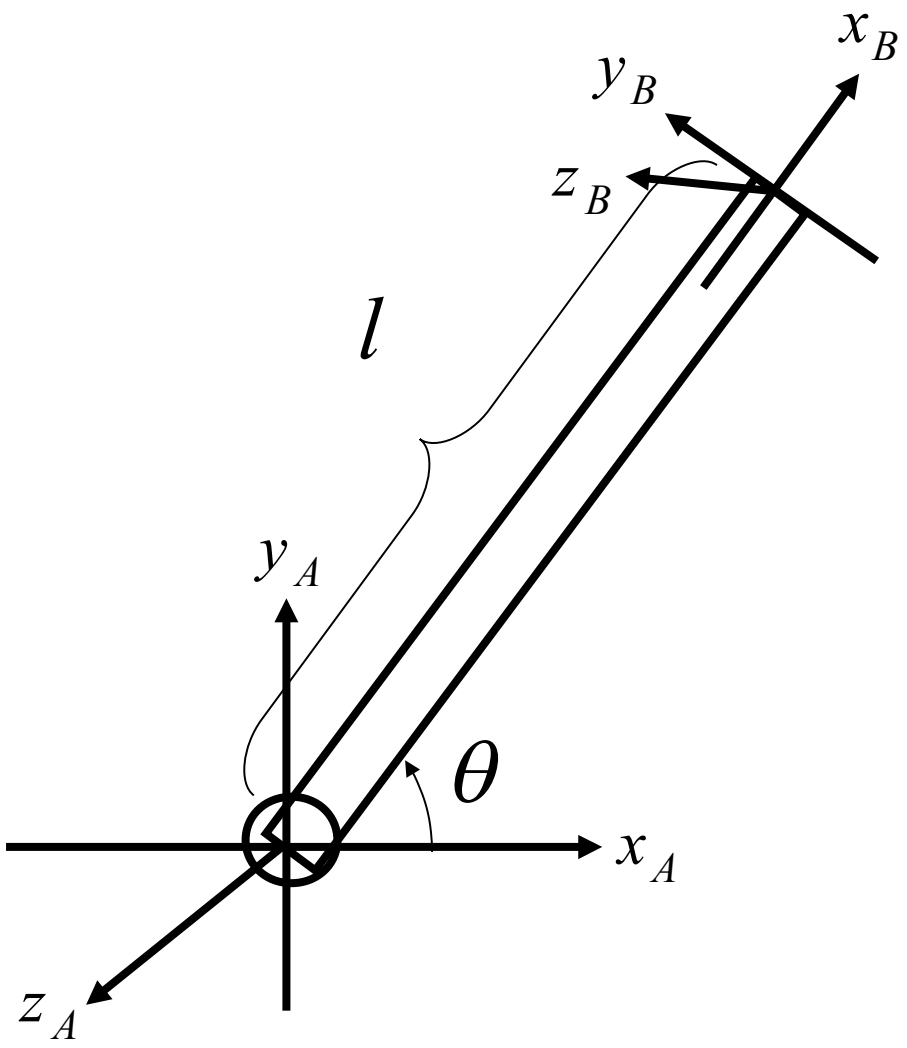

$${}^{A}R_{B} = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$${}^{B}d_{A} = \begin{pmatrix} -l \\ 0 \\ 0 \end{pmatrix} \qquad {}^{B}T_{A} = \begin{pmatrix} {}^{B}R_{A} & {}^{B}d_{A} \\ 0 & 1 \end{pmatrix}$$

$${}^{B}T_{A} = \begin{pmatrix} \cos(\theta) & \sin(\theta) & 0 & -l \\ -\sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



# Recap: Example




$$
{}^{B}T_{A} = \begin{pmatrix} \cos(\theta) & \sin(\theta) & 0 & -l \\ -\sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}
$$

How about ${}^{A}T_{B}$?

# Recap: Example

$$ {}^{B}T_{A}^{-1} = \begin{pmatrix} {}^{B}R_{A}^{T} & -{}^{B}R_{A}^{T}\,{}^{B}d_{A} \\ 0 & 1 \end{pmatrix} $$

# Recap: Example

$$ {}^{B}T_{A}^{-1} = \begin{pmatrix} {}^{B}R_{A}^{T} & -{}^{B}R_{A}^{T}\,{}^{B}d_{A} \\ 0 & 1 \end{pmatrix} \qquad {}^{A}R_{B} = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} $$

# Recap: Example

$$ {}^{B}T_{A}^{-1} = \begin{pmatrix} {}^{B}R_{A}^{T} & -{}^{B}R_{A}^{T}\,{}^{B}d_{A} \\ 0 & 1 \end{pmatrix} \qquad {}^{A}R_{B} = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} $$

$$ {}^{B}d_{A} = \begin{pmatrix} -l \\ 0 \\ 0 \end{pmatrix} \qquad -{}^{A}R_{B}\,{}^{B}d_{A} = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} -l \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} l\cos(\theta) \\ l\sin(\theta) \\ 0 \end{pmatrix} $$

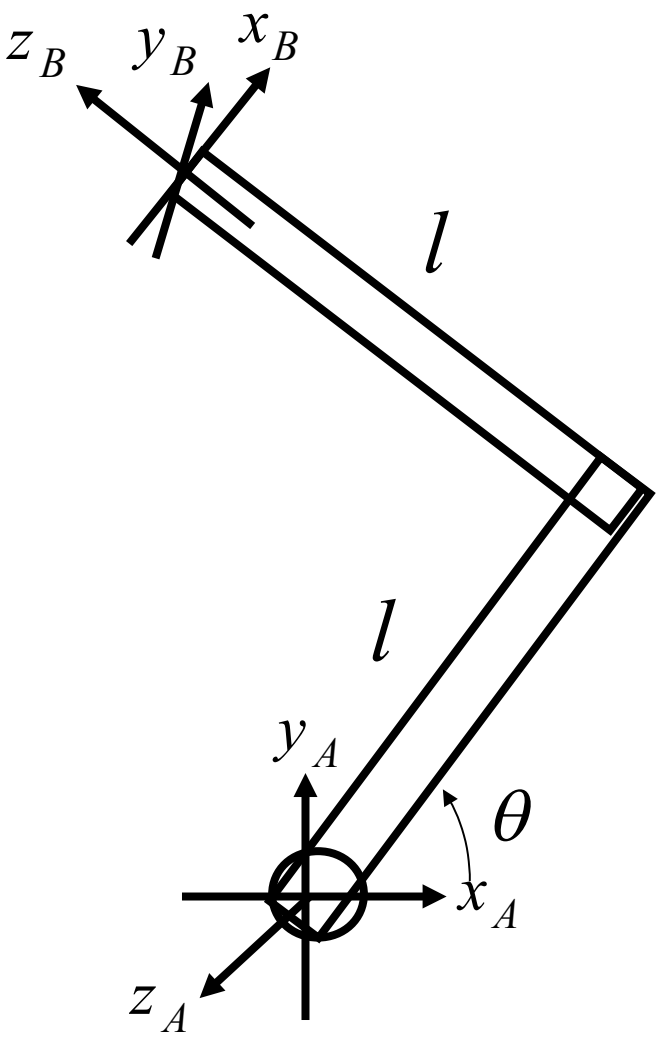
# Recap: Example

$$ {}^{B}T_{A}^{-1} = \begin{pmatrix} {}^{B}R_{A}^{T} & -{}^{B}R_{A}^{T}\,{}^{B}d_{A} \\ 0 & 1 \end{pmatrix} \qquad {}^{A}R_{B} = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} $$

$$ {}^{B}d_{A} = \begin{pmatrix} -l \\ 0 \\ 0 \end{pmatrix} \qquad -{}^{A}R_{B}\,{}^{B}d_{A} = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} -l \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} l\cos(\theta) \\ l\sin(\theta) \\ 0 \end{pmatrix} $$

$$ {}^{B}T_{A}^{-1} = {}^{A}T_{B} = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 & l\cos(\theta) \\ \sin(\theta) & \cos(\theta) & 0 & l\sin(\theta) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

# Try this at home




This arm rotates about the $z_A$ axis

Find ${}^{A}T_{B}$ and ${}^{B}T_{A}$

# Summary: 3-D transformation

$$\begin{pmatrix} {}^{A}x \\ {}^{A}y \\ {}^{A}z \\ 1 \end{pmatrix} = \begin{pmatrix} {}^{A}\boldsymbol{R}_{B} & \boldsymbol{t} \\ \boldsymbol{0}_{1\times 3} & 1 \end{pmatrix} \begin{pmatrix} {}^{B}x \\ {}^{B}y \\ {}^{B}z \\ 1 \end{pmatrix}$$

$$\begin{aligned} {}^{A}\tilde{\boldsymbol{p}} &= \begin{pmatrix} {}^{A}\boldsymbol{R}_{B} & \boldsymbol{t} \\ \boldsymbol{0}_{1\times 3} & 1 \end{pmatrix} {}^{B}\tilde{\boldsymbol{p}} \\ &= {}^{A}\boldsymbol{T}_{B}\, {}^{B}\tilde{\boldsymbol{p}} \end{aligned}$$

A concrete representation of relative pose is $\xi \sim T \in \mathbf{SE}(3)$ and $T_1 \oplus T_2 \mapsto T_1 T_2$ which is standard matrix multiplication.

$$T_1 T_2 = \begin{pmatrix} \boldsymbol{R}_1 & \boldsymbol{t}_1 \\ \boldsymbol{0}_{1\times 3} & 1 \end{pmatrix} \begin{pmatrix} \boldsymbol{R}_2 & \boldsymbol{t}_2 \\ \boldsymbol{0}_{1\times 3} & 1 \end{pmatrix} = \begin{pmatrix} \boldsymbol{R}_1\boldsymbol{R}_2 & \boldsymbol{t}_1 + \boldsymbol{R}_1\boldsymbol{t}_2 \\ \boldsymbol{0}_{1\times 3} & 1 \end{pmatrix} \tag{2.24}$$

One of the rules of pose algebra from page 21 is $\xi \oplus 0 = \xi$. For matrices we know that $TI = T$, where $I$ is the identify matrix, so for pose $0 \mapsto I$ the identity matrix. Another rule of pose algebra was that $\xi \ominus \xi = 0$. We know for matrices that $TT^{-1} = I$ which implies that $\ominus T \mapsto T^{-1}$

$$T^{-1} = \begin{pmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0}_{1\times 3} & 1 \end{pmatrix}^{-1} = \begin{pmatrix} \boldsymbol{R}^{T} & -\boldsymbol{R}^{T}\boldsymbol{t} \\ \boldsymbol{0}_{1\times 3} & 1 \end{pmatrix} \tag{2.25}$$

# Outline

- ✔ SO(2): 2-D rotation / orientation
- ✔ SE(2): 2-D transformations
- ✔ SO(3): 3-D rotation / orientation
- ✔ SE(3): 3-D transformations

More representations for 3-D rotations
- Euler angles
- Axis-angle representations
- Quaternions

Forward kinematics (FK)

Inverse kinematics (IK)

Jacobians + more IK (time permitting)

# Euler angles / Cardan angles

## Orientation in 3-Dimensions

*Any two independent orthonormal coordinate frames*
*can be related by a sequence of rotations (not more than **three**)*
*about coordinate axes, where no two successive rotations may be about the same axis.*
Euler's rotation theorem (Kuipers 1999).

# Euler angles / Cardan angles

The Eulerian type involves repetition, but not successive, of rotations about one particular axis: XYX, XZX, YXY, YZY, ZXZ, or ZYZ. The Cardanian type is characterized by rotations about all three axes: XYZ, XZY, YZX, YXZ, ZXY, or ZYX.

It is common practice to refer to all 3-angle representations as Euler angles but this is underspecified since there are twelve different types to choose from. The particular angle sequence is often a convention within a particular technological field.

The ZYZ sequence

$$\boldsymbol{R} = \boldsymbol{R}_z(\phi)\boldsymbol{R}_y(\theta)\boldsymbol{R}_z(\psi) \tag{2.14}$$

is commonly used in aeronautics and mechanical dynamics, and is used in the Toolbox. The Euler angles are the 3-vector $\boldsymbol{\Gamma} = (\phi, \theta, \psi)$.

# ZYZ Euler angles

$$r_{zyz} = \begin{pmatrix} \phi \\ \theta \\ \psi \end{pmatrix}$$

To get from *A* to *B:*

Rotate $\phi$ about *z* axis

*Then* rotate $\theta$ about y axis

*Then* rotate $\psi$ about z axis

$$R_z(\phi) = \begin{pmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$R_y(\theta) = \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix}$$

$$R_z(\psi) = \begin{pmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

# ZYZ Euler angles

Remember that $R_z(\phi)\ R_y(\theta)\ R_z(\psi)$ encodes the desired rotation in the pre-rotation reference frame:

$$R_z(\phi) = {}^{pre-rotation}R_{post-rotation}$$

Therefore, the sequence of rotations is concatentated as follows:

$$R_{zyz}(\phi,\theta,\psi) = R_z(\phi)R_y(\theta)R_z(\psi)$$

$$R_{zyz}(\phi,\theta,\psi) = \begin{pmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} \begin{pmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$R_{zyz}(\phi,\theta,\psi) = \begin{pmatrix} c_\phi c_\theta c_\psi - s_\phi s_\psi & -c_\phi c_\theta s_\psi - s_\phi c_\psi & c_\phi s_\theta \\ s_\phi c_\theta c_\psi + c_\phi s_\psi & -s_\phi c_\theta s_\psi + c_\phi c_\psi & s_\phi s_\theta \\ -s_\theta c_\psi & s_\theta s_\psi & c_\theta \end{pmatrix}$$

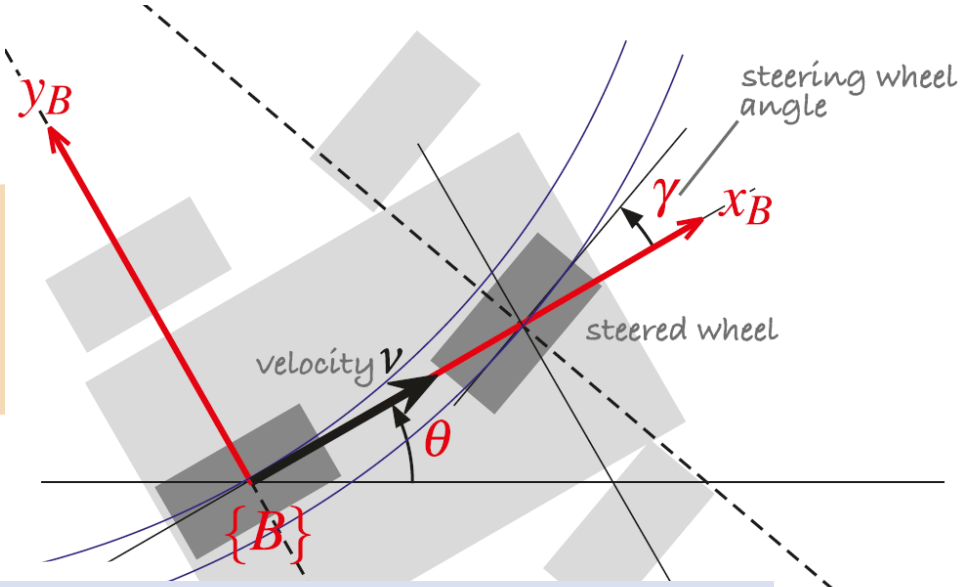
# Roll-pitch-yaw (RPY)

## Orientation in 3-Dimensions

*Any two independent orthonormal coordinate frames*
*can be related by a sequence of rotations (not more than **three**)*
*about coordinate axes, where no two successive rotations may be about the same axis.*
Euler's rotation theorem (Kuipers 1999).

leads to the ZYX angle sequence

$$\boldsymbol{R} = \boldsymbol{R}_z(\theta_y)\boldsymbol{R}_y(\theta_p)\boldsymbol{R}_x(\theta_r)$$




**Vehicle coordinate system.** The coordinate system that we will use, and a common one for vehicles of all sorts is that the $x$-axis is forward (longitudinal motion), the $y$-axis is to the left side (lateral motion) which implies that the $z$-axis is upward. For aerospace and underwater applications the $z$-axis is often downward and the $x$-axis is forward.

# Roll-pitch-yaw (RPY)

To get from *A* to *B:*

Rotate $\phi$ about *z* axis

*Then* rotate $\theta$ about y axis

*Then* rotate $\psi$ about x axis

$$R_z(\phi)=\begin{pmatrix}\cos\phi & -\sin\phi & 0\\ \sin\phi & \cos\phi & 0\\ 0 & 0 & 1\end{pmatrix}$$

$$R_y(\theta)=\begin{pmatrix}\cos\theta & 0 & \sin\theta\\ 0 & 1 & 0\\ -\sin\theta & 0 & \cos\theta\end{pmatrix}$$

$$R_x(\psi)=\begin{pmatrix}1 & 0 & 0\\ 0 & \cos\psi & -\sin\psi\\ 0 & \sin\psi & \cos\psi\end{pmatrix}$$

$$R_{zyx}(\phi,\theta,\psi)=R_z(\phi)R_y(\theta)R_x(\psi)$$

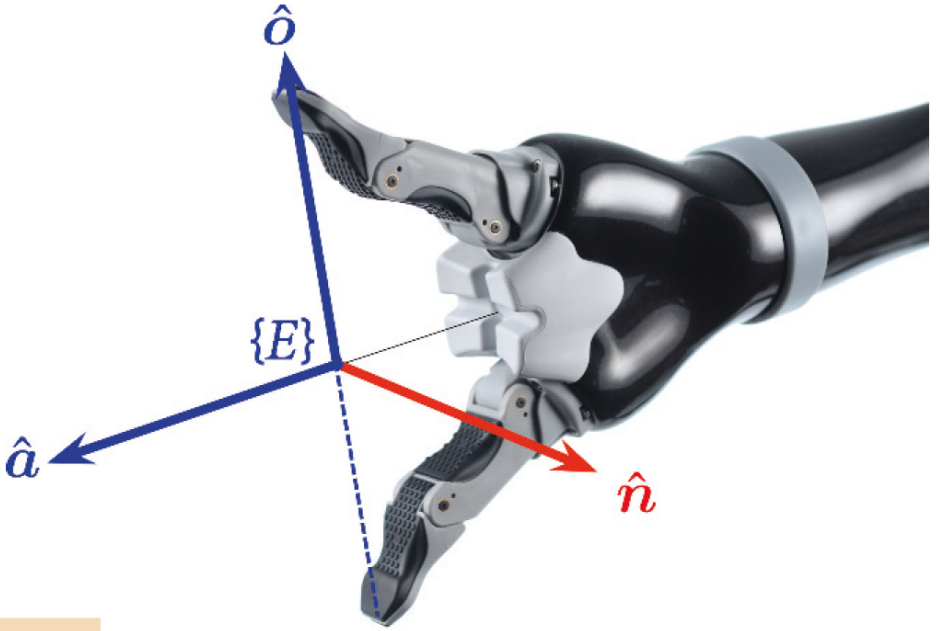
$$R_{zyx}(\phi,\theta,\psi)=\begin{pmatrix}\cos\phi & -\sin\phi & 0\\ \sin\phi & \cos\phi & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}\cos\theta & 0 & \sin\theta\\ 0 & 1 & 0\\ -\sin\theta & 0 & \cos\theta\end{pmatrix}\begin{pmatrix}1 & 0 & 0\\ 0 & \cos\psi & -\sin\psi\\ 0 & \sin\psi & \cos\psi\end{pmatrix}$$

# Roll-pitch-yaw (RPY)

## Orientation in 3-Dimensions

*Any two independent orthonormal coordinate frames can be related by a sequence of rotations (not more than **three**) about coordinate axes, where no two successive rotations may be about the same axis.*
Euler's rotation theorem (Kuipers 1999).

**Fig. 2.16.**
Robot end-effector coordinate system defines the pose in terms of an *approach* vector $\hat{a}$ and an *orientation* vector $\hat{o}$, from which $\hat{n}$ can be computed. $\hat{n}$, $\hat{o}$ and $\hat{a}$ vectors correspond to the $x$-, $y$- and $z$-axes respectively of the end-effector coordinate frame. (courtesy of Kinova Robotics)




to the XYZ angle sequence

$$\boldsymbol{R} = \boldsymbol{R}_x\left(\theta_y\right)\boldsymbol{R}_y\left(\theta_p\right)\boldsymbol{R}_z\left(\theta_r\right)$$

# Axis-angle representation

Euler's rotation theorem (1776):

*Theorema. Quomodocunque sphaera circa centrum suum conuertatur, semper assignari potest diameter, cuius directio in situ translato conueniat cum situ initiali.*

https://en.wikipedia.org/wiki/Euler%27s_rotation_theorem

# Axis-angle representation

Euler's rotation theorem (1776):

*Theorema. Quomodocunque sphaera circa centrum suum conuertatur, semper assignari potest diameter, cuius directio in situ translato conueniat cum situ initiali.*

“When a sphere is moved around its centre
it is always possible to find a diameter
whose direction in the displaced position
is the same as in the initial position.”

In 3-D space, any two Cartesian coordinate systems
with a common origin are related
by a rotation about some fixed axis.

https://en.wikipedia.org/wiki/Euler%27s_rotation_theorem

# Axis-angle representation

Two coordinate frames of arbitrary orientation are related by a *single* rotation about some axis in space.

# Axis-angle representation

Two coordinate frames of arbitrary orientation are related by a *single* rotation about some axis in space.

From the definition of eigenvalues and eigenvectors we recall that

$$\boldsymbol{R}\boldsymbol{v} = \lambda\boldsymbol{v}$$

where $\boldsymbol{v}$ is the eigenvector corresponding to the eigenvalue $\lambda$. For the case $\lambda = 1$

$$\boldsymbol{R}\boldsymbol{v} = \boldsymbol{v}$$

which implies that the corresponding eigenvector $\boldsymbol{v}$ is *unchanged* by the rotation.

# Axis-angle representation

Two coordinate frames of arbitrary orientation are related by a *single* rotation about some axis in space.

From the definition of eigenvalues and eigenvectors we recall that

$$\boldsymbol{R}\boldsymbol{v} = \lambda\boldsymbol{v}$$

where $\boldsymbol{v}$ is the eigenvector corresponding to the eigenvalue $\lambda$. For the case $\lambda = 1$

$$\boldsymbol{R}\boldsymbol{v} = \boldsymbol{v}$$

which implies that the corresponding eigenvector $\boldsymbol{v}$ is *unchanged* by the rotation. There is only one such vector and that is the one *about which* the rotation occurs. In the example the third eigenvalue is equal to one, so the rotation axis is the third column of x.

An orthonormal rotation matrix will always have one real eigenvalue at $\lambda = 1$ and in general a complex pair $\lambda = \cos\theta \pm i \sin\theta$ where $\theta$ is the rotation angle.

# Rotation matrix → Axis-angle

```
>> R = rpy2r(0.1 , 0.2, 0.3);
```

we can determine such an angle and vector by

```
>> [theta, v] = tr2angvec(R)
th =
    0.3655
v =
    0.1886    0.5834    0.7900
```

where `theta` is the angle of rotation and `v` is the vector around which the rotation occurs.

This information is encoded in the eigenvalues and eigenvectors of $\boldsymbol{R}$. Using the built-in MATLAB function `eig`

```
>> [x,e] = eig(R)
x =
  -0.6944 + 0.0000i  -0.6944 + 0.0000i   0.1886 + 0.0000i
   0.0792 + 0.5688i   0.0792 - 0.5688i   0.5834 + 0.0000i
   0.1073 - 0.4200i   0.1073 + 0.4200i   0.7900 + 0.0000i
e =
   0.9339 + 0.3574i   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.9339 - 0.3574i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i   1.0000 + 0.0000i
```

the eigenvalues are returned on the diagonal of the matrix `e` and the corresponding eigenvectors are the corresponding columns of `x`.

```
>> theta = angle(e(1,1))
theta =
    0.3655
```

# Axis-angle → Rotation matrix

The inverse problem, converting from angle and vector to a rotation matrix, is achieved using Rodrigues' rotation formula

$$\boldsymbol{R} = \boldsymbol{I}_{3\times 3} + \sin\theta[\hat{\boldsymbol{v}}]_\times + (1-\cos\theta)[\hat{\boldsymbol{v}}]_\times^2 \tag{2.18}$$

where $[\hat{v}]_\times$ is a skew-symmetric matrix.

# Axis-angle → Rotation matrix

The inverse problem, converting from angle and vector to a rotation matrix, is achieved using Rodrigues' rotation formula

$$\boldsymbol{R} = \boldsymbol{I}_{3\times 3} + \sin\theta[\hat{\boldsymbol{v}}]_\times + (1-\cos\theta)[\hat{\boldsymbol{v}}]_\times^2 \tag{2.18}$$

where $[\hat{v}]_\times$ is a skew-symmetric matrix.

In 3-dimensions the skew-symmetric matrix has the form

$$[\boldsymbol{\omega}]_\times = \begin{pmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{pmatrix}$$

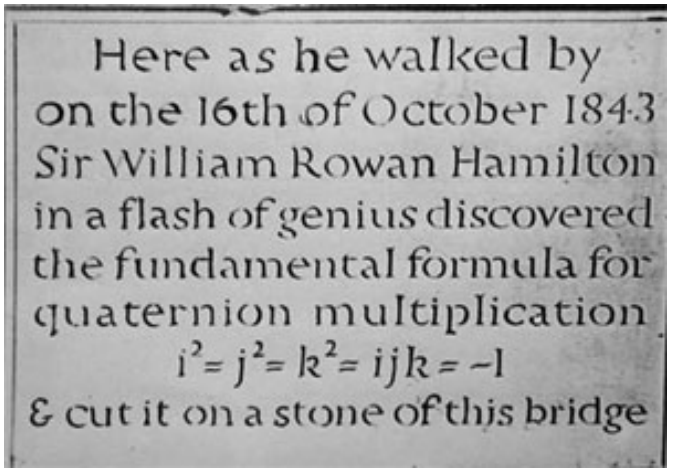
which has clear structure and only three unique elements $\boldsymbol{\omega} \in \mathbb{R}^3$. implement the vector cross product $\boldsymbol{v}_1 \times \boldsymbol{v}_2 = [\boldsymbol{v}_1]_\times \boldsymbol{v}_2$.

# Quaternions

## Unit Quaternions

*Quaternions came from Hamilton after his really good work had been done; and, though beautifully ingenious, have been an unmixed evil to those who have touched them in any way, including Clark Maxwell.*

Lord Kelvin, 1892



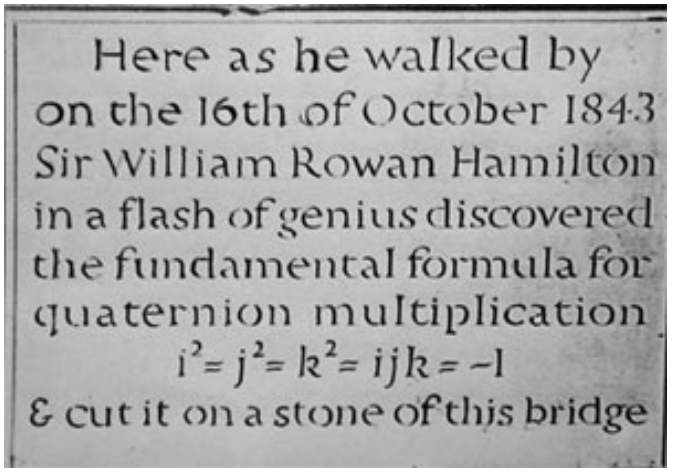
# Quaternions

## Unit Quaternions

*Quaternions came from Hamilton after his really good work had been done; and, though beautifully ingenious, have been an unmixed evil to those who have touched them in any way, including Clark Maxwell.*
Lord Kelvin, 1892

$$q = s + \boldsymbol{v}$$
$$= s + v_1 i + v_2 j + v_3 k$$

$$i^2 = j^2 = k^2 = ijk = -1$$

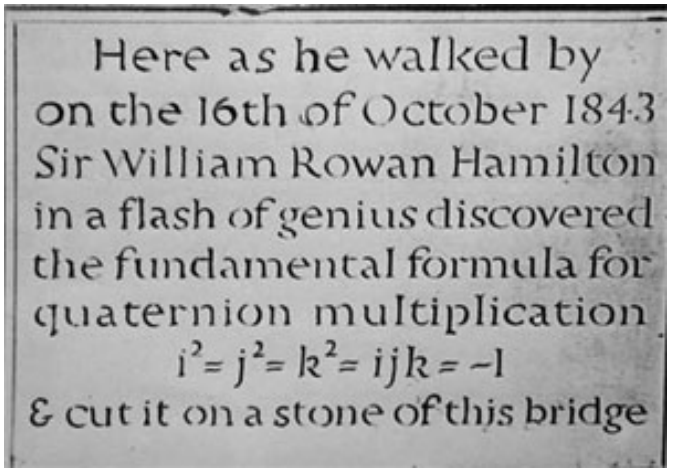


# Quaternions

## Unit Quaternions

*Quaternions came from Hamilton after his really good work had been done; and, though beautifully ingenious, have been an unmixed evil to those who have touched them in any way, including Clark Maxwell.*
Lord Kelvin, 1892

$$q = s + v$$
$$= s + v_1 i + v_2 j + v_3 k$$




$$i^2 = j^2 = k^2 = ijk = -1$$

$$q = s < v_1, v_2, v_3 >$$

$$q_1 \circ q_2 = s_1 s_2 - v_1 \cdot v_2 < s_1 v_2 + s_2 v_1 + v_1 \times v_2 >$$

# Quaternions

## Unit Quaternions

*Quaternions came from Hamilton after his really good work had been done; and, though beautifully ingenious, have been an unmixed evil to those who have touched them in any way, including Clark Maxwell.*
Lord Kelvin, 1892

To represent orientation: Unit quaternions (3 DOF)

$$\mathring{q} = \cos\frac{\theta}{2} < \hat{v}\sin\frac{\theta}{2} >$$

$$\|q\| = s^2 + v_1^2 + v_2^2 + v_3^2 = 1$$

$$\mathring{q} \circ \mathring{q}' = \begin{pmatrix} s & -v_1 & -v_2 & -v_3 \\ v_1 & s & -v_3 & v_2 \\ v_2 & v_3 & s & -v_1 \\ v_3 & -v_2 & v_1 & s \end{pmatrix} \begin{pmatrix} s' \\ v_1' \\ v_2' \\ v_3' \end{pmatrix}$$

# Quaternions

### 2.2.2.2 Vector-Quaternion Pair

A compact and practical representation is the vector and unit quaternion pair. It represents pose using just 7 numbers, is easy to compound, and singularity free.◀

For the vector-quaternion case $\xi \sim (\boldsymbol{t}, \mathring{q})$ where $\boldsymbol{t} \in \mathbb{R}^3$ is a vector defining the frame's origin with respect to the reference coordinate frame, and $\mathring{q} \in \mathbb{S}^3$ is the frame's orientation with respect to the reference frame.

Composition is defined by

$$\xi_1 \oplus \xi_2 = (\boldsymbol{t}_1 + \mathring{q}_1 \bullet \boldsymbol{t}_2, \mathring{q}_1 \circ \mathring{q}_2)$$

and negation is

$$\ominus \xi = \left(-\mathring{q}^{-1} \bullet \boldsymbol{t}, \mathring{q}^{-1}\right)$$

and a point coordinate vector is transformed to a coordinate frame by

$${}^{X}\boldsymbol{p} = {}^{X}\xi_{Y} \bullet {}^{Y}\boldsymbol{p} = \mathring{q} \bullet {}^{Y}\boldsymbol{p} + \boldsymbol{t}$$

# The dark side

RVC 2.3.2 Lie algebra and the exponential map

RVC 2.3.3 Screw theory (screw, wrench, twist)

# Summary

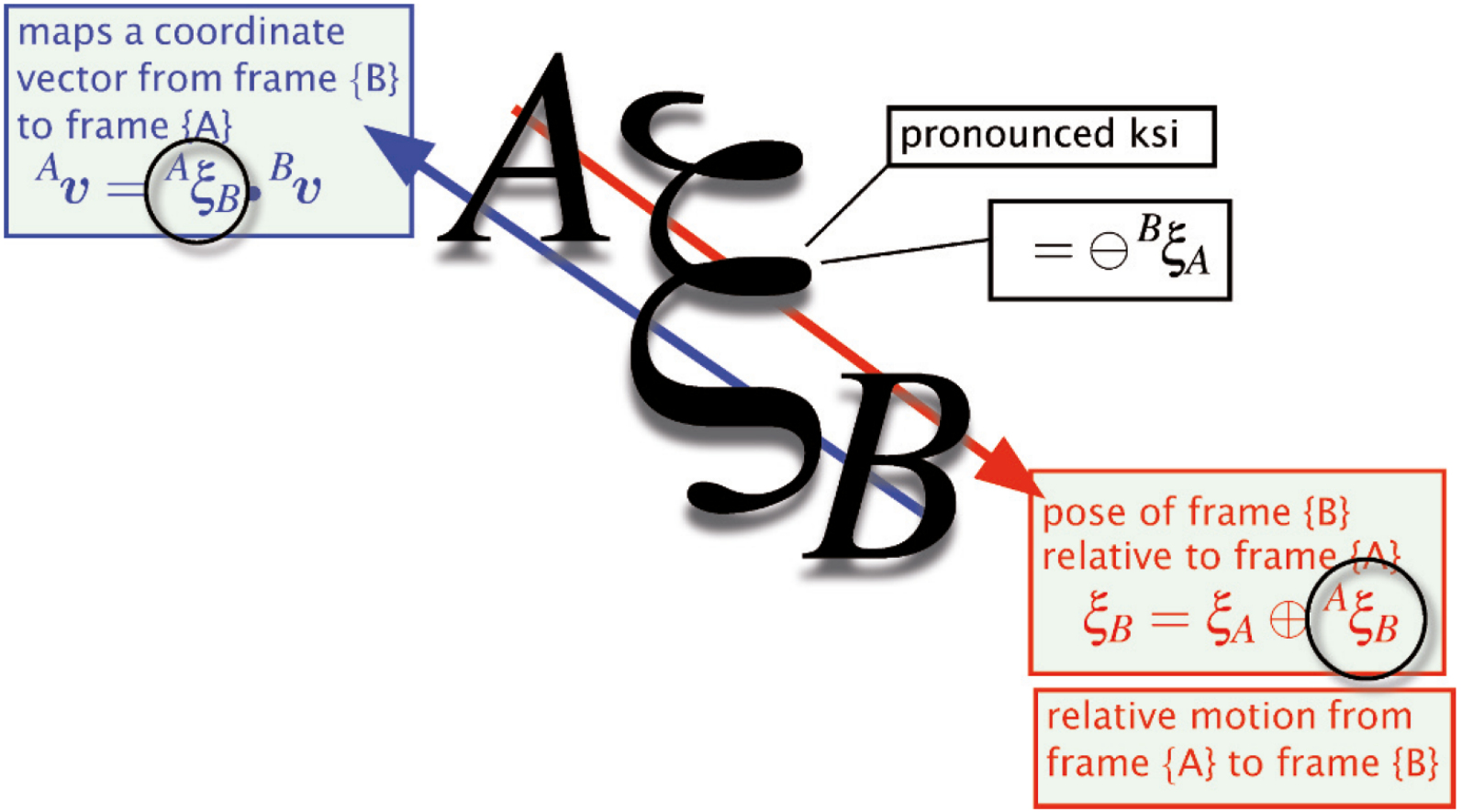



**Fig. 2.19.**
Everything you need to know about pose

# Summary

| | Output type | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Input type** | $t$ | $\theta$ | $R$ | $T$ | Twist vector | Twist | SO2 | SE2 |
| $t$ (2-vector) | | | | `transl2` | | `Twist('T')` | | `SE2()` |
| $\theta$ (scalar) | | | `rot2` | `trot2` | | `Twist('R')` | `SO2()` | `SE2()` |
| $R$ ($2 \times 2$ matrix) | | | | `r2t` | | | `SO2()` | `SE2()` |
| $T$ ($3 \times 3$ matrix) | `transl2` | | `t2r` | | | `Twist()` | | `SE2()` |
| Twist vector (1- or 3-vector) | | | `trexp2` | `trexp2` | | `Twist()` | `SO2.exp()` | `SE3.exp()` |
| **Twist** | | | | `.T` | `.S` | | | `.SE` |
| **SO2** | | `.theta` | `.R` | `.T` | `.log` | | | `.SE2` |
| **SE2** | `.t` | `.theta` | `.R` | `.T` | `.log` | `.Twist` | `.SO2` | |

**Table 2.1.** Toolbox supported data types for representing 2D pose: constructors and conversions

| | Output type | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Input type** | $t$ | Euler | RPY | $\theta, \boldsymbol{v}$ | $R$ | $T$ | Twist vector | Twist | Unit-Quaternion | SO3 | SE3 |
| $t$ (3-vector) | | | | | | `transl` | | `Twist('T')` | | | `SE3()` |
| Euler (3-vector) | | | | | `eul2r` | `eul2tr` | | | `UnitQuaternion.eul()` | `SO3.eul()` | `SE3.eul()` |
| RPY (3-vector) | | | | | `rpy2r` | `rpy2tr` | | | `UnitQuaternion.rpy()` | `SO3.rpy()` | `SE3.rpy()` |
| $\theta, \boldsymbol{v}$ (scalar + 3-vector) | | | | | `angvec2r` | `angvec2tr` | | | `UnitQuaternion.angvec()` | `SO3.angvec()` | `SE3.angvec()` |
| $R$ ($3 \times 3$ matrix) | | `tr2eul` | `tr2rpy` | `tr2angvec` | | `r2t` | `trlog` | | `UnitQuaternion()` | `SO3()` | `SE3()` |
| $T$ ($4 \times 4$ matrix) | `transl` | `tr2eul` | `tr2rpy` | `tr2angvec` | `t2r` | | `trlog` | `Twist()` | `UnitQuaternion()` | `SO3()` | `SE3()` |
| Twist vector (3- or 6-vector) | | | | | `trexp` | `trexp` | | `Twist()` | | `SO3.exp()` | `SE3.exp()` |
| Twist | | | | | | `.T` | `.S` | | | | `.SE` |
| Unit-Quaternion | | `.toeul` | `.torpy` | `.toangvec` | `.R` | `.T` | | | | `.SO3` | `.SE3` |
| SO3 | | `.toeul` | `.torpy` | `.toangvec` | `.R` | `.T` | `.log` | | `.UnitQuaternion` | | `.SE3` |
| SE3 | `.t` | `.toeul` | `.torpy` | `.toangvec` | `.R` | `.T` | `.log` | `.Twist` | `.UnitQuaternion` | `.SO3` | |

**Table 2.2.** Toolbox supported data types for representing 3D pose: constructors and conversions

# Outline

- ✔ SO(2): 2-D rotation / orientation
- ✔ SE(2): 2-D transformations
- ✔ SO(3): 3-D rotation / orientation
- ✔ SE(3): 3-D transformations

- ✔ More representations for 3-D rotations
  - Euler angles
  - Axis-angle representations
  - Quaternions

Forward kinematics (FK)

Inverse kinematics (IK)

Jacobians + more IK (time permitting)

# Forward kinematics

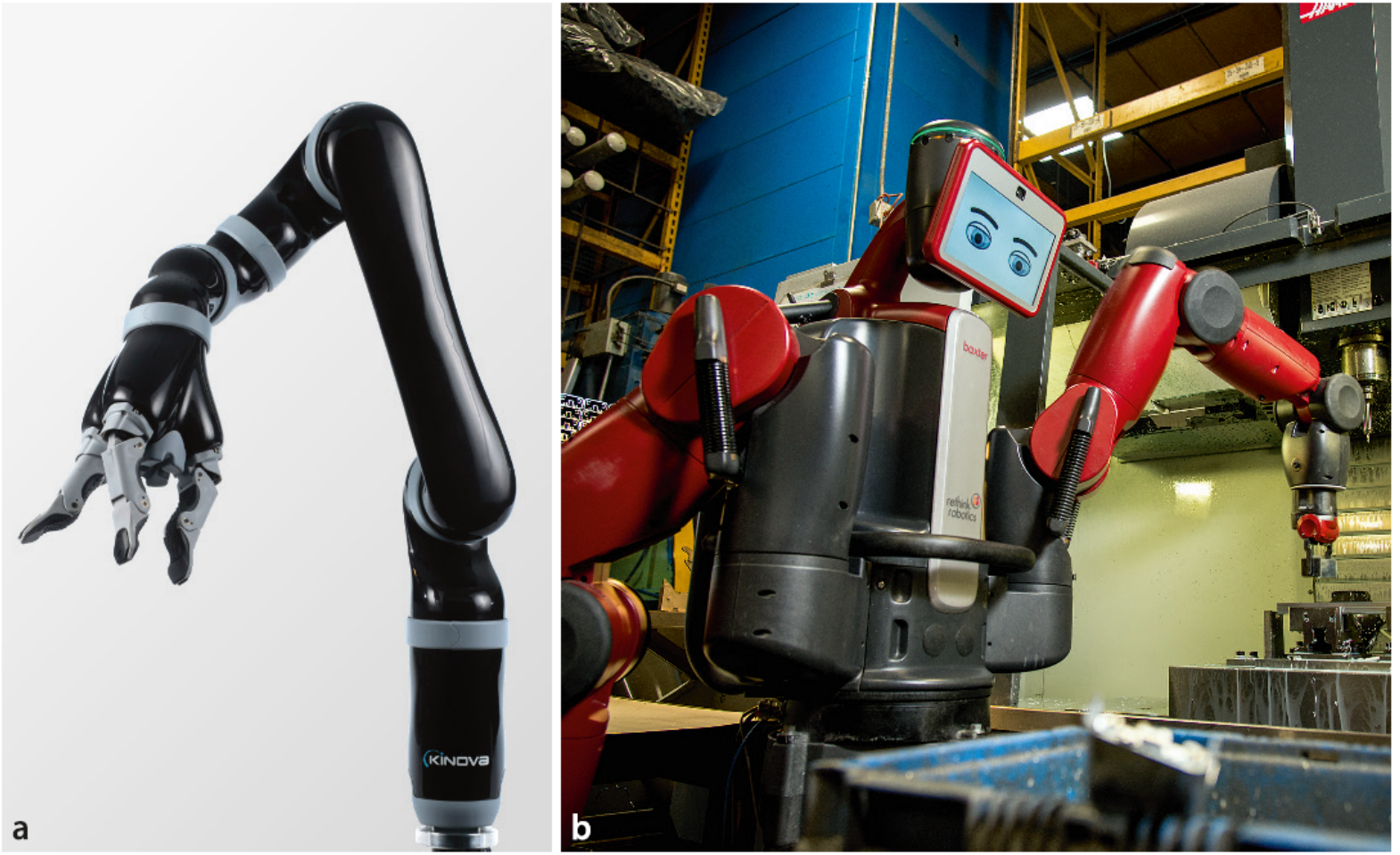
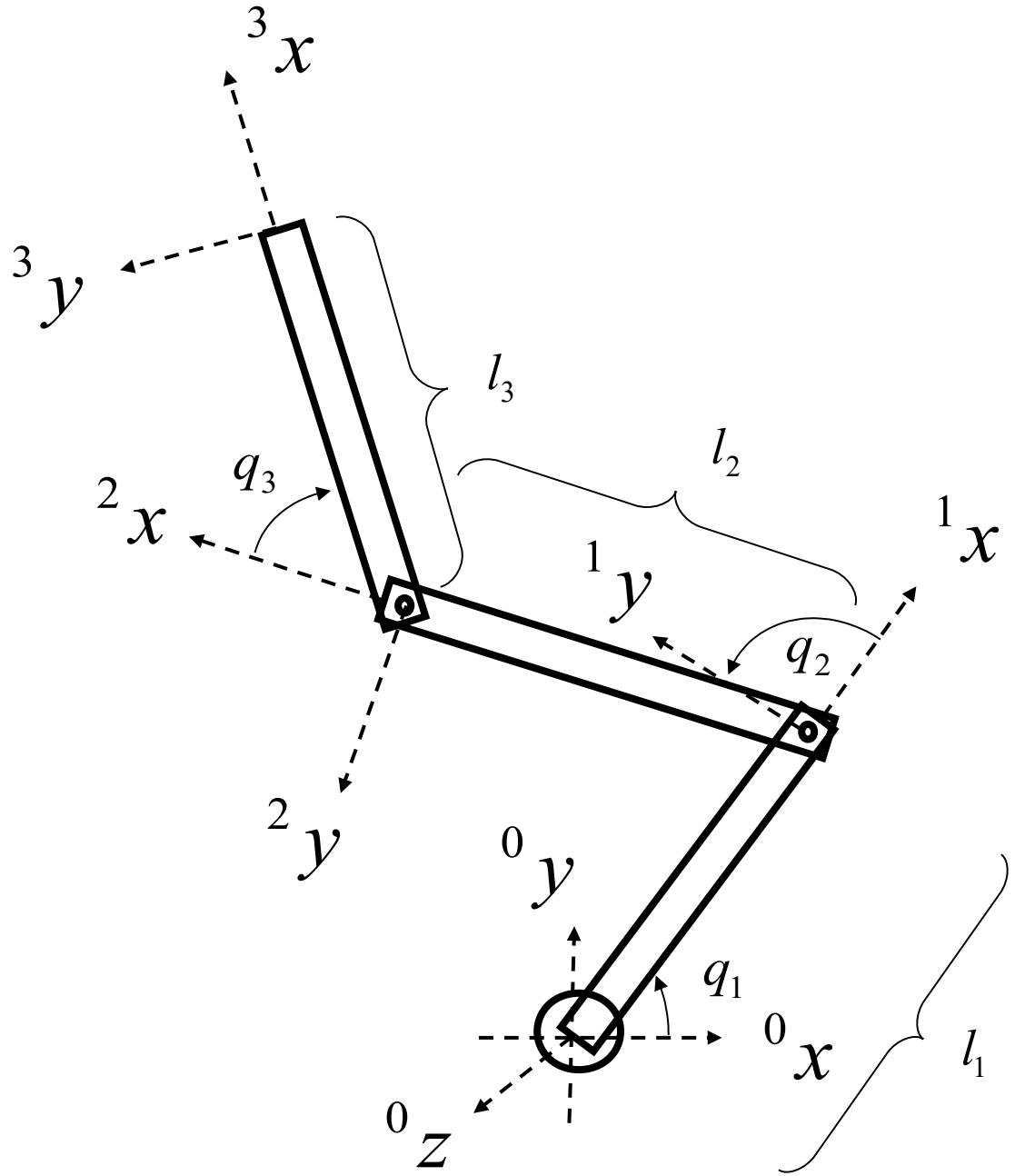



**Fig. 7.1.**
**a** Mico 6-joint robot with 3-fingered hand (courtesy of Kinova Robotics). **b** Baxter 2-armed robotic coworker, each arm has 7 joints (courtesy of Rethink Robotics)

Where is the end-effector with respect to the base frame?

# FK: Composition of homogeneous transformations



# FK: Composition of homogeneous transformations

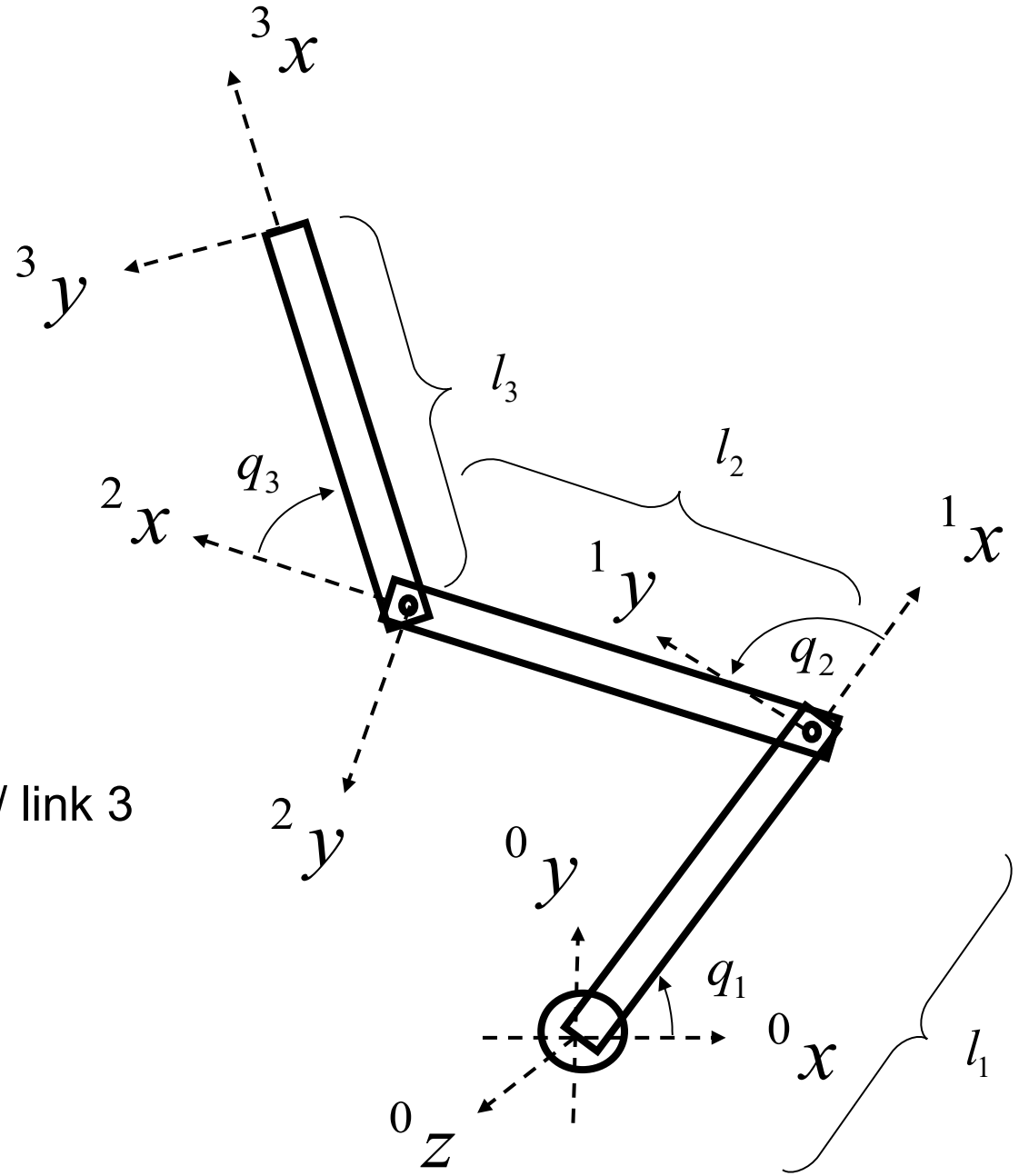
Base to end-effector transform

$${}^0T_3 = {}^0T_1\,{}^1T_2\,{}^2T_3$$

Transform associated w/ link 3

Transform associated w/ link 2

Transform associated w/ link 1

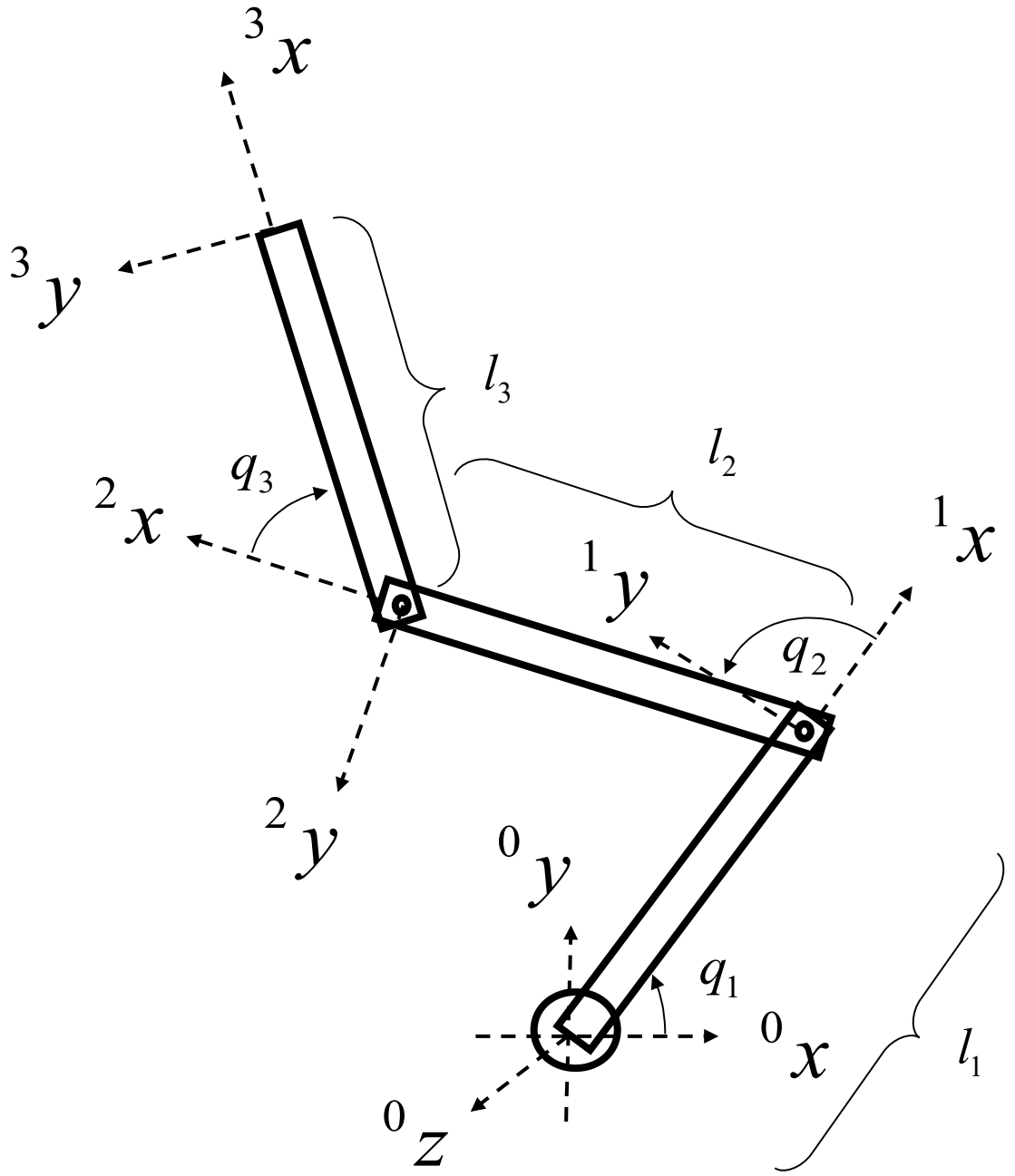


# FK: Composition of homogeneous transformations

$$ {}^0T_3 = {}^0T_1\,{}^1T_2\,{}^2T_3 $$

$$ {}^0T_1 = \begin{pmatrix} c_1 & -s_1 & 0 & l_1 c_1 \\ s_1 & c_1 & 0 & l_1 s_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

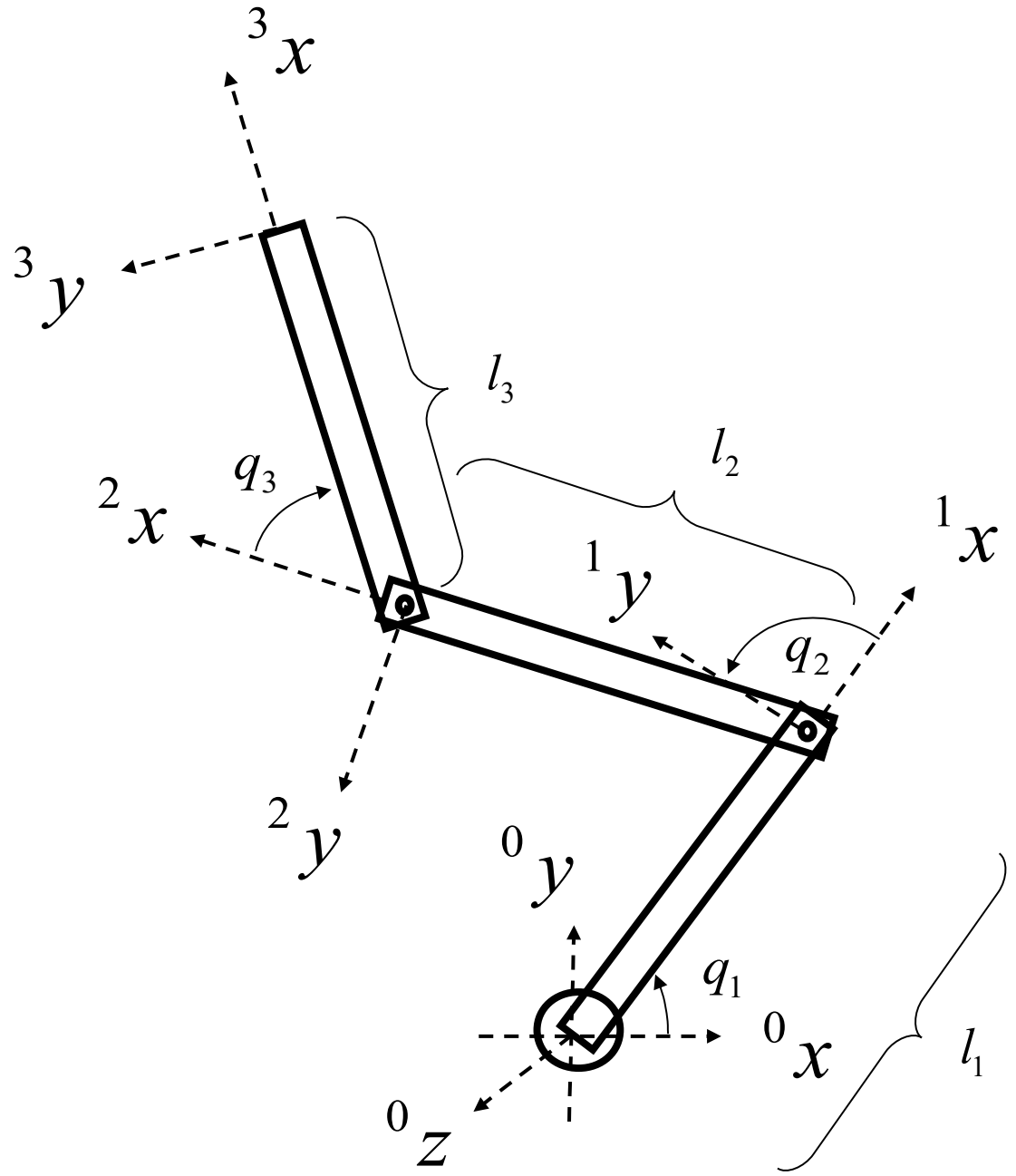
$$ {}^1T_2 = \begin{pmatrix} c_2 & -s_2 & 0 & l_2 c_2 \\ s_2 & c_2 & 0 & l_2 s_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$



# FK: Composition of homogeneous transformations

$$ {}^{0}T_{3} = {}^{0}T_{1}\,{}^{1}T_{2}\,{}^{2}T_{3} $$

$$ {}^{2}T_{3} = \begin{pmatrix} c_3 & -s_3 & 0 & l_3 c_3 \\ s_3 & c_3 & 0 & l_3 s_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$



# FK: Composition of homogeneous transformations

$$ {}^{0}T_{3} = {}^{0}T_{1}\,{}^{1}T_{2}\,{}^{2}T_{3} $$

$$ {}^{0}T_{3} = \begin{pmatrix} c_1 & -s_1 & 0 & l_1 c_1 \\ s_1 & c_1 & 0 & l_1 s_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} c_2 & -s_2 & 0 & l_2 c_2 \\ s_2 & c_2 & 0 & l_2 s_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} c_3 & -s_3 & 0 & l_3 c_3 \\ s_3 & c_3 & 0 & l_3 s_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

# FK: Composition of homogeneous transformations

$$ {}^{0}T_{3} = {}^{0}T_{1}\,{}^{1}T_{2}\,{}^{2}T_{3} $$

$$ {}^{0}T_{3} = \begin{pmatrix} c_1 & -s_1 & 0 & l_1 c_1 \\ s_1 & c_1 & 0 & l_1 s_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} c_2 & -s_2 & 0 & l_2 c_2 \\ s_2 & c_2 & 0 & l_2 s_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} c_3 & -s_3 & 0 & l_3 c_3 \\ s_3 & c_3 & 0 & l_3 s_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

$$ {}^{0}T_{3} = \begin{pmatrix} c_{123} & -s_{123} & 0 & l_1 c_1 + l_2 c_{12} + l_3 c_{123} \\ s_{123} & c_{123} & 0 & l_1 s_1 + l_2 s_{12} + l_3 s_{123} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

# FK: Composition of homogeneous transformations

(Remember double-angle formulas?)

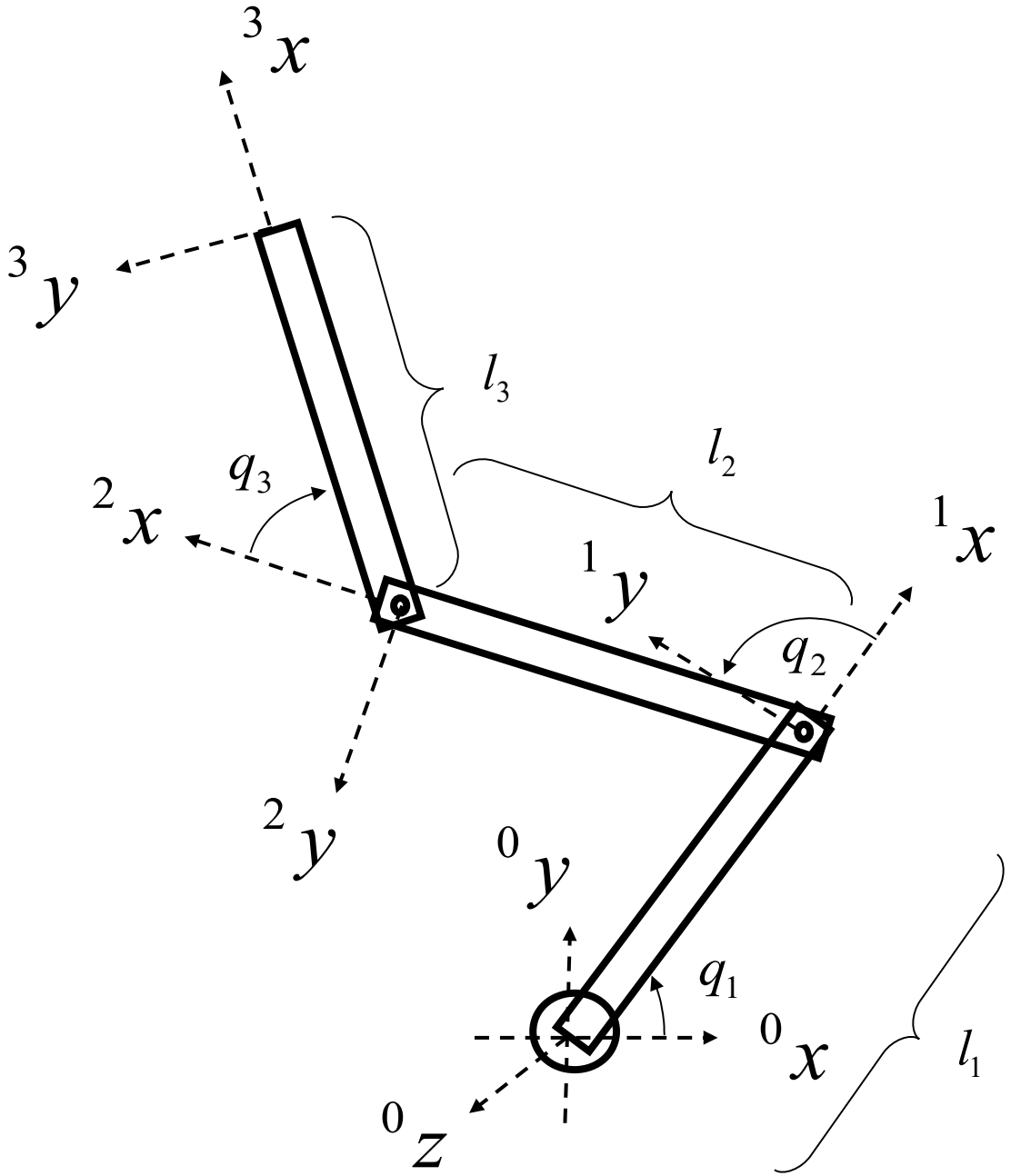
$$\sin(\theta \pm \phi) = \sin(\theta)\cos(\phi) \pm \cos(\theta)\sin(\phi)$$

$$\cos(\theta \pm \phi) = \cos(\theta)\cos(\phi) \mp \sin(\theta)\sin(\phi)$$

# FK: Composition of homogeneous transformations

Where is the end-effector with respect to the base?

$$
{}^{0}T_{3} = \begin{pmatrix} c_{123} & -s_{123} & 0 & l_1 c_1 + l_2 c_{12} + l_3 c_{123} \\ s_{123} & c_{123} & 0 & l_1 s_1 + l_2 s_{12} + l_3 s_{123} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}
$$



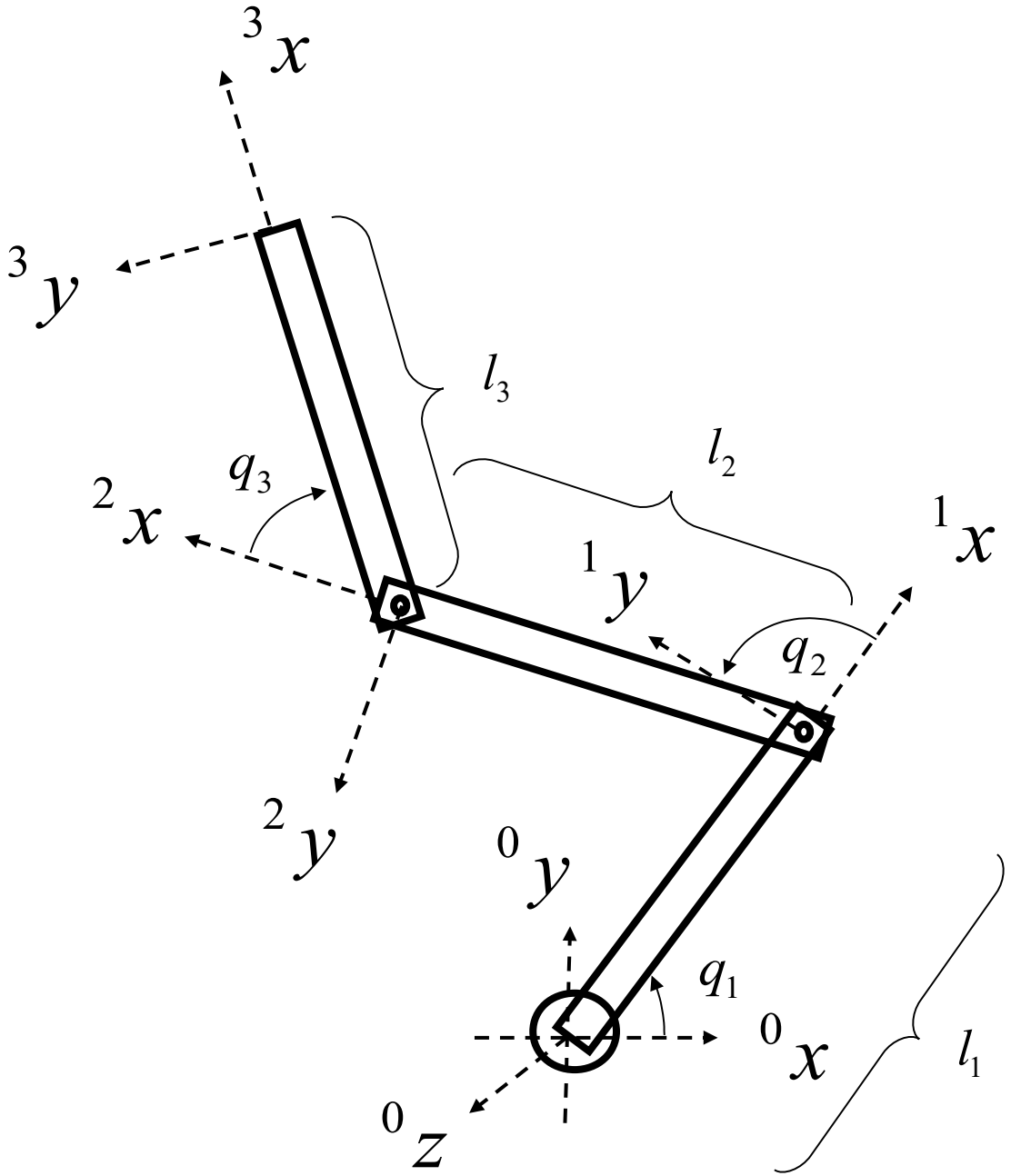
# FK: Composition of homogeneous transformations

Where is the end-effector with respect to the base?

$$
{}^{0}T_{3} = \begin{pmatrix} c_{123} & -s_{123} & 0 & l_1 c_1 + l_2 c_{12} + l_3 c_{123} \\ s_{123} & c_{123} & 0 & l_1 s_1 + l_2 s_{12} + l_3 s_{123} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}
$$

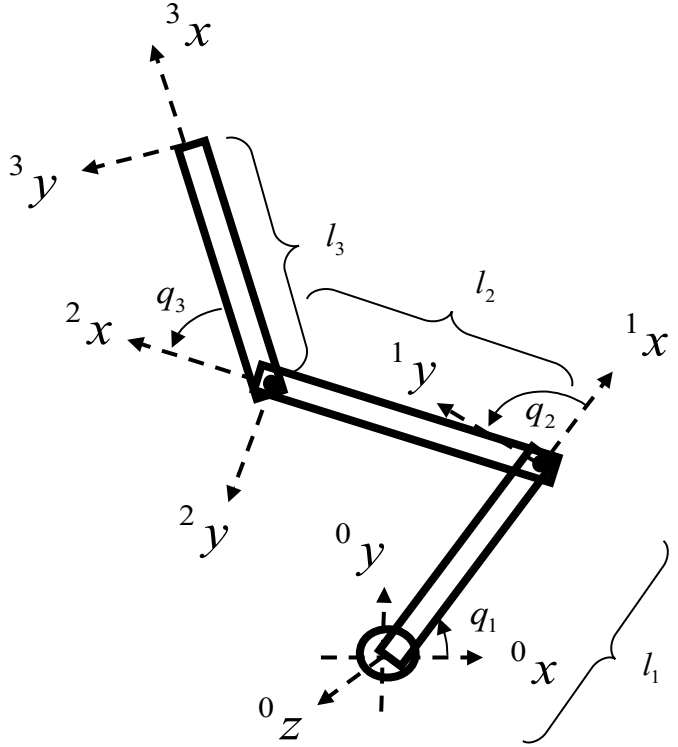
More abstractly:

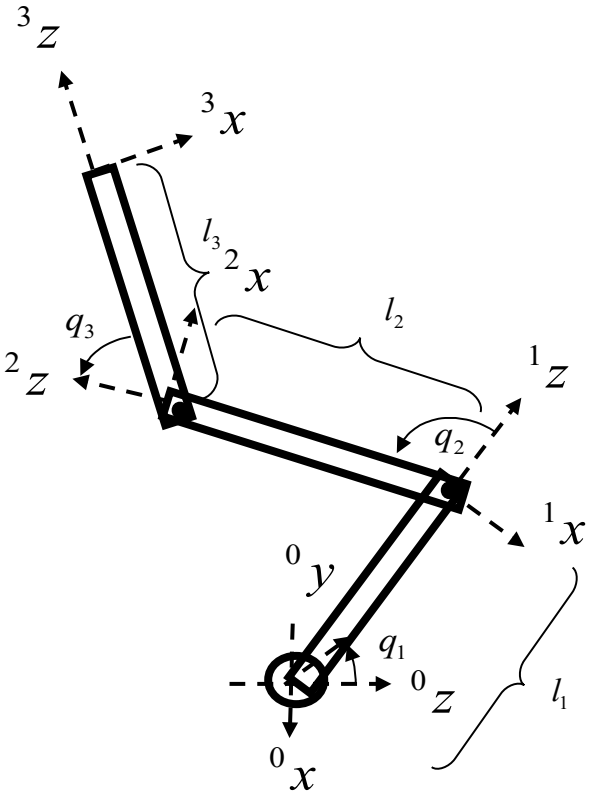
EE pose = FK(q)



# DH parameters

- There are a large number of ways that homogeneous transformations can encode the kinematics of a manipulator
- We will sacrifice some of this flexibility for a more systematic approach: DH (Denavit-Hartenberg) parameters
- DH parameters is a standard for describing a series of transforms for arbitrary mechanisms



# FK with DH parameters

**Table 7.1.** Denavit-Hartenberg parameters: their physical meaning, symbol and formal definition

| | | | |
|---|---|---|---|
| Joint angle | $\theta_j$ | the angle between the $x_{j-1}$ and $x_j$ axes about the $z_{j-1}$ axis | revolute joint variable |
| Link offset | $d_j$ | the distance from the origin of frame $j-1$ to the $x_j$ axis along the $z_{j-1}$ axis | prismatic joint variable |
| Link length | $a_j$ | the distance between the $z_{j-1}$ and $z_j$ axes along the $x_j$ axis; for intersecting axes is parallel to $\hat{z}_{j-1} \times \hat{z}_j$ | constant |
| Link twist | $\alpha_j$ | the angle from the $z_{j-1}$ axis to the $z_j$ axis about the $x_j$ axis | constant |
| Joint type | $\sigma_j$ | $\sigma = R$ for a revolute joint, $\sigma = P$ for a prismatic joint | constant |

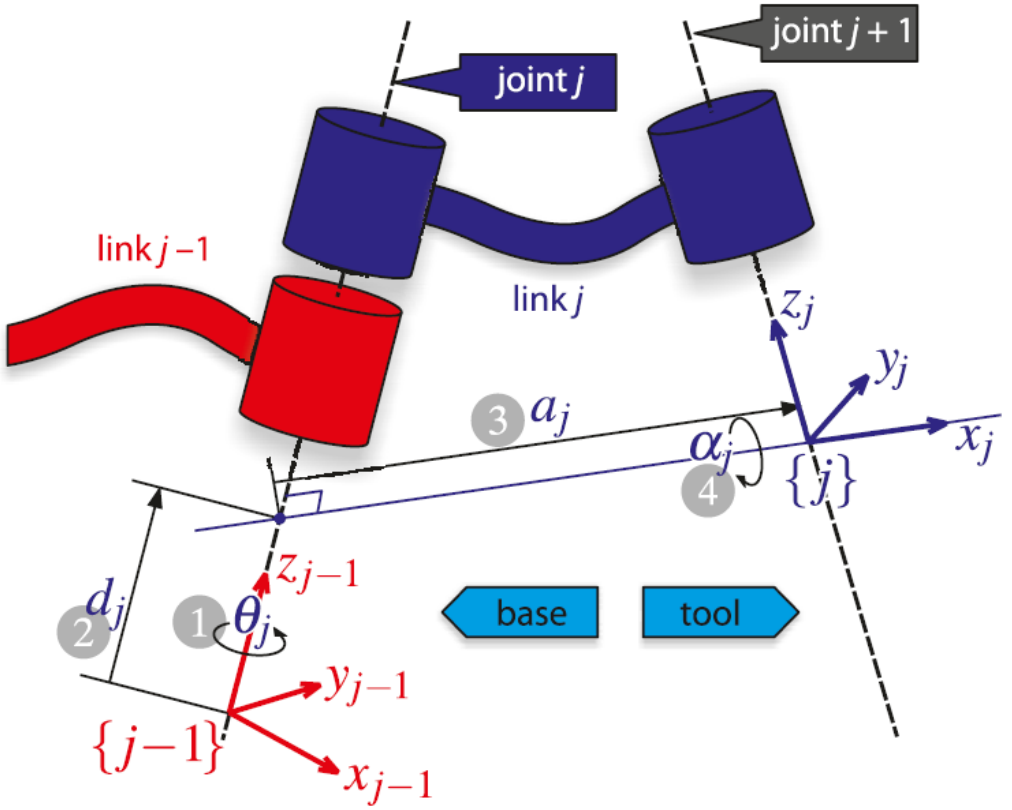



**Fig. 7.5.** Definition of standard Denavit and Hartenberg link parameters. The colors red and blue denote all things associated with links $j-1$ and $j$ respectively. The numbers in circles represent the order in which the elementary transforms are applied. $x_j$ is parallel to $z_{j-1} \times z_j$ and if those two axes are parallel then $d_j$ can be arbitrarily chosen

# FK with DH parameters

The transformation from link coordinate frame $\{j-1\}$ to frame $\{j\}$ is defined in terms of elementary rotations and translations as

$$^{j-1}\xi_j\left(\theta_j, d_j, a_j, \alpha_j\right) = \mathscr{R}_z\left(\theta_j\right) \oplus \mathscr{T}_z\left(d_j\right) \oplus \mathscr{T}_x\left(a_j\right) \oplus \mathscr{R}_x\left(\alpha_j\right) \tag{7.2}$$

# FK with DH parameters

The transformation from link coordinate frame $\{j-1\}$ to frame $\{j\}$ is defined in terms of elementary rotations and translations as

$$ {}^{j-1}\xi_j(\theta_j, d_j, a_j, \alpha_j) = \mathscr{R}_z(\theta_j) \oplus \mathscr{T}_z(d_j) \oplus \mathscr{T}_x(a_j) \oplus \mathscr{R}_x(\alpha_j) \tag{7.2} $$

$$ T = \underbrace{\begin{pmatrix} c_{\theta_i} & -s_{\theta_i} & 0 & 0 \\ s_{\theta_i} & c_{\theta_i} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{\text{First, translate by } d_i \text{ along } z \text{ axis and rotate by } \theta_i \text{ about } z \text{ axis}} \underbrace{\begin{pmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & c_{\alpha_i} & -s_{\alpha_i} & 0 \\ 0 & s_{\alpha_i} & c_{\alpha_i} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{\text{Then, translate by } a_i \text{ along } x \text{ axis and rotate by } \alpha_i \text{ about } x \text{ axis}} $$

# FK with DH parameters

These four DH parameters,

$$\begin{pmatrix} a_i & \alpha_i & d_i & \theta_i \end{pmatrix}$$

represent the following homogeneous matrix:

$$T = \underbrace{\begin{pmatrix} c_{\theta_i} & -s_{\theta_i} & 0 & 0 \\ s_{\theta_i} & c_{\theta_i} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{\text{First, translate by } d_i \text{ along } z \text{ axis and rotate by } \theta_i \text{ about } z \text{ axis}} \underbrace{\begin{pmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & c_{\alpha_i} & -s_{\alpha_i} & 0 \\ 0 & s_{\alpha_i} & c_{\alpha_i} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{\text{Then, translate by } a_i \text{ along } x \text{ axis and rotate by } \alpha_i \text{ about } x \text{ axis}}$$

# FK with DH parameters

$$T = \begin{pmatrix} c_{\theta_i} & -s_{\theta_i} & 0 & 0 \\ s_{\theta_i} & c_{\theta_i} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & c_{\alpha_i} & -s_{\alpha_i} & 0 \\ 0 & s_{\alpha_i} & c_{\alpha_i} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} c_{\theta_i} & -s_{\theta_i} c_{\alpha_i} & s_{\theta_i} s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i} c_{\alpha_i} & -c_{\theta_i} s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

# FK with DH parameters

The transformation from link coordinate frame $\{j-1\}$ to frame $\{j\}$ is defined in terms of elementary rotations and translations as

$$^{j-1}\xi_j\left(\theta_j, d_j, a_j, \alpha_j\right) = \mathscr{R}_z\left(\theta_j\right) \oplus \mathscr{T}_z\left(d_j\right) \oplus \mathscr{T}_x\left(a_j\right) \oplus \mathscr{R}_x\left(\alpha_j\right) \tag{7.2}$$

which can be expanded in homogeneous matrix form as

$$^{j-1}\boldsymbol{A}_j = \begin{pmatrix} \cos\theta_j & -\sin\theta_j\cos\alpha_j & \sin\theta_j\sin\alpha_j & a_j\cos\theta_j \\ \sin\theta_j & \cos\theta_j\cos\alpha_j & -\cos\theta_j\sin\alpha_j & \alpha_j\sin\theta_j \\ 0 & \sin\alpha_j & \cos\alpha_j & d_j \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{7.3}$$

# FK with DH parameters

The transformation from link coordinate frame $\{j-1\}$ to frame $\{j\}$ is defined in terms of elementary rotations and translations as

$$^{j-1}\xi_j\left(\theta_j, d_j, a_j, \alpha_j\right) = \mathscr{R}_z\left(\theta_j\right) \oplus \mathscr{T}_z\left(d_j\right) \oplus \mathscr{T}_x\left(a_j\right) \oplus \mathscr{R}_x\left(\alpha_j\right) \tag{7.2}$$

which can be expanded in homogeneous matrix form as

$$^{j-1}\boldsymbol{A}_j = \begin{pmatrix} \cos\theta_j & -\sin\theta_j\cos\alpha_j & \sin\theta_j\sin\alpha_j & a_j\cos\theta_j \\ \sin\theta_j & \cos\theta_j\cos\alpha_j & -\cos\theta_j\sin\alpha_j & \alpha_j\sin\theta_j \\ 0 & \sin\alpha_j & \cos\alpha_j & d_j \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{7.3}$$

For <u>revolute</u> joints, $\theta_j$ varies

For <u>prismatic</u> joints, $d_j$ varies

# FK with DH parameters: Example

### 7.1.2.3 6-Axis Industrial Robot

Truly useful robots have a task space $\mathcal{T} \subset \mathbf{SE}(3)$ enabling arbitrary position and attitude of the end-effector – the task space has six spatial degrees of freedom: three translational and three rotational. This requires a robot with a configuration space $\mathcal{C} \subset \mathbb{R}^6$ which can be achieved by a robot with six joints. In this section we will use the Puma 560 as an example of the class of all-revolute six-axis robot manipulators. We define an instance of a Puma 560 robot using the script

```
>> mdl_puma560
```

which creates a `SerialLink` object, `p560`, in the workspace. Displaying the variable shows the table of its Denavit-Hartenberg parameters

```
>> p560
Puma 560 [Unimation]:: 6 axis, RRRRRR, stdDH, slowRNE
 - viscous friction; params of 8/95;
+---+----------+----------+----------+----------+----------+
| j |     theta |         d |         a |     alpha |    offset |
+---+----------+----------+----------+----------+----------+
|  1|         q1|         0|         0|     1.571|         0|
|  2|         q2|         0|    0.4318|         0|         0|
|  3|         q3|      0.15|    0.0203|    -1.571|         0|
|  4|         q4|    0.4318|         0|     1.571|         0|
|  5|         q5|         0|         0|    -1.571|         0|
|  6|         q6|         0|         0|         0|         0|
+---+----------+----------+----------+----------+----------+
```

# Outline

- ✔ SO(2): 2-D rotation / orientation
- ✔ SE(2): 2-D transformations
- ✔ SO(3): 3-D rotation / orientation
- ✔ SE(3): 3-D transformations

- ✔ More representations for 3-D rotations
  - Euler angles
  - Axis-angle representations
  - Quaternions

- ✔ Forward kinematics (FK)

Inverse kinematics (IK)

Jacobians + more IK (time permitting)

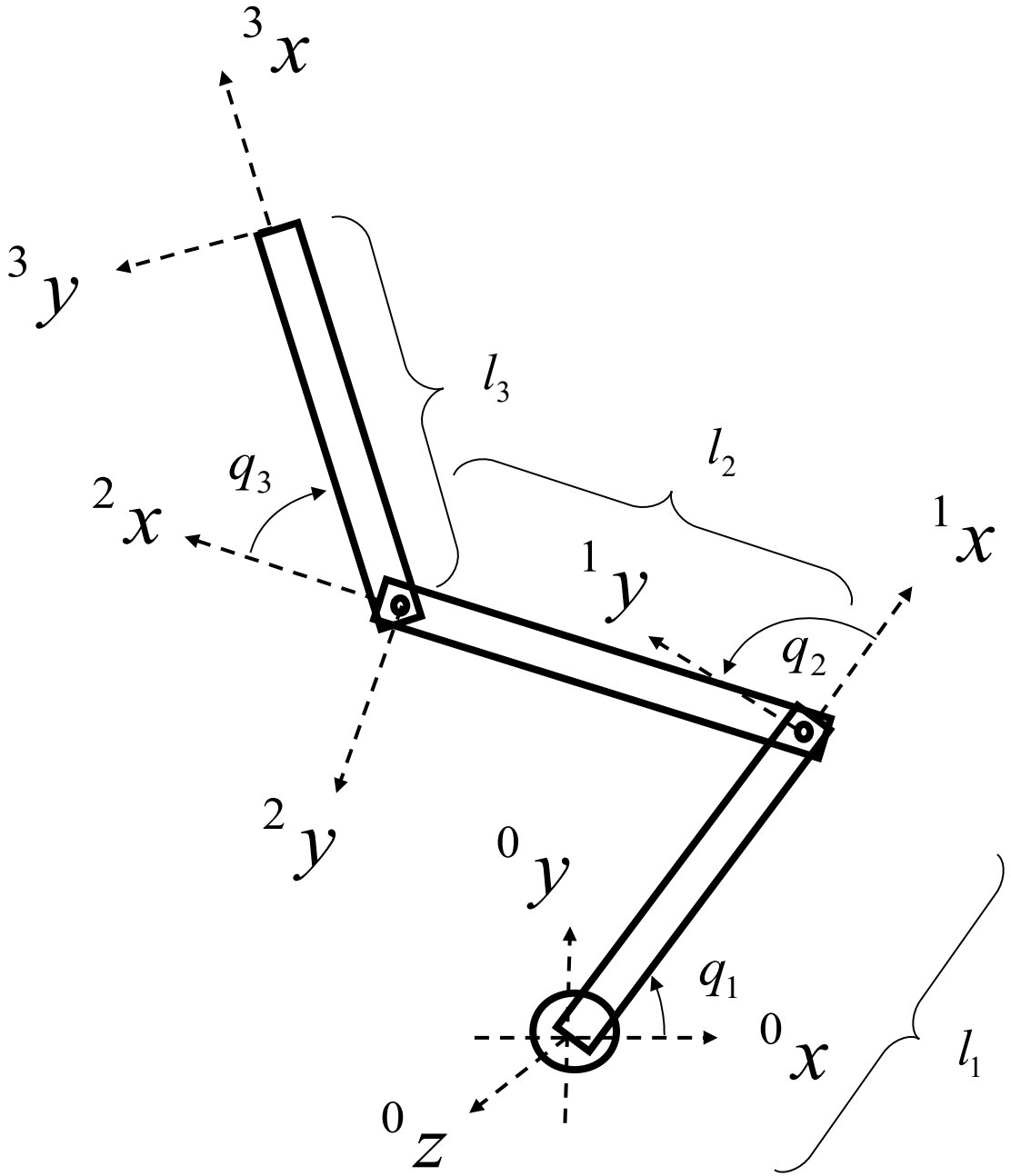
# Forward kinematics vs. inverse kinematics

Where is the end-effector with respect to the base?

$$
{}^{0}T_{3} = \begin{pmatrix} c_{123} & -s_{123} & 0 & l_1 c_1 + l_2 c_{12} + l_3 c_{123} \\ s_{123} & c_{123} & 0 & l_1 s_1 + l_2 s_{12} + l_3 s_{123} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}
$$

More abstractly:

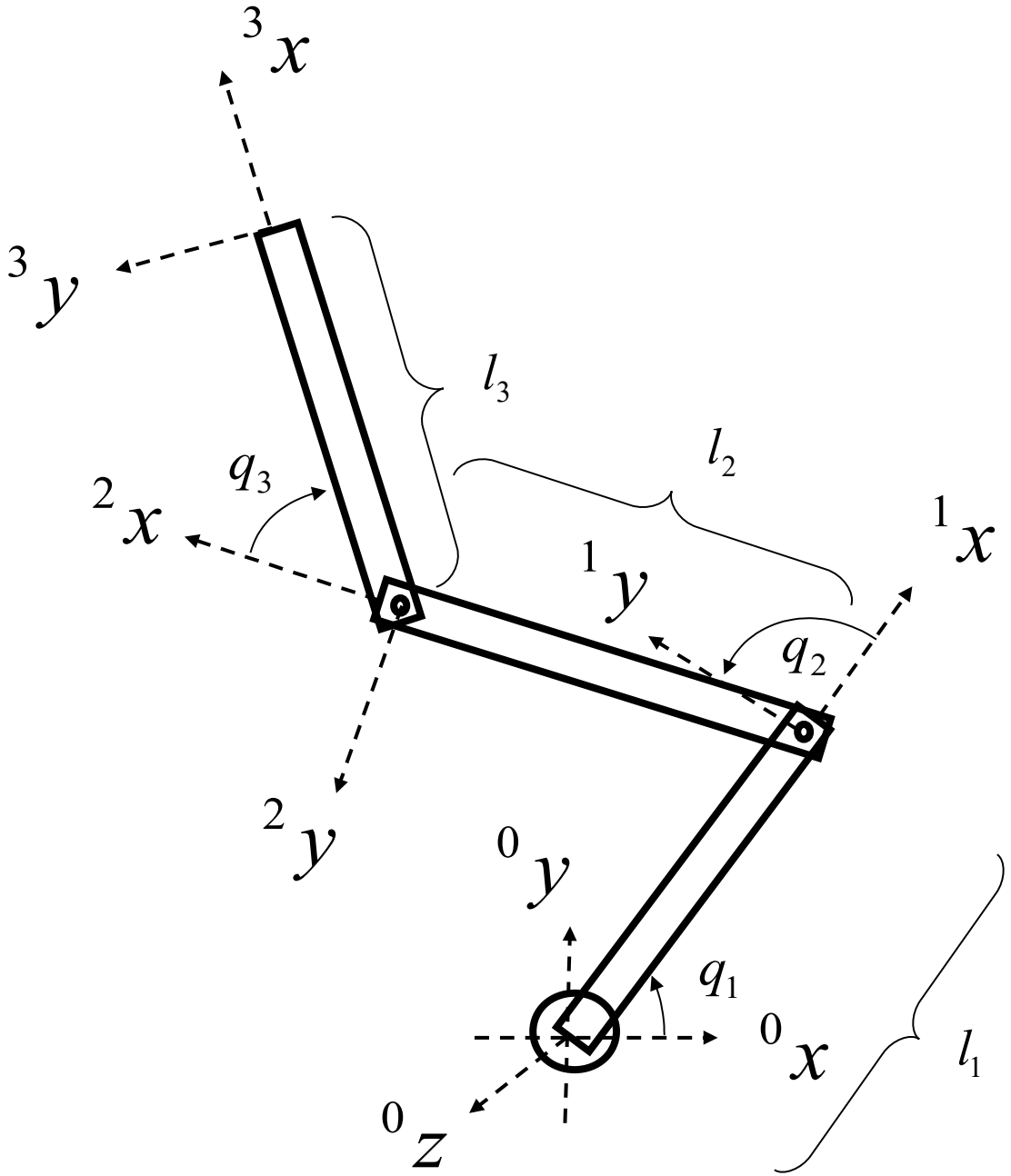
EE pose = FK(q)



# Forward kinematics vs. inverse kinematics

Where is the end-effector with respect to the base?

$$
{}^{0}T_{3} = \begin{pmatrix} c_{123} & -s_{123} & 0 & l_1 c_1 + l_2 c_{12} + l_3 c_{123} \\ s_{123} & c_{123} & 0 & l_1 s_1 + l_2 s_{12} + l_3 s_{123} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}
$$




More abstractly:

EE pose = FK(q)

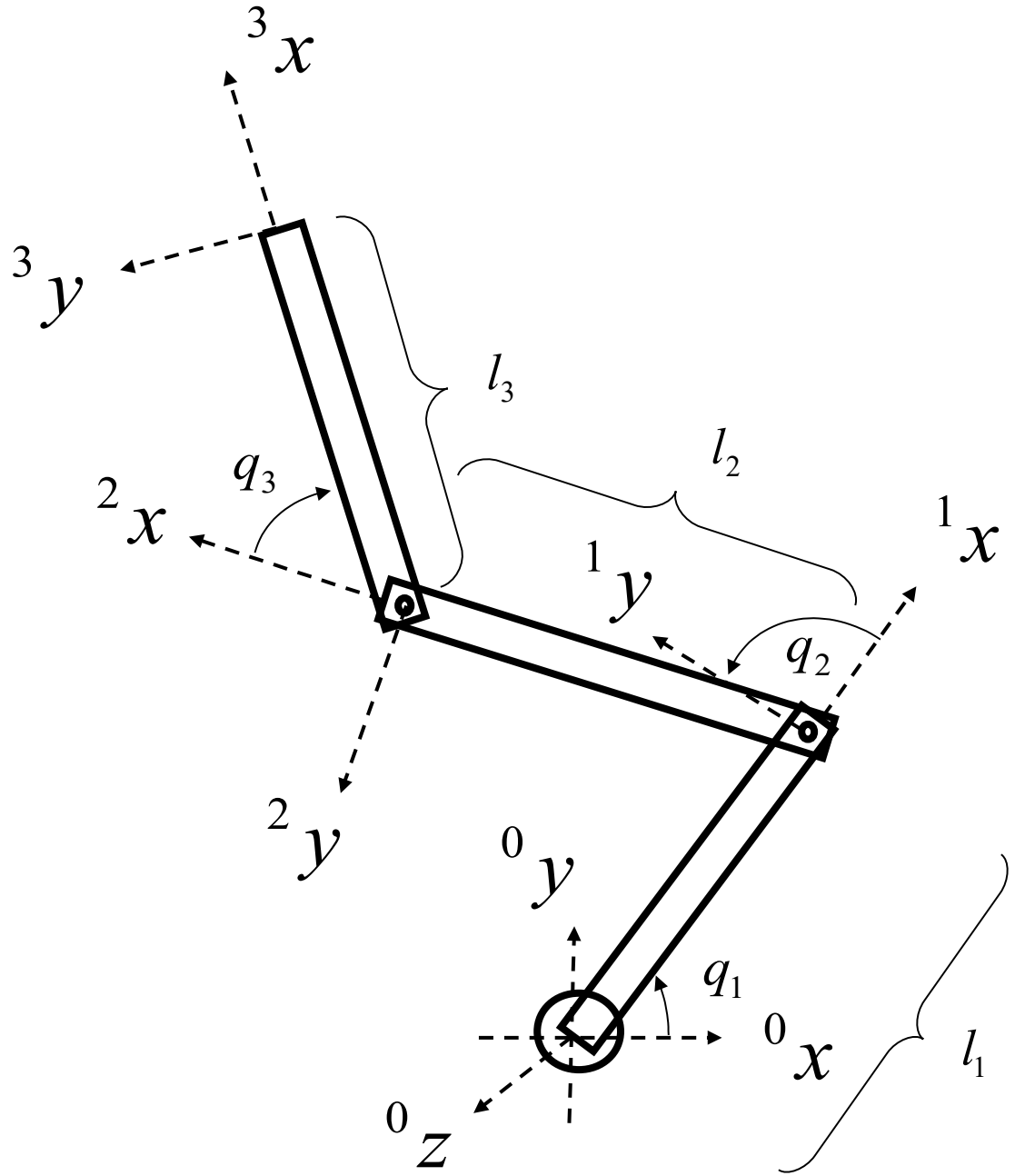
Inverse kinematics:

q = IK(EE pose)

# Forward kinematics vs. inverse kinematics

Where is the end-effector with respect to the base?

$$
{}^{0}T_{3} = \begin{pmatrix} c_{123} & -s_{123} & 0 & l_1 c_1 + l_2 c_{12} + l_3 c_{123} \\ s_{123} & c_{123} & 0 & l_1 s_1 + l_2 s_{12} + l_3 s_{123} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}
$$




More abstractly:

EE pose = FK(q)

Inverse kinematics:

$$\boldsymbol{q} = \mathcal{K}^{-1}(\xi)$$

# Inverse kinematics

$$\boldsymbol{q} = \mathcal{K}^{-1}(\xi)$$

Unlike FK, inverse problems are typically harder

There can be no solutions, 1 solution,
or multiple solutions (possibly infinite)

# Inverse kinematics

$$\boldsymbol{q} = \mathcal{K}^{-1}(\xi)$$

Unlike FK, inverse problems are typically harder

There can be no solutions, 1 solution,
or multiple solutions (possibly infinite)

Two solution approaches:

- Closed-form (analytical) solution

- Numerical (iterative) solution

# Inverse kinematics

$$\boldsymbol{q} = \mathcal{K}^{-1}(\xi)$$

Unlike FK, inverse problems are typically harder

There can be no solutions, 1 solution,
or multiple solutions (possibly infinite)

Two solution approaches:

- Closed-form (analytical) solution
  - Solve FK equations for the joint angles exactly
    (see Ex1 Q5) – can be hard to derive!
  - Typically faster / can be computed offline
- Numerical (iterative) solution
  - Iterate toward desired goal position
  - Easier to think about / universally applicable

# Feedback

## Piazza thread: 1/31 Lec 04 Feedback

Please post your answers to the following anonymously.

1. What did you like so far?

2. What was unclear?

3. What can we provide to guide
   your initial thinking about the project?

4. Any additional feedback / comments?