r / Vector arithmetic and linear algebra:

- matrix/vector dimensions
- vector addition
- matrix addition
- vector norm and direction
- matrix-vector multiplication
- identity matrix
- zero matrix, zero vector
- matrix transpose
- matrix inverse

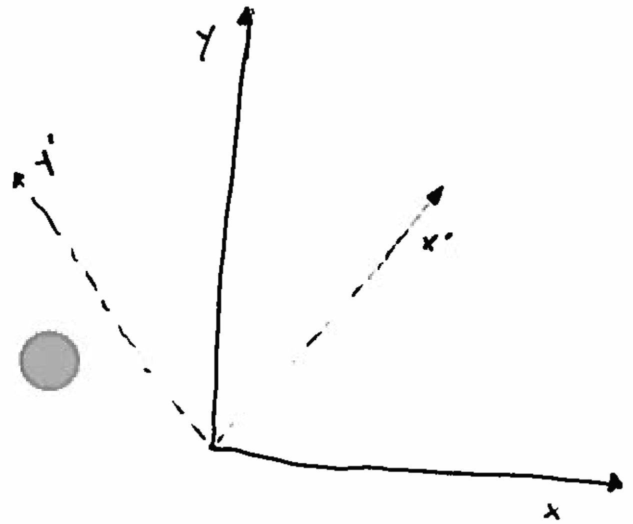
- Transforming points & vectors between coordinate systems
  - May want to know position & orientation of object w.r.t. robot's wrist, finger, foot, etc. [relative transformation]
  - May want to know object/robot's "global" reference frame (GPS coordinates) [absolute transformation]
  - Necessary for graphical rendering
- Projecting points in a 3D world to a 2D display (rendering / computer vision)

2D rotation

We can rotate a point in 2D using the following matrix:

$$R_z = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

Then, $p' = R_z p$ (this is of course a linear operation)

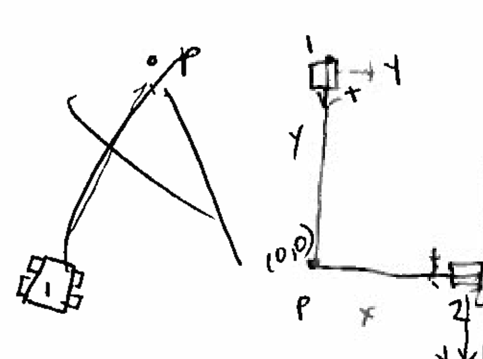


$$R = \begin{bmatrix} \sqrt{2}/2 & -\sqrt{2}/2 \\ \sqrt{2}/2 & \sqrt{2}/2 \end{bmatrix}$$

$$\begin{array}{cc} \uparrow & \uparrow \\ x' & y' \end{array}$$

Note that R transforms a point in its <u>frame</u> (we'll discuss this term in detail later) to the global frame. For example, point $\begin{bmatrix} 2 \\ 0 \end{bmatrix}$ in R's frame is equivalent to $\begin{bmatrix} \sqrt{2} \\ \sqrt{2} \end{bmatrix}$ in the global frame (draw this). R^{-1} transforms a point in the global frame to R's frame:

$$R^{-1} = \begin{bmatrix} \sqrt{2}/2 & \sqrt{2}/2 \\ -\sqrt{2}/2 & \sqrt{2}/2 \end{bmatrix} \begin{bmatrix} \sqrt{2}/2 \\ \sqrt{2}/2 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$



* Show frame convention!

Inverse rotation

- Rotation by $-\theta$ is equal to transpose of rotation matrix by θ:

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

(from trig. identities)

$$\cos -\theta = \cos\theta$$

$$\sin -\theta = -\sin\theta$$

- Notice that rotation matrix is orthogonal:

$$\cos\theta \sin\theta - \cos\theta \sin\theta = 0$$

(this true of both $R_z(\theta)$ and $R_z(-\theta)$)

- Rotation matrix also ~~a normal matrix~~ has all unit-vector columns (and rows!):

$$\cos^2\theta + \sin^2\theta = 1$$

Both properties combined make R_z (as well as R_z^T) orthonormal

Orthonormal matrix has property $R^{-1} = R^T$

- always (easily!) invertible
- $\det(R) = \pm 1$
- $Rp = q \Rightarrow R^T q = p$

Rotations are length preserving!

A rotation is a linear transformation. Types of linear transformation operations:

- rotation
- scaling $\begin{bmatrix} 2 & 0 \\ 0 & 1/2 \end{bmatrix}$ scaling
- shearing $\begin{bmatrix} 1 & m \\ 0 & 1 \end{bmatrix}$ shearing
- reflection $\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$ reflection
- ~~shee~~ projection $\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$ projection

We cannot do translation using a linear operation:

$$P = P_1 + P_2$$

Instead, we need an affine transformation - linear map followed by translation:

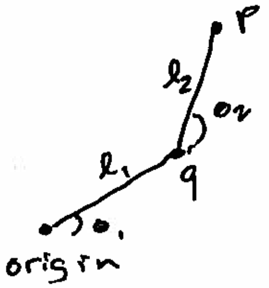
$$y = Ax + b$$

We can represent affine transform using single matrix multiplication:

$$\begin{bmatrix} y \\ 1 \end{bmatrix} = \begin{bmatrix} A & b \\ 0 \dots 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ 1 \end{bmatrix}$$

Why do this?

- one operation, one data structure (w/ waste)
- composition of transformations



Let's assume we want to find point p's position.

Position of point q:

$$R(\theta_1) \begin{bmatrix} l_1 \\ 0 \end{bmatrix}$$

Position of point p:

$$R(\theta_1)\, R(\theta_2) \begin{bmatrix} l_2 \\ 0 \end{bmatrix} + q$$

- This is a pain for longer robot arms...
- We can also look at this as a sequence of the following steps: (seen visually by going 'up' the arm)
  - ~~go along +x for l_1 units~~
  - ~~rotate by θ~~
  - ~~go along +x' for l_2 units~~
  - ~~rotate by θ_2~~ around q

  - rotate by θ_1
  - move l_1 units
  - rotate by θ_2
  - move l_2 units
- This is equiv. to:

$$\cancel{X(l_2)\, R(\theta_2)\, X(l_1)\, R(\theta_1)}$$

$$R(\theta_1)\, X(l_1)\, R(\theta_2)\, X(l_2) \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

where

$$R = \begin{bmatrix} c_\theta & -s_\theta & 0 \\ s_\theta & c_\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad X = \begin{bmatrix} 1 & 0 & l_\ell \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

concatenating transformations

A sequence of transforms $T_1 \cdot T_2 \cdot \ldots \cdot T_n$ is equiv. to Transforming by T_n, then transforming by $T_{n-1}, \ldots,$ followed by transforming by T_1

$$T_1 \cdot T_2 \cdot T_3 \cdot p = p'''$$

$$p' = T_3 \cdot p$$

$$p'' = T_2 \cdot p'$$

$$p''' = T_1 \cdot p''$$

Q: how to debug transformation & concatenations?

3D Rotations

one 3D rotation is just rotation around 'z' axis:

$$R_z = \begin{bmatrix} c_\theta & -s_\theta & 0 \\ s_\theta & c_\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

(others)

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\theta & s_\theta \\ 0 & -s_\theta & c_\theta \end{bmatrix}$$

$$R_y = \begin{bmatrix} c_\theta & 0 & -s_\theta \\ s_\theta & 1 & c_\theta \\ 0 & 0 & 0 \end{bmatrix}$$

We can also think of rotating around an arbitrary (unit) vector by an angle θ

- Conversion of A.A. to rotation matrix:

$$\begin{bmatrix} 1+(1-c_\theta)(x^2-1) & -z * s_\theta + (1-c_\theta)xy & y s_\theta + (1-c_\theta)xz \\ z s_\theta + (1-c_\theta)xy & 1+(1-c_\theta)(y^2-1) & -x s_\theta + (1-c_\theta)yz \\ -y s_\theta + (1-c_\theta)xz & x s_\theta + (1-c_\theta)yz & 1+(1-c_\theta)(z^2-1) \end{bmatrix}$$

- Conversion of rot. mat. to A.A.:
  - Equations can be found online <u>or</u> Conversion to quaternion then conversion to axis-angle
  - main component:

    $\theta = \cos^{-1}\left(\left[R_{xx} + R_{yy} + R_{zz} - 1\right]/2\right)$
  - Two singularities:

    $\theta = 0, \pi$
    - $\theta = 0$ (axis is arbitrary)
    - $\theta = \pi$ (axis not arbitrary, lots of code devoted to this)

Euler angles are also susceptible to a condition called <u>Gimbal lock</u> where two axes become aligned and a loss of a degree-of-freedom occurs.

- show example (applet?)

Gimbal lock is a real (i.e., not just mathematical) phenomenon that occurs with physical mechanisms.

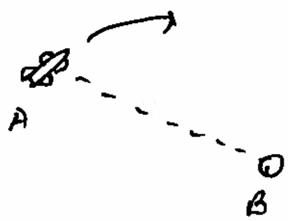
- 12 permutations on orthogonal axes are possible. If the frame of rotation rotates as well, then the Euler angles are called roll-pitch-yaw angles.
- With roll-pitch-yaw angles number of permutations rises to 24
- Best to see Ken Shoemake's algorithms (if you're going to use Euler angles)

Affine transformations

A rigid body in 3D has position completely described by a 3D position (of the c.o.m. typically) and a 3D rotation. The body's configuration can be w.r.t. a "global" origin and orientation, w.r.t. another body's position/orientation, or an arbitrary origin/orientation.

* The combination of a reference origin and orientation is known as a "frame of reference" or "frame"

3). What if frame is rotating /moving? ~~What if u vector (point on rigid body) is moving?~~



Velocity of
Pose def'd relative to global frame:

$$\begin{bmatrix} -s_\theta \cdot \dot{\theta} & -c_\theta \cdot \dot{\theta} & \dot{x} \\ c_\theta \cdot \dot{\theta} & -s_\theta \cdot \dot{\theta} & \dot{y} \\ 0 & 0 & 0 \end{bmatrix} \Rightarrow \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix}$$

Pose def'd relative to other frame:
(moving) (not moving)

$${}_B T_A = \begin{bmatrix} c_\theta & -s_\theta & x_A \\ s_\theta & c_\theta & y_A \\ 0 & 0 & 1 \end{bmatrix}$$

$$\Rightarrow \quad {}_W T_B \cdot {}_B \dot{T}_A$$

if both are moving $\Rightarrow {}_W T_B \cdot {}_B \dot{T}_A +$
$${}_W \dot{T}_B \cdot {}_B T_A$$

4) Poses in 3D

- belongs to $SE(3)$
- orientation $\in SO(3)$, position in $\mathbb{R}^3$
- $SO(3) \times \mathbb{R}^3 = SE(3)$

Rotations in 3D

- Hard to think this way
- Roll / pitch / yaw (body frame rotations)
- Euler angles (world frame rotations)
- Rotation metrices (redundant, 1 point per rotation)
- Unit quaternions (fast, 2 points for every rotation)
- axis angle (redundant, singularities)

all produce points in $SO(3)$

Rotation around z

$$\begin{bmatrix} c_\theta & -s_\theta & 0 \\ s_\theta & c_\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

X

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\theta & -s_\theta \\ 0 & s_\theta & c_\theta \end{bmatrix}$$

Y

$$\begin{bmatrix} c_\theta & 0 & s_\theta \\ 0 & 1 & 0 \\ -s_\theta & 0 & c_\theta \end{bmatrix}$$

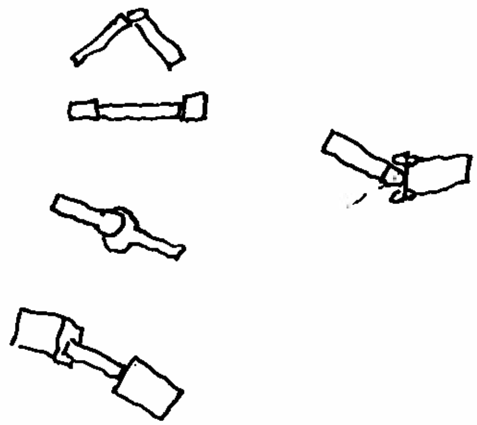
- can convert from each to rotation matrix
- in general, converting back isn't easy
- formulas to convert between each in Robotics Toolbox

5

- Rigid body degrees-of-freedom
  - 3 for translation
  - 3 for rotation

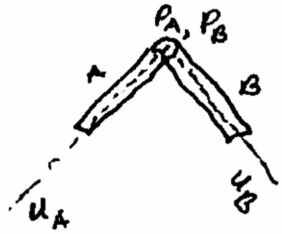
Types of joints:

- Revolute
- Prismatic
- Universal
- Spherical
- Fixed



Joints constrain two rigid bodies together by requiring bilateral constraint to be satisfied:

<u>Revolute</u>



$\| p_A - p_B \| = 0$ (3 constraints)

$u_A^T z = 0$ (1 constraint)

$u_B^T z = 0$ (1 constraint)

- z is the axis of the joint

<u>Spherical</u>

$\| p_A - p_B \| = 0$ (3 constraints)

1. ~~Forward~~ Robot Kinematics

- Body is made of (rigid) links and joints
- Base of body can be fixed or "floating"
- Degrees-of-freedom of system:
  - For fixed base, equal to # of joint axes
  - For Floating base, equal to # of joint axes plus six (DOF of free rigid body in 3D)

Another way to look at this:

- Each link has 6 DOF in 3D
- For n links, there are $6 * n$ DOF if bodies are free
- Subtract joint constraint equations and you should get same number of DOF

Ex. Two free bodies connected by revolute joint (floating base)

- Method 1: 6 DOF for floating base + 1 for revolute joint

  = 7 DOF

- Method 2: $2 * 6$ DOF − 5 constraint equations for revolute joint = 7 DOF

7] · Kinematic topologies

- Serial

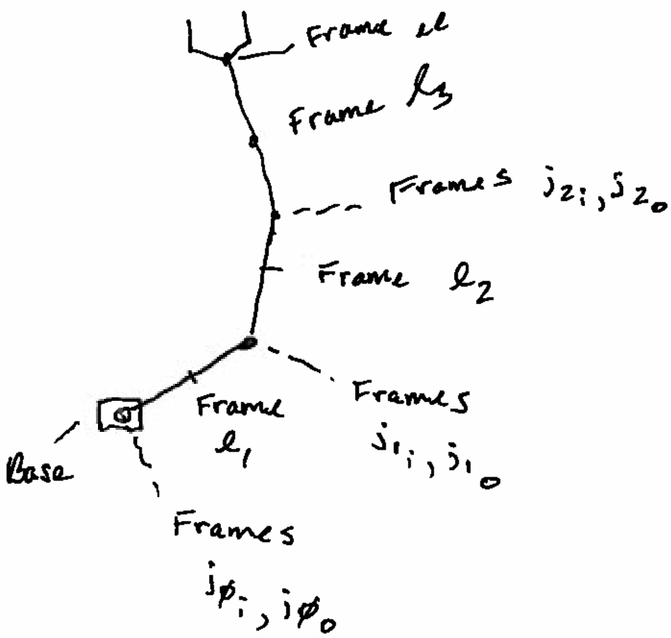
- Branched

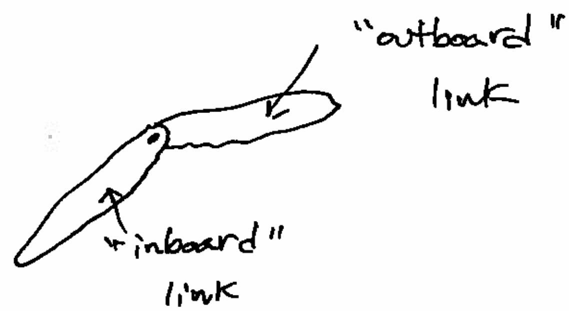
- Parallel
  (aka "closed")

Four bar mechanism

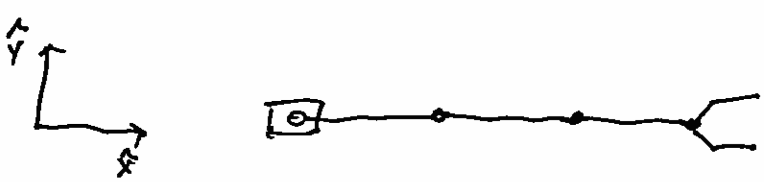
8. Forward Kinematics

Problem: Given current joint positions (angles if all revolute joints) and base pose (if floating), where is some point on the robot? Related: how fast is point on the robot moving?





Robot in "zero" configuration:



1] · We do this with a number of frame transformations

First define:

L_1 = dist. from joint ϕ to ℓ_1

L_2 = dist. from ℓ_1 to joint 1

L_3 = dist. from joint 1 to ℓ_2

L_4 = dist. from ℓ_2 to joint 2

L_5 = " " joint 2 to ℓ_3

L_6 = " " ℓ_3 to ee

$\theta_0, \theta_1, \theta_2$ are angles at respective joints

Pose at end-effector given by:

$${}^{j_{\phi_i}}T_{j_{\phi_0}} \cdot {}^{j_{\phi_0}}T_{\ell_1} \cdot {}^{\ell_1}T_{j_{1_i}} \cdot {}^{j_{1_i}}T_{j_{1_0}} \cdot {}^{j_{1_0}}T_{\ell_2} \cdot {}^{\ell_2}T_{j_{2_i}} \cdot {}^{j_{2_i}}T_{j_{2_0}} \cdot {}^{j_{2_0}}T_{\ell_3} \cdot {}^{\ell_3}T_{ee}$$

(just a series of matrix multiplications)

$${}^{j_{\phi_i}}T_{j_{\phi_0}} = \begin{bmatrix} c_{\theta_\phi} & -s_{\theta_\phi} & 0 \\ s_{\theta_\phi} & c_{\theta_\phi} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$${}^{j_{\phi_0}}T_{\ell_1} = \begin{bmatrix} 1 & 0 & L_1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$${}^{\ell_1}T_{j_{1_i}} = \begin{bmatrix} 1 & 0 & L_2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$${}^{j_{1_i}}T_{j_{1_0}} = \begin{bmatrix} c_{\theta_1} & -s_{\theta_1} & 0 \\ s_{\theta_1} & c_{\theta_1} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

10

$$ {}^{j_{1_0}}T_{l_2} = \begin{bmatrix} 1 & 0 & L_3 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{l_2}T_{j_{2_i}} = \begin{bmatrix} 1 & 0 & L_4 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{j_{2_i}}T_{j_{2_0}} = \begin{bmatrix} c\theta_2 & -s\theta_2 & 0 \\ s\theta_2 & c\theta_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{j_{2_0}}T_{l_3} = \begin{bmatrix} 1 & 0 & L_5 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{l_3}T_{ee} = \begin{bmatrix} 1 & 0 & L_6 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} $$