Robotics, Vision, and Mechatronics for Manufacturing

Chapter 3: time, motion, and derivatives of a pose

Xu Chen

Sp 2021

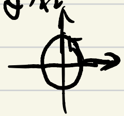
Recall: 2D

$$e^{[\theta]_\times} = R$$

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

corresponds to

$$\begin{cases} \dot{x}_1 = -\dot{\theta} x_2 \\ \dot{x}_2 = \dot{\theta} x_1 \end{cases}$$



3D rotation matrix & its systems representations

$$R = e^{[\omega]_\times \theta}$$

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

skew-symmetric matrix representing angular velocities

$$\hat{\omega} = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$$

unit vector parallel to the rotation axis

Underlying rationale:

Vector cross product

$$\|a \times b\| = \|a\|\|b\| \cdot \sin\theta$$

$$a \times b = (a_1 i + a_2 j + a_3 k) \times (b_1 i + b_2 j + b_3 k)$$

$$= a_1 i \times b_1 i + a_1 i \times b_2 j + a_1 i \times b_3 k$$

$$+ a_2 j \times b_1 i + a_2 j \times b_2 j + a_2 j \times b_3 k + \cdots$$

$$= \underline{a_1 b_2}\, k + a_1 b_3 (-j) + \underline{a_2 b_1} (-k) + \cdots$$

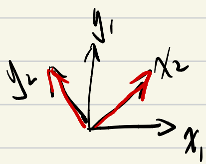
$$= \begin{vmatrix} i & j & k \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{vmatrix}$$

$$\boxed{\text{important fact!} \quad a \times b = [a]_\times b = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} -a_3 b_2 + a_2 b_3 \\ a_3 b_1 - a_1 b_3 \\ -a_2 b_1 + a_1 b_2 \end{bmatrix}}$$

3D rotation matrix & its system representation

Recall: 2D rotation

$$R=\begin{bmatrix}\cos\theta & -\sin\theta\\ \sin\theta & \cos\theta\end{bmatrix}=e^{\overbrace{\begin{bmatrix}0 & -1\\ 1 & 0\end{bmatrix}}^{[\theta]_x}\theta}$$



$$[\theta]_x=\begin{bmatrix}0 & -1\\ 1 & 0\end{bmatrix}\theta=\begin{bmatrix}0 & -\theta\\ \theta & 0\end{bmatrix}$$

$[\theta]_x$: rotation parameter

$\begin{bmatrix}0 & -1\\ 1 & 0\end{bmatrix}\triangleq[1]_x$: unit vector parallel to the rotation axis

$$\boxed{[\theta]_x=[1]_x\cdot\theta}$$

3D rotation: analogous representation

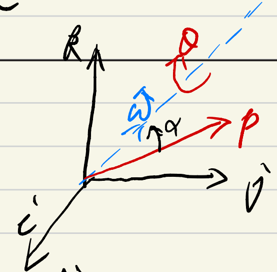
$$R=e^{[\hat{\omega}]_x\theta}$$

$\hat{\omega}=\begin{bmatrix}\hat{\omega}_x\\ \hat{\omega}_y\\ \hat{\omega}_z\end{bmatrix}$: unit vector parallel to the rotation axis

$[\hat{\omega}]_x=\begin{bmatrix}0 & -\hat{\omega}_z & \hat{\omega}_y\\ \hat{\omega}_z & 0 & -\hat{\omega}_x\\ -\hat{\omega}_y & \hat{\omega}_x & 0\end{bmatrix}$: corresponding skew-symmetric matrix

$[\hat{\omega}]_x\theta$: rotation parameters known as the exponential coordinates.

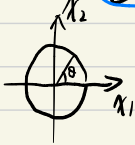
3D rotation matrix & its exponential representation $R = e^{[\hat{\omega}]_\times \theta}$



Recall: 2D rotation

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} = e^{\overbrace{\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}}^{[\theta]_\times}\theta}$$

corresponding linear system:



$$\begin{cases} x_1 = \cos\theta \\ x_2 = \sin\theta \end{cases}$$

$$\begin{cases} \dot{x}_1 = -\sin\theta\,\dot{\theta} \\ \quad = -\dot{\theta}\, x_2 \\ \dot{x}_2 = \dot{\theta}\, x_1 \end{cases}$$

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 & -\dot{\theta} \\ \dot{\theta} & 0 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

If $\dot{\theta} = \omega$ is a constant,

then $\frac{d}{dt}\begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} = \begin{bmatrix} 0 & -\omega \\ \omega & 0 \end{bmatrix}\begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}$

$$\boxed{\dot{x} = [\omega]_\times x}$$

From linear systems:

$$\begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} = e^{\begin{bmatrix} 0 & -\omega \\ \omega & 0 \end{bmatrix}t}\begin{bmatrix} x_1(0) \\ x_2(0) \end{bmatrix}$$

$$= e^{\begin{bmatrix} 0 & -\theta \\ \theta & 0 \end{bmatrix}}\begin{bmatrix} x_1(0) \\ x_2(0) \end{bmatrix}$$

$$= R(\theta)\begin{bmatrix} x_1(0) \\ x_2(0) \end{bmatrix}$$

3D rotation: analogous representation

linear rotation system

vector p rotating at angular velocity

$\omega = \hat{\omega} \cdot \dot{\theta}$ unit vector along rotation axis

velocity of p: from mechanics

$\dot{p} = \omega \times p$ $\begin{cases} \text{magnitude depends on } \|\omega\|, \|p\|, \text{ \& } \alpha \\ \text{direction is normal to } \omega \text{ \& } p(t) \end{cases}$

$$\boxed{\dot{p} = [\omega]_\times p = [\hat{\omega}\dot{\theta}]_\times p}$$

$\Rightarrow p(t) = e^{[\omega]_\times t} p(0)$, if $\dot{\theta} = \text{const}$, $p(t) = e^{[\hat{\omega}]_\times \dot{\theta} t} p(0) = e^{[\hat{\omega}]_\times \theta(t)} p(0)$

$= R(\theta) p(0)$

$$\Rightarrow \boxed{R(\theta) = e^{[\hat{\omega}]_\times \theta}}$$

cross product & skew-symmetric matrices:

$a = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = a_1 i + a_2 j + a_3 k$, $[a]_\times = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}$

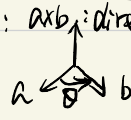
unit vectors

Fact: $\boxed{a \times b = [a]_\times b}$

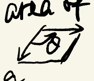
$a \times b = (a_1 i + a_2 j + a_3 k) \times (b_1 i + b_2 j + b_3 k)$

$= (a_2 b_3 - a_3 b_2) i + (a_3 b_1 - a_1 b_3) j + (a_1 b_2 - a_2 b_1) k = [a]_\times b$

Reasons: $a \times b$: direction is normal to a & b



$\|a \times b\| = \|a\|\|b\|\sin\theta$ = area of



$i \times i = 0$ $\quad i \times j = k$ $\quad i \times k = -j$

$j \times i = -k$ $\quad j \times j = 0$ $\quad j \times k = i$

$$R(\theta) = e^{[\hat{\omega}]_\times \theta}$$

$[\hat{\omega}]_\times \theta$: skew symmetric $= \begin{bmatrix} 0 & -\hat{\omega}_z & \hat{\omega}_y \\ \hat{\omega}_z & 0 & -\hat{\omega}_x \\ -\hat{\omega}_y & \hat{\omega}_x & 0 \end{bmatrix} \theta$

MATLAB: example

```
>> R = rotx(0.3)
```

$\Rightarrow R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos 0.3 & -\sin 0.3 \\ 0 & \sin 0.3 & \cos 0.3 \end{bmatrix}$

$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} 0.3$

//

```
>> S = logm(R)
```

gives $[\hat{\omega}]_\times \theta = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -0.3 \\ 0 & 0.3 & 0 \end{bmatrix}$

```
>> [theta, w_hat] = trlog(R)
```

gives theta = 0.3 w_hat = $\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$

unit vector along rotation axis (the x axis)

Time & Motion : Derivative of rotation matrix

Now we know:

$${}^{A}R_{B} = e^{[{}^{A}\hat{\omega}_{B}]_{\times}\theta(t)}$$

where:

${}^{A}\hat{\omega}_{B}$: rotation axis unit vector

$\theta(t)$: rotation angle

$[\ ]_{\times}$: skew-symmetric matrix

Consider the case when the rotation direction is fixed at t

i.e. ${}^{A}\hat{\omega}_{B}$ is fixed.

then

$$\frac{d}{dt}{}^{A}R_{B}(t) = \frac{d}{dt} e^{[{}^{A}\hat{\omega}_{B}]_{\times}\theta(t)} = [{}^{A}\hat{\omega}_{B}]_{\times}\dot{\theta}(t)\, e^{[{}^{A}\hat{\omega}_{B}]_{\times}\theta(t)} = \underbrace{[{}^{A}\hat{\omega}_{B}]_{\times}\dot{\theta}(t)}_{\text{angular velocity parameter}}\, {}^{A}R_{B}(t)$$

define, for brevity, $\omega = \hat{\omega}\cdot\dot{\theta} = \begin{bmatrix}\hat{\omega}_x \\ \hat{\omega}_y \\ \hat{\omega}_z\end{bmatrix}\dot{\theta}$

angularity velocity vector

defines instantaneous rotation axis & rate.

then $[{}^{A}\hat{\omega}_{B}]_{\times}\dot{\theta} = [{}^{A}\omega_{B}]_{\times}$

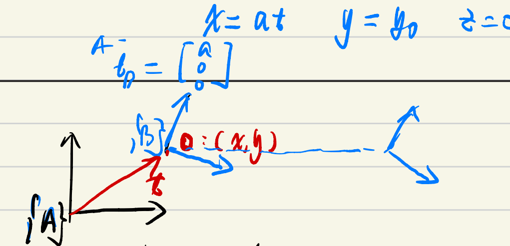
Application: if we know angular velocity in $\{A\}$

then can do: ${}^{A}\omega = {}^{A}R_{B}\,{}^{B}\omega \Rightarrow {}^{B}\omega = {}^{B}R_{A}\,{}^{A}\omega = ({}^{A}R_{B})^{T}\,{}^{A}\omega$

to obtain angular velocity in another frame.

Time & Motion: derivative of pose

just add translation now:



$$\xi \sim {}^A P_B = \begin{bmatrix} {}^A R_B & {}^A t_B \\ 0_{1\times3} & 1 \end{bmatrix}$$

$$\dot{\xi} \sim \frac{d}{dt}\, {}^A P_B = \begin{bmatrix} {}^A \dot{R}_B & {}^A \dot{t}_B \\ 0_{1\times3} & 0 \end{bmatrix}$$

${}^A \dot{R}_B$: angular velocity

${}^A \dot{t}_B$: linear/translation velocity (velocity of the origin of $\{B\}$ w.r.t. $\{A\}$)

Total velocity vector $\;{}^A \nu_B = \begin{bmatrix} {}^A \dot{t}_B & {}^A \omega_B \end{bmatrix}$

reads "nu"

Going from analysis to application: Reference Pose Generation

Goal: generate a reference pose that is smooth in translation & rotation

Defn:

path — a spatial concept ~~of~~ from an initial pose to a final pose.

trajectory — a path with specified timing

e.g. path: home to school

trajectory: home to school in 10 min

smooth — position & trajectory vary smoothly w/ time.

Generating 1D smooth trajectory

Fact: Given a finite-length path, any continuous trajectories can be approximated by a polynomial.

Common candidate:

$$s(t) = At^5 + Bt^4 + Ct^3 + Dt^2 + Et + F \qquad t \in [0, T]$$

coefficients determined by boundary conditions

t	s	$\dot{s}$	$\ddot{s}$
0	s_0	$\dot{s}_0$	$\ddot{s}_0$
T	s_T	$\dot{s}_T$	$\ddot{s}_T$

6 constraints, 6 unknowns

$\Rightarrow$ solution exists for $(A, \ldots, F)$.

$$S(t) = At^5 + Bt^4 + Ct^3 + Dt^2 + Et + F$$

specifically.

$$\dot{S}(t) = 5At^4 + 4Bt^3 + 3Ct^2 + 2Dt + E$$

$$\ddot{S}(t) = 20At^3 + 12Bt^2 + 6Ct + 2D$$

$$\Rightarrow \begin{bmatrix} S(0) \\ S(T) \\ \dot{S}(0) \\ \dot{S}(T) \\ \ddot{S}(0) \\ \ddot{S}(T) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ T^5 & T^4 & T^3 & T^2 & T & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 5T^4 & 4T^3 & 3T^2 & 2T & 1 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 20T^3 & 12T^2 & 6T & 2 & 0 & 0 \end{bmatrix} \begin{bmatrix} A \\ B \\ C \\ D \\ E \\ F \end{bmatrix}$$

$\Rightarrow$ Solve for $\begin{bmatrix} A \\ B \\ C \\ D \\ E \\ F \end{bmatrix}$

init pos

end pos

duration T

MATLAB : >> tpoly(0, 1, 50)

assume 0 boundary velocity & acceleration

>> tpoly(0, 1, 50, 0.5, 0)

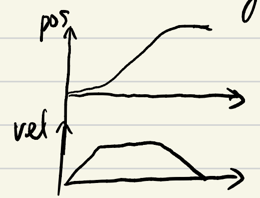
init v

final v.

>> help tpoly

In reality, robots have velocity & acceleration limits

e.g.



We often use >> lspb(0, 1, 50)

init position

final position

steps

linear speed parabolic blends

or >> lspb(0, 1, 50, 0.025)

max speed

Multi-dimensional Trajectories

Denote $q \in \mathbb{R}^N$ as a robot configuration

e.g. 3-joint robot $q = (q_1, q_2, q_3)$

wheeled mobile robot $q = (x, y)$

3D orientation $q = (\theta_r, \theta_p, \theta_y)$

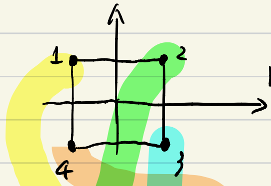
3D pose $q = (x, y, z, \theta_r, \theta_p, \theta_y)$

$$\gg \; q = \text{mtraj}\big(\underbrace{\text{@lspb}}_{\text{use lspb for each coordinate}},\ \underbrace{[0 \;\; 2]}_{\text{init } q},\ \underbrace{[1 \;\; -1]}_{\text{final } q},\ \underbrace{50}_{\text{steps}}\big)$$

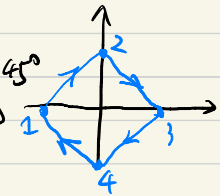
figure, plot(q)

Multi-segment trajectory (specifying waypoints between init & final poses)

Goal: move smoothly along a path through one or more intermediate points without stopping



rotate 45° $\Longrightarrow$



$$W = \begin{bmatrix} -1 & 1 & 1 & -1 \\ 1 & 1 & -1 & -1 \end{bmatrix}$$

via = SO2(45, 'deg') · W

waypoint w_1, w_2, w_3, w_4

multi-segment multi-axis trajectory

max speed along each axis

initial position @ w_1

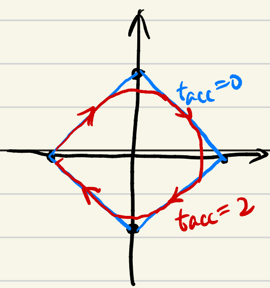
time step

t_{acc}: acceleration time (smaller gives sharper turns)

```
⇒ q0 = mstraj(via(:,[2 3 4 1])', [2, 1], [ ], via(:,1)', 0.2, 0)
```

waypoints to go
$w_2 \to w_3 \to w_4 \to w_1$

duration (empty when max speed is specified)



compare with:

```
q0 = mstraj(via(:,[2 3 4 1])', [2, 1], [ ], via(:,1)', 0.2, 2)
figure, plot(q0(:,1), q0(:,2))
```