INFO0948
Representing Positions and Orientations

Slides by Renaud Detry

Presented by Antoine Lejeune

University of Liège, Belgium

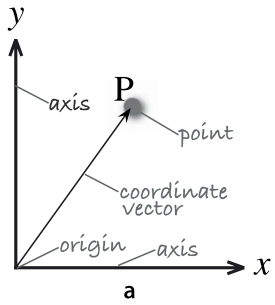
Updated February 9, 2017

These slides are based on Chapter 2 of the book *Robotics, Vision and Control: Fundamental Algorithms in MATLAB* by Peter Corke, published by Springer in 2011.

Plan

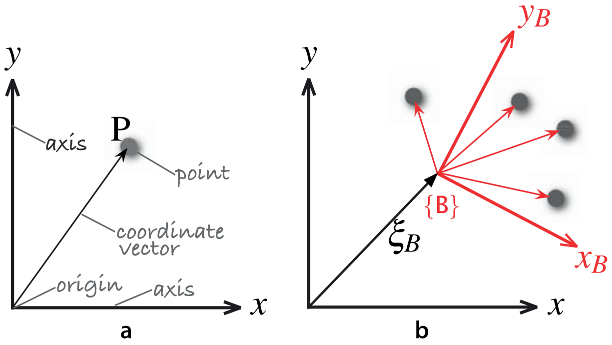
Positions and Orientations

- Need for representing positions and orientations in space.
- The position of a point is represented by a vector of coordinates.

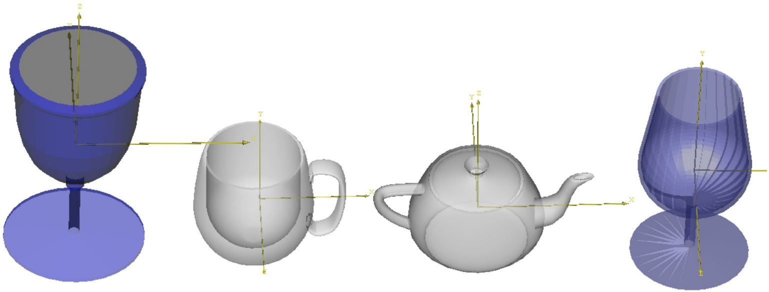


Representing a set of points

- Method 1: Represent the position and orientation of each point separately
- Method 2: Represent the points in the set's reference frame and represent the position and orientation of that frame (with ξ_B).

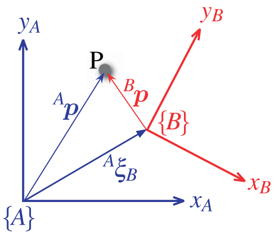


Representing a set of points



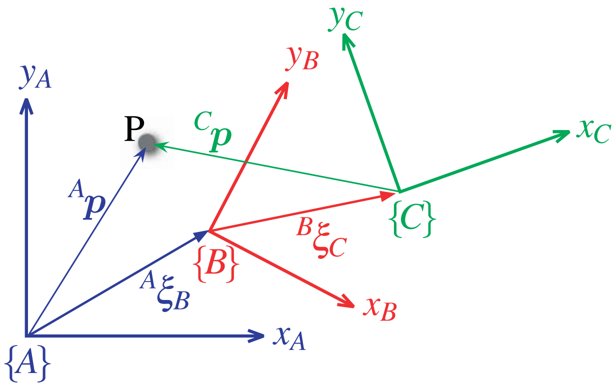
Representing a set of points

- The position and orientation of a coordinate frame is known as its pose.
- Two interpretations for ${}^{A}\xi_{B}$:
  - Picking up A and transforming it with ${}^{A}\xi_{B}$ leaves A at B's place.
  - Let ${}^{B}p$ be the coordinates of point P in the coordinate system B. Transforming ${}^{B}p$ with ${}^{A}\xi_{B}$ gives the coordinates of P in A, denoted by ${}^{A}p$.



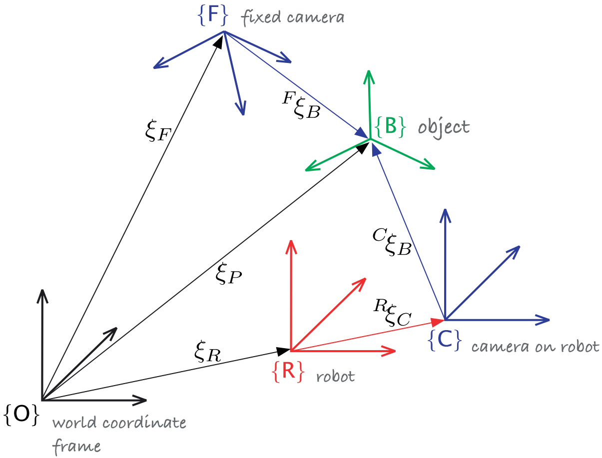
In mathematical objects terms poses constitute a group – a set of objects that supports an associative binary operator (composition) whose result belongs to the group, an inverse operation and an identity element. In this case the group is the special Euclidean group in either 2 or 3 dimensions which are commonly referred to as *SE*(2) or *SE*(3) respectively.

Composition of relative poses



$$ {}^{A}\boldsymbol{p} = ({}^{A}\xi_B \oplus {}^{B}\xi_C) \cdot {}^{C}\boldsymbol{p} $$

A 3D example



$$\xi_F \oplus {}^F\xi_B = \xi_R \oplus {}^R\xi_C \oplus {}^C\xi_B$$

There are just a few algebraic rules:

$$\xi \oplus 0 = \xi, \ \ \xi \ominus 0 = \xi$$
$$\xi \ominus \xi = 0, \ \ \ominus \xi \oplus \xi = 0$$

where 0 represents a zero relative pose. A pose has an inverse

$$\ominus {}^X\xi_Y = {}^Y\xi_X$$

which is represented graphically by an arrow from Y to X. Relative poses can also be composed or compounded

$${}^X\xi_Y \oplus {}^Y\xi_Z = {}^X\xi_Z$$

It is important to note that the algebraic rules for poses are different to normal algebra and that composition is *not* commutative

$$\xi_1 \oplus \xi_2 \neq \xi_2 \oplus \xi_1$$

with the exception being the case where $\xi_1 \oplus \xi_2 = 0$. A relative pose can transform a point expressed as a vector relative to one frame to a vector relative to another

$${}^X\boldsymbol{p} = {}^X\xi_Y \cdot {}^Y\boldsymbol{p}$$

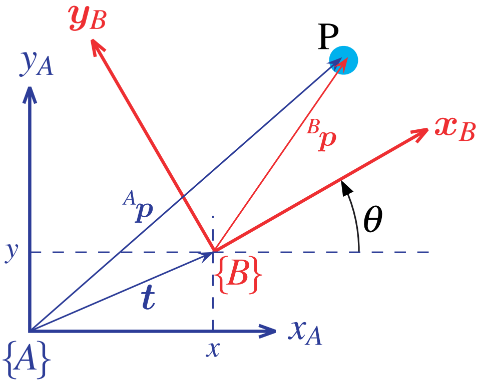
In mathematical objects terms poses constitute a group – a set of objects that supports an associative binary operator (composition) whose result belongs to the group, an inverse operation and an identity element. In this case the group is the special Euclidean group in either 2 or 3 dimensions which are commonly referred to as $SE(2)$ or $SE(3)$ respectively.

Recap

1. A point is described by a coordinate vector that represents its displacement from a reference coordinate system;
2. A set of points that represent a rigid object can be described by a single coordinate frame, and its constituent points are described by displacements from that coordinate frame;
3. The position and orientation of an object's coordinate frame is referred to as its pose;
4. A relative pose describes the pose of one coordinate frame with respect to another and is denoted by an algebraic variable ξ;
5. A coordinate vector describing a point can be represented with respect to a different coordinate frame by applying the relative pose to the vector using the $\cdot$ operator;
6. We can perform algebraic manipulation of expressions written in terms of relative poses.

Plan

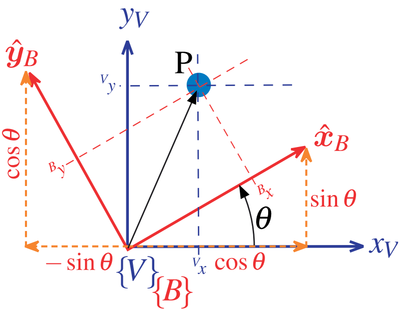
Representing poses in 2D



Is this a good representation?

$(x_1, y_1, \theta_1) \oplus (x_2, y_2, \theta_2)$

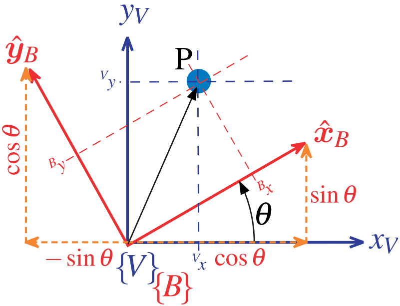
Representing orientations in 2D: rotation matrices



$$\begin{pmatrix} {}^V x \\ {}^V y \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} {}^B x \\ {}^B y \end{pmatrix} \qquad \begin{pmatrix} {}^V x \\ {}^V y \end{pmatrix} = {}^V\boldsymbol{R}_B \begin{pmatrix} {}^B x \\ {}^B y \end{pmatrix}$$

- The rotation matrix that applies a $\{V\}$-to-$\{B\}$ coordinate change corresponds to a *counter-clockwise* rotation by θ.
- The lines of ${}^V R_B$ correspond to the unit vectors that define $\{V\}$ with respect to $\{B\}$.

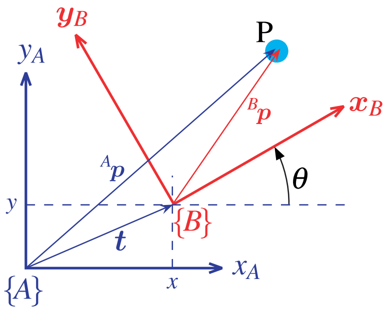
Representing orientations in 2D: rotation matrices



$$\begin{pmatrix} {}^V x \\ {}^V y \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} {}^B x \\ {}^B y \end{pmatrix} \qquad \begin{pmatrix} {}^V x \\ {}^V y \end{pmatrix} = {}^V\boldsymbol{R}_B \begin{pmatrix} {}^B x \\ {}^B y \end{pmatrix}$$

$$\begin{pmatrix} {}^B x \\ {}^B y \end{pmatrix} = ({}^V\boldsymbol{R}_B)^{-1} \begin{pmatrix} {}^V x \\ {}^V y \end{pmatrix} = ({}^V\boldsymbol{R}_B)^T \begin{pmatrix} {}^V x \\ {}^V y \end{pmatrix} = {}^B\boldsymbol{R}_V \begin{pmatrix} {}^V x \\ {}^V y \end{pmatrix}$$

Representing poses in 2D: transformation matrices



$$\begin{pmatrix} {}^Ax \\ {}^Ay \end{pmatrix} = \begin{pmatrix} {}^Vx \\ {}^Vy \end{pmatrix} + \begin{pmatrix} x \\ y \end{pmatrix}$$

$$= \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} {}^Bx \\ {}^By \end{pmatrix} + \begin{pmatrix} x \\ y \end{pmatrix}$$

$$= \begin{pmatrix} \cos\theta & -\sin\theta & x \\ \sin\theta & \cos\theta & y \end{pmatrix} \begin{pmatrix} {}^Bx \\ {}^By \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} {}^Ax \\ {}^Ay \\ 1 \end{pmatrix} = \begin{pmatrix} {}^A\boldsymbol{R}_B & \boldsymbol{t} \\ \boldsymbol{0}_{1\times 2} & 1 \end{pmatrix} \begin{pmatrix} {}^Bx \\ {}^By \\ 1 \end{pmatrix}$$

Now we can define a concrete representation of 2D poses

A concrete representation of relative pose ξ is $\xi \sim \boldsymbol{T} \in SE(2)$ and $\boldsymbol{T}_1 \oplus \boldsymbol{T}_2 \mapsto \boldsymbol{T}_1 \boldsymbol{T}_2$ which is standard matrix multiplication.

$$\boldsymbol{T}_1 \boldsymbol{T}_2 = \begin{pmatrix} \boldsymbol{R}_1 & \boldsymbol{t}_1 \\ \boldsymbol{0}_{1\times 2} & 1 \end{pmatrix} \begin{pmatrix} \boldsymbol{R}_2 & \boldsymbol{t}_2 \\ \boldsymbol{0}_{1\times 2} & 1 \end{pmatrix} = \begin{pmatrix} \boldsymbol{R}_1 \boldsymbol{R}_2 & \boldsymbol{t}_1 + \boldsymbol{R}_1 \boldsymbol{t}_2 \\ \boldsymbol{0}_{1\times 2} & 1 \end{pmatrix}$$

One of the algebraic rules from page 18 is $\xi \oplus 0 = \xi$. For matrices we know that $\boldsymbol{T}\boldsymbol{I} = \boldsymbol{T}$, where $\boldsymbol{I}$ is the identify matrix, so for pose $0 \mapsto \boldsymbol{I}$ the identity matrix. Another rule was that $\xi \ominus \xi = 0$. We know for matrices that $\boldsymbol{T}\boldsymbol{T}^{-1} = \boldsymbol{I}$ which implies that $\ominus \boldsymbol{T} \mapsto \boldsymbol{T}^{-1}$

$$\boldsymbol{T}^{-1} = \begin{pmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0}_{1\times 2} & 1 \end{pmatrix}^{-1} = \begin{pmatrix} \boldsymbol{R}^T & -\boldsymbol{R}^T \boldsymbol{t} \\ \boldsymbol{0}_{1\times 2} & 1 \end{pmatrix}$$

For a point $\tilde{\boldsymbol{p}} \in \mathbb{P}^2$ then $\boldsymbol{T} \cdot \tilde{\boldsymbol{p}} \mapsto \boldsymbol{T}\tilde{\boldsymbol{p}}$ which is a standard matrix-vector product.

Peter Corke's Robotics Toolbox

Homepage: `http://petercorke.com/Robotics_Toolbox.html`

The file you need to download is: `http://www.petercorke.com/RTB/dl-zip.php?file=current/robot-9.8.zip`

To install the Toolbox simply unpack the archive which will create the directory rvctools, and within that the directories robot, simulink, and common:

- Adjust your MATLABPATH to include rvctools:
  - Either via the Matlab menu
  - or via

    ```
    >> path(path, "/home/username/.../rcvtools")
    ```
- Run

  ```
  >> rvctools/startup_rvc.m
  ```
- Run the demo command (website's *rtdemo* is incorrect)

  ```
  >> rtbdemo
  ```

Plan

Plan

Representing 3D Orientations with Rotation Matrices

3D rotations around the origin of $\mathbb{R}^3$ form the rotation group.

3D rotations can be uniquely parametrized by special orthogonal matrices (rotation matrices).

$$\begin{pmatrix} {}^{A}x \\ {}^{A}y \\ {}^{A}z \end{pmatrix} = {}^{A}\boldsymbol{R}_{B} \begin{pmatrix} {}^{B}x \\ {}^{B}y \\ {}^{B}z \end{pmatrix}$$

Compounding still holds: ${}^{C}R_{A} = {}^{C}R_{B}\,{}^{B}R_{A}$.

Rows of a rotation matrix give the directions of the new frame's axes relative to the current frame.

More parameters than degrees of freedom.

The rotation group is often referred to as the special orthogonal group $SO(3)$.

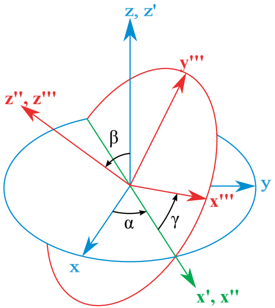
Plan

Representing 3D Orientations with Three Angles

Three-angle representation with intrinsic rotations

Given three angles (α, β, γ), let a-b'-c'' denote the application of

- a rotation of α around axis a,
- a rotation of β around axis b',
  b' is b rotated by α around a
- and a rotation of γ around axis c'' (in that order), c'' is c rotated by α around a, then by β around b'



Three angles (α, β, γ), 12 representations of a 3D rotation:

- Euler angles: z-x'-z'', x-y'-x'', y-z'-y'', z-y'-z'', x-z'-x'', y-x'-y''.
- Tait–Bryan angles: x-y'-z'', y-z'-x'', z-x'-y'', x-z'-y'', z-y'-x'', y-x'-z''.

This is called the three-angle representation with intrinsic rotations.

Representing 3D Orientations with Three Angles

Three-angle representation with extrinsic rotations

Given three angles (α, β, γ), let a-b-c denote the application of

- a rotation of α around axis a,
- a rotation of β around axis b,
- and a rotation of γ around axis c (in that order).

Three angles (α, β, γ), another 12 representations of a 3D rotation:

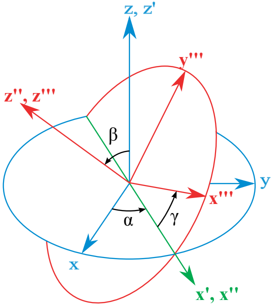
- Euler angles: z-x-z, x-y-x, y-z-y, z-y-z, x-z-x, y-x-y.
- Tait–Bryan angles: x-y-z, y-z-x, z-x-y, x-z-y, z-y-x, y-x-z.

This is called the three-angle representation with extrinsic rotations.

Visualizing Intrinsic Three-angles

Example: Given three angles (α, β, γ), z-x'-z'' denotes the application of

- a rotation of α around axis z,
- a rotation of β around axis x',
  x' is x rotated by α around z
- and a rotation of γ around axis z'' (in that order),
  z'' is z rotated by α around z, then by β around x'



Three-angles: Matrix Equivalents

Intrinsic rotations equivalent with rotation matrices:

- x-y'-z'': $R_x(\alpha)R_y(\beta)R_z(\gamma)$
- z-x'-z'': $R_z(\alpha)R_x(\beta)R_z(\gamma)$

Compare with:

Extrinsic rotations equivalent with rotation matrices:

- x-y-z: $R_z(\gamma)R_y(\beta)R_x(\alpha)$
- z-x-z: $R_z(\gamma)R_x(\beta)R_z(\alpha)$

Converting Between Three-angle Representations

Let R be a rotation whose Euler angles are $(.1, .2, .3)$ (intrinsic y-z'-y''). Give at least one extrinsic Euler representation of R.

Converting Between Three-angle Representations

Let R be a rotation whose Euler angles are $(.1, .2, .3)$ (intrinsic y-z'-y''). Give at least one extrinsic Euler representation of R.

$R = R_y(.1)R_z(.2)R_y(.3)$
$(.3, .2, .1)$ is the Euler y-z-y (extrinsic) representation of R.

There Are 24 Three-angles Representations

Extrinsic Euler: z-x-z, x-y-x, y-z-y, z-y-z, x-z-x, y-x-y.
Intrinsic Euler: z-x'-z'', x-y'-x'', y-z'-y'', z-y'-z'', x-z'-x'', y-x'-y''.
Extrinsic Tait-Bryan: x-y-z, y-z-x, z-x-y, x-z-y, z-y-x, y-x-z.
Intrinsic Tait-Bryan: x-y'-z'', y-z'-x'', z-x'-y'', x-z'-y'', z-y'-x'', y-x'-z''.

When you read a book, or code documentations, you almost never know if the author is using intrinsic or extrinsic ($'$ and $''$ are omitted). The Corke book doesn't say it's using intrinsic rotations.

Often, you will not know which of the Euler or TB representation the author is using.

With TB representations, some authors give the three angles in inverse order (i.e., in the order of the rotation matrix form).

All of the above are the first reason why three-angles are horrible.

Why Three-angles Are Horrible

1. 24 representations, plus different names (TB also called yaw-pitch-roll, TB sometimes referred to as Euler).
2. Singularities: Assuming Euler z-y'-z'', $(\alpha, 0, \gamma) \sim (\alpha + c, 0, \gamma - c) \forall c \in \mathbb{R}$
3. Numerical instability.

Three-angle representations are horrible.

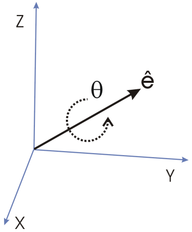
You must not use them in this class.

However, you will come across them outside of this class. Thus you must know how to read them.

Plan

Representing Rotations with Axis-angles

Any 3D rotation can be expressed as a rotation of θ around an axis v.



This is an intuitive representation, but compositions are not as easy as with rotation matrices.

Plan

Quaternions: History & Definition

- Extension of complex numbers:

$$\begin{aligned}\mathring{q} &= \underbrace{w}_{scalar} + \underbrace{xi + yj + zk}_{pure,vector} \quad \in \quad \mathbb{H} \\ &= w \left\langle x, y, z \right\rangle\end{aligned}$$

with

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

$$\begin{array}{rclcrcl} ij &=& k, & \qquad & ji &=& -k, \\ jk &=& i, & & kj &=& -i, \\ ki &=& j, & & ik &=& -j. \end{array}$$

- First described by the Irish mathematician Sir W. Hamilton in 1843.
- Now replaced by vectors in most applications...
- Except for representing 3D rotations, where they work very well.

Quaternion Arithmetic: Sum, Product

Let

$$
\begin{aligned}
\mathring{x} &= 3 + i \\
\mathring{y} &= 5i + j - 2k
\end{aligned}
$$

Then

$$
\mathring{x} + \mathring{y} = 3 + 6i + j - 2k
$$

$$
\begin{aligned}
\mathring{x}\mathring{y} &= (3+i)(5i + j - 2k) \\
&= 15i + 3j - 6k + 5i^2 + ij - 2ik \\
&= 15i + 3j - 6k - 5 + k + 2j \\
&= -5 + 15i + 5j - 5k
\end{aligned}
$$

$$
\begin{aligned}
\mathring{y}\mathring{x} &= (5i + j - 2k)(3+i) \\
&= 15i + 5i^2 + 3j + ji - 6k - 2ki \\
&= 15i - 5 + 3j - k - 6k - 2j \\
&= -5 + 15i + j - 7k
\end{aligned}
$$

Quaternion Arithmetic: Product

Let $\mathring{p} = a + bi + cj + dk$ and $\mathring{q} = w + xi + yj + zk$.

- Usual non-commutative multiplication "Grassmann product":

$$\begin{aligned} \mathring{p}\mathring{q} &= (aw - bx - cy - dz) + (bw + ax + cz - dy)i + \\ &\quad (cw + ay + dx - bz)j + (dw + az + by - cx)k \\ &\in \mathbb{H} \end{aligned}$$

- Dot product:
  - $$\mathring{p} \cdot \mathring{q} = aw + bx + cy + dz \ \in \mathbb{R}$$
  - Also denoted by $\mathring{p}^{\mathsf{T}}\mathring{q}$

Quaternion Arithmetic: Conjugate, Absolute value, Inverse

$$\mathring{z} = a + bi + cj + dk$$

- Conjugate: $\mathring{z}^* = a - bi - cj - dk$.
- Absolute value: $|\mathring{z}| = \sqrt{\mathring{z}\mathring{z}^*} = \sqrt{a^2 + b^2 + c^2 + d^2}$.
- Inverse: $\mathring{z}^{-1} = \frac{\mathring{z}^*}{\mathring{z}\mathring{z}^*}$ $\qquad (\mathring{z}^{-1}\mathring{z} = 1)$

For unit quaternions, $\mathring{q}^* = \mathring{q}^{-1}$

Representing 3D Rotations with Unit Quaternions

- A rotation of θ rad about unit vector $v = (v_x, v_y, v_z)$ is represented with
$$\mathring{q} = \cos\frac{\theta}{2} + \sin\frac{\theta}{2}\,(v_x i + v_y j + v_z k)$$
- Intuitively equivalent to a rotation of $-\theta$ about $-v$.
- Unit quaternions $\mathring{q} \in \mathbb{S}^3 \subset \mathbb{H}$ form a double cover of $SO(3)$, $\mathring{q}$ and $-\mathring{q}$ represent the same rotation.

Unit quaternions for a 3D rotation

Rotating $s = (s_x, s_y, s_z)$ with $\mathring{q}$:

$$is^r_x + js^r_y + ks^r_z = \mathring{q}(is_x + js_y + ks_z)\mathring{q}^{-1}$$

Composing $\mathring{q}_1$ and $\mathring{q}_2$: $\mathring{q}_1\mathring{q}_2$

Inverse rotation of $\mathring{q}$: $\mathring{q}^{-1}$

Distance Between Two Rotations

The distance between two rotations $\mathring{q}$ and $\mathring{q}'$ is often defined as the angle of the 3D rotation that maps $\mathring{q}$ onto $\mathring{q}'$.

This angle is equal to twice the shortest path between $\mathring{q}$ and $\mathring{q}'$ on the 3–sphere:

$$\text{distance}(\mathring{q}, \mathring{q}') = 2\arccos\left|\mathring{q}^\top \mathring{q}'\right|,$$

Caution: we take the absolute value $|\mathring{q}^\top \mathring{q}'|$ to take into account the double cover issue mentioned above.

Benefits of Unit Quaternions for Representing 3D Rotations

Not even speaking about three-angles.
Axis-angles are awkward to compose.

Compared to rotation matrices:

- Compact representation;
- More efficient for composition;
- Intuitive metric;
- Smooth interpolation;
- Robustness to numerical drift

Plan

Representing 3D transformation/poses with 3-vectors and Quaternions

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

Composition is defined by

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

and negation is

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

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

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

Representing 3D transformations/poses with Homogeneous Transformations

A concrete representation of relative pose ξ is $\xi \sim \boldsymbol{T} \in SE(3)$ and $\boldsymbol{T}_1 \oplus \boldsymbol{T}_2 \mapsto \boldsymbol{T}_1 \boldsymbol{T}_2$ which is standard matrix multiplication.

$$\boldsymbol{T}_1 \boldsymbol{T}_2 = \begin{pmatrix} \boldsymbol{R}_1 & \boldsymbol{t}_1 \\ \mathbf{0}_{1\times 3} & 1 \end{pmatrix} \begin{pmatrix} \boldsymbol{R}_2 & \boldsymbol{t}_2 \\ \mathbf{0}_{1\times 3} & 1 \end{pmatrix} = \begin{pmatrix} \boldsymbol{R}_1 \boldsymbol{R}_2 & \boldsymbol{t}_1 + \boldsymbol{R}_1 \boldsymbol{t}_2 \\ \mathbf{0}_{1\times 3} & 1 \end{pmatrix} \tag{2.20}$$

One of the rules of pose algebra from page 18 is $\xi \oplus 0 = \xi$. For matrices we know that $\boldsymbol{T}\boldsymbol{I} = \boldsymbol{T}$, where $\boldsymbol{I}$ is the identify matrix, so for pose $0 \mapsto \boldsymbol{I}$ the identity matrix. Another rule of pose algebra was that $\xi \ominus \xi = 0$. We know for matrices that $\boldsymbol{T}\boldsymbol{T}^{-1} = \boldsymbol{I}$ which implies that $\ominus \boldsymbol{T} \mapsto \boldsymbol{T}^{-1}$

$$\boldsymbol{T}^{-1} = \begin{pmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \mathbf{0}_{1\times 3} & 1 \end{pmatrix}^{-1} = \begin{pmatrix} \boldsymbol{R}^T & -\boldsymbol{R}^T \boldsymbol{t} \\ \mathbf{0}_{1\times 3} & 1 \end{pmatrix} \tag{2.21}$$