Rotations

Rotations

3D rotations have 3 dof, No unique representation

Matrix

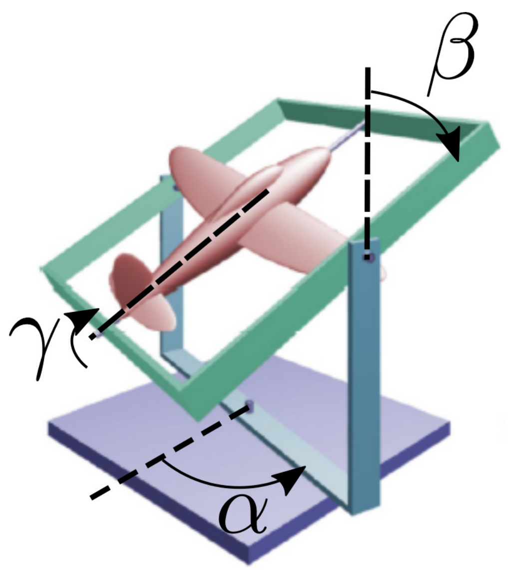
$$R = \begin{pmatrix} R_{xx} & R_{xy} & R_{xz} \\ R_{yx} & R_{yy} & R_{yz} \\ R_{zx} & R_{zy} & R_{zz} \end{pmatrix}$$

$R^T R = \mathrm{I}$
$\det(R) = 1$

Euler Angles

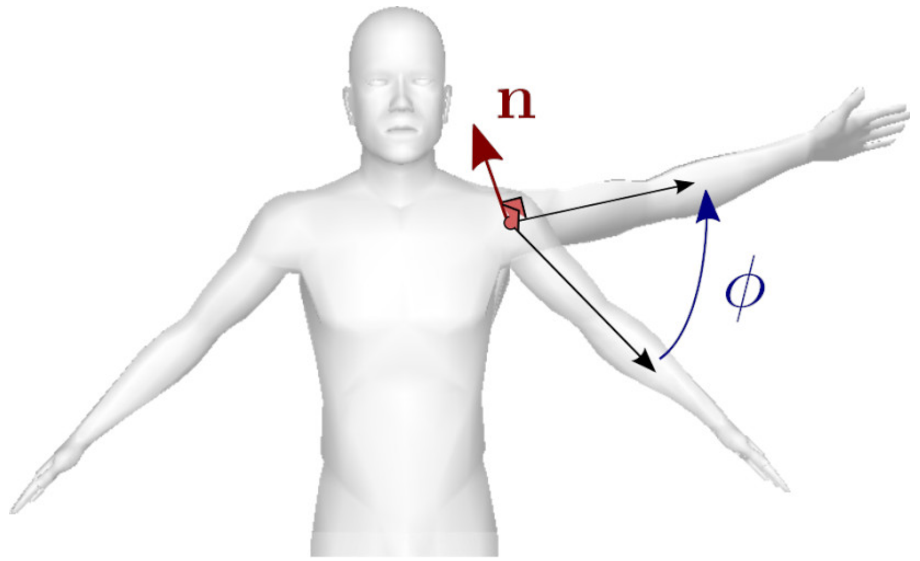
3 angles: (α, β, γ)

Composition of rotation around basic axes



Axis angle

$(\mathbf{n}, \theta)$



Quaternion

$$q = (x, y, z, w) = \left(\mathbf{n} \sin\left(\frac{\theta}{2}\right), \cos\left(\frac{\theta}{2}\right)\right)$$

Rotation: Focus Euler angles

Three consecutive rotations along fixed axis along (x, y, z) coordinates.

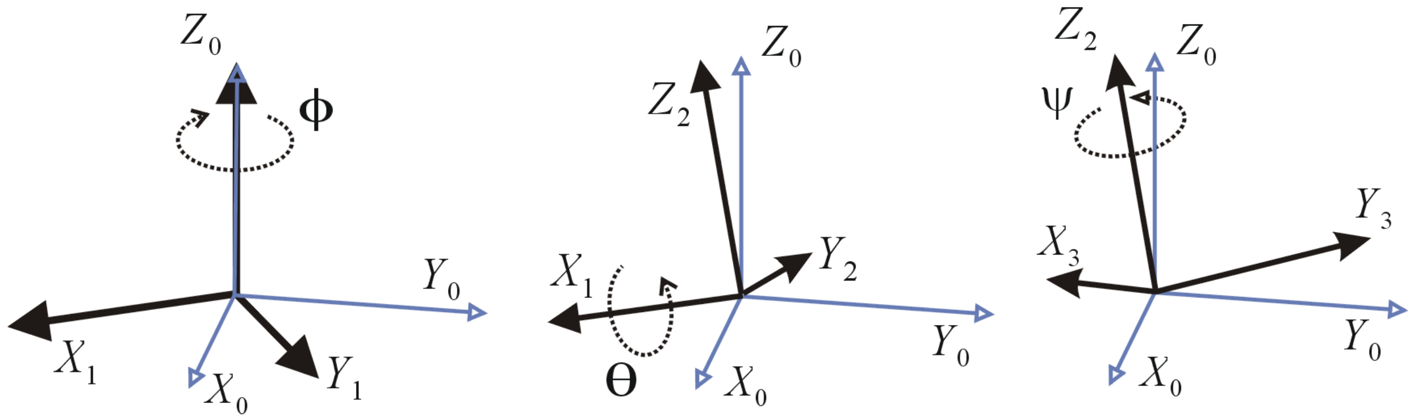
Can use basic rotation matrices composition

$$R_x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & \sin(\alpha) \\ 0 & -\sin\alpha) & \cos(\alpha) \end{pmatrix} \quad R_y = \begin{pmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{pmatrix} \quad R_z = \begin{pmatrix} \cos(\gamma) & \sin(\gamma) & 0 \\ -\sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Multiple Euler angles conventions

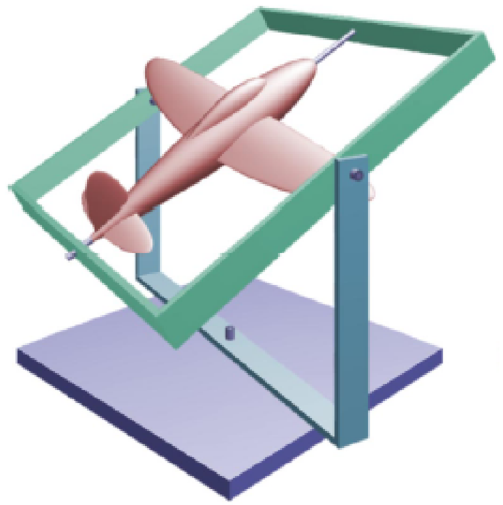
- *Proper Euler : z-x-z', x-y-x', y-z-y', ...*
- *Trait-Bryan : x-y-z, z-y-x, x-z-y, ...*

Take care when exporting/importing/parsing between Softwares



Pro

- Combination of rotation around known axis
- Comprehensive parameters (3 dof)
- Animators can interact with angular curves
- *Widely used in robotics*



Rotation: Focus Euler angles

Limitations of Euler Angle

- *Gimbal Lock* when composing b/w some rotations

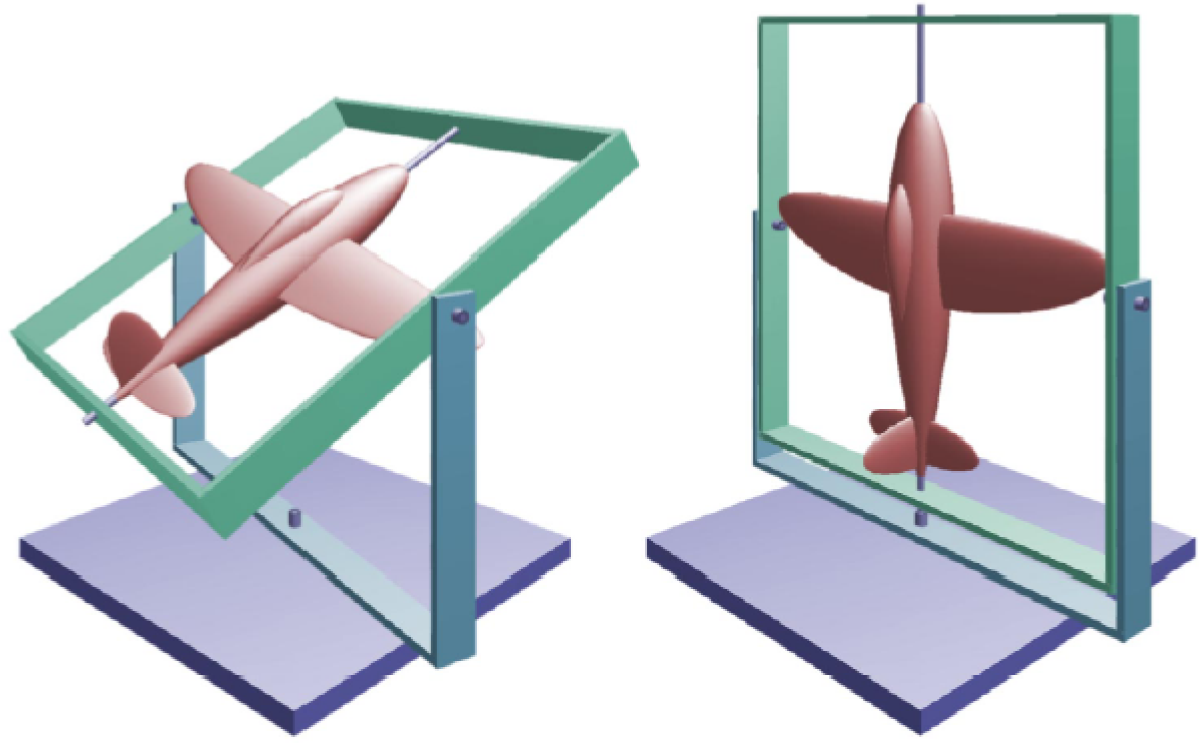
  Loose one degree of freedom in specific configuration

$$\text{ex. } R_x(\alpha)\,R_y(\pi/2)\,R_z(\gamma) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{pmatrix} \begin{pmatrix} \cos(\gamma) & -\sin(\gamma) & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} 0 & 0 & 1 \\ \sin(\alpha)\cos(\gamma) + \cos(\alpha)\sin(\gamma) & -\sin(\alpha)\sin(\gamma) + \cos(\alpha)\cos(\gamma) & 0 \\ -\cos(\alpha)\cos(\gamma) + \sin(\alpha)\sin(\gamma) & \cos(\alpha)\sin(\gamma) + \sin(\alpha)\cos(\gamma) & 0 \end{pmatrix}$$

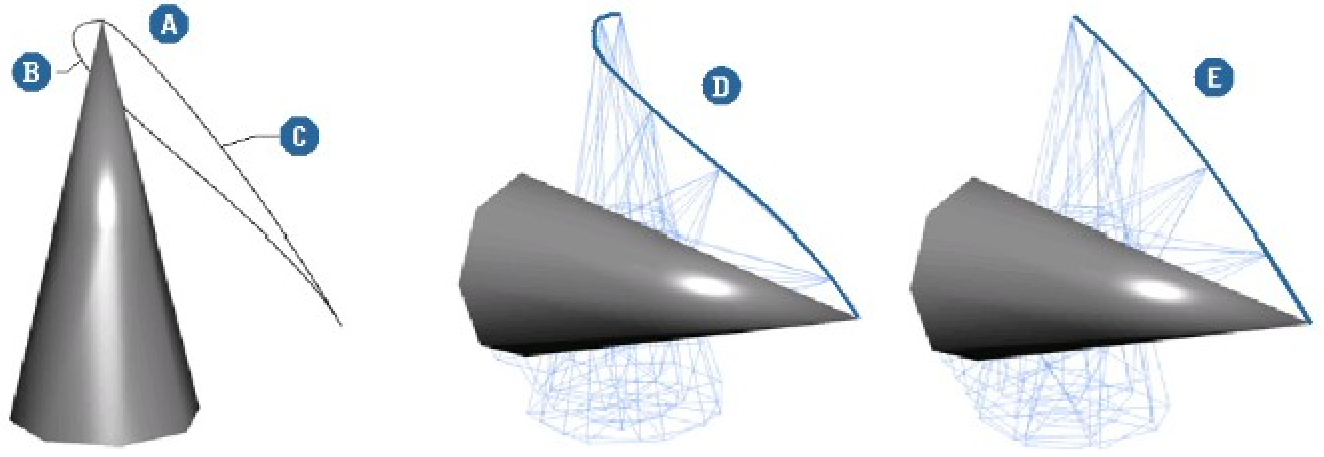
$$= \begin{pmatrix} 0 & 0 & 1 \\ \sin(\alpha+\gamma) & \cos(\alpha+\gamma) & 0 \\ -\cos(\alpha+\gamma) & \sin(\alpha+\gamma) & 0 \end{pmatrix}$$

Expect 2-dof, but get 1-dof



http://www.fho-emden.de/~hoffmann/gimbal09082002.pdf

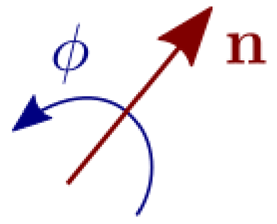
- Interpolation of 3 angles possible but not necessarily with the simplest trajectory.



Rotation: Focus Axis Angle

Any 3D rotation can be represented by

- A unit axis n
- An angle θ

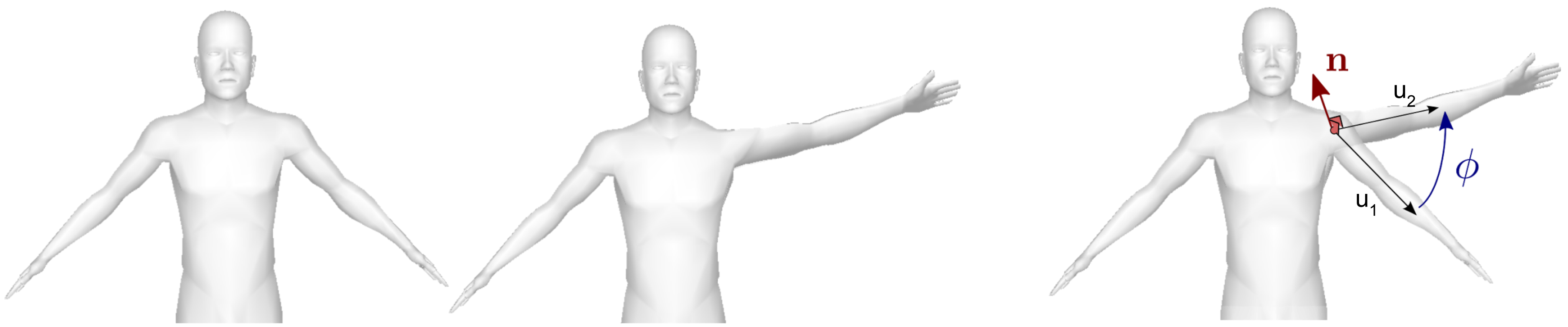


Pro.

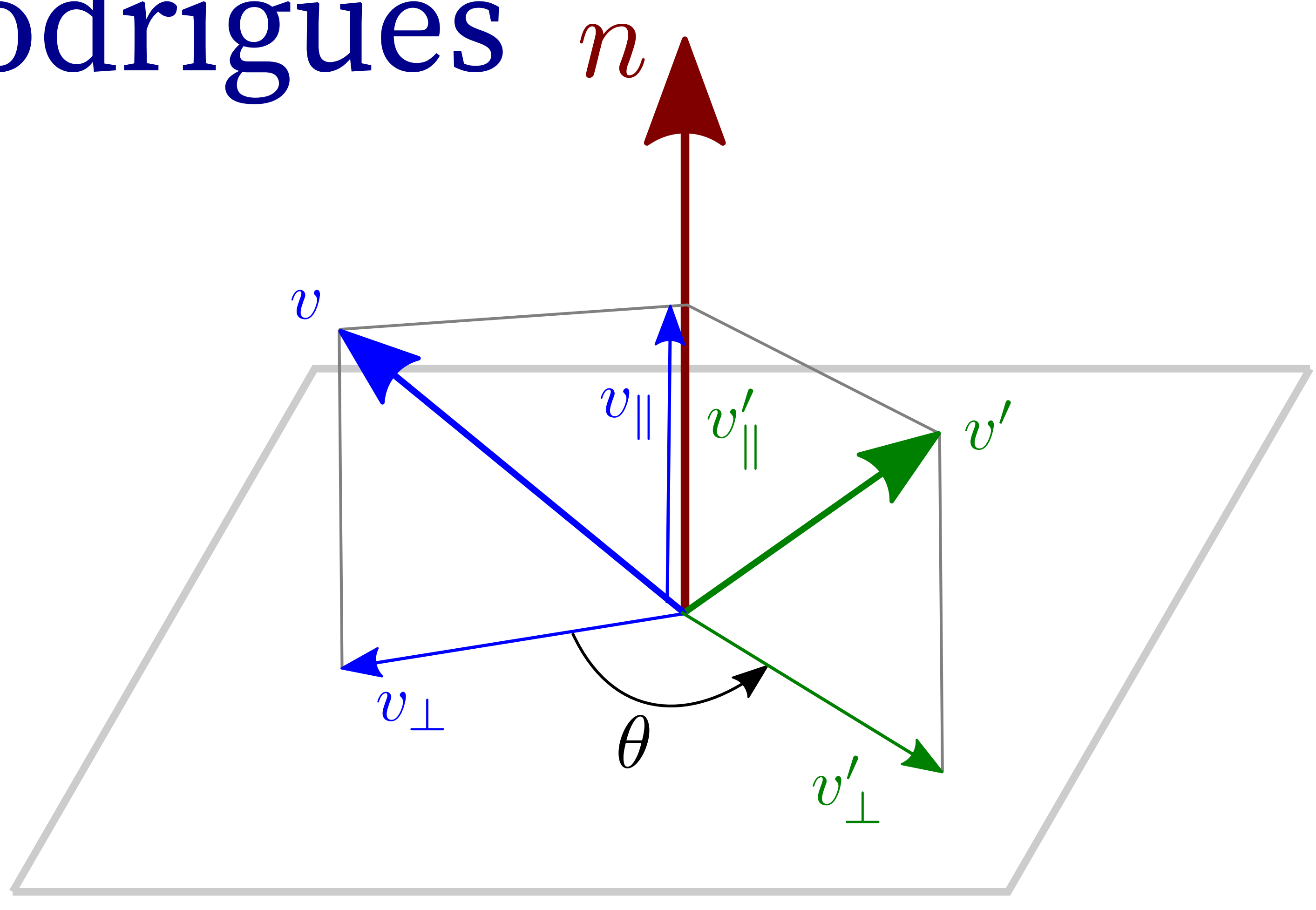
- Concise representation: 3 DOF
- Meaningfull parameters
- Describes well the rotation between two vectors

Ex. Rotation between u_1 and u_2

- Axis of rotation $n = (u_1 \times u_2)/\|u_1 \times u_2\|$
- Angle of rotation $\theta = \text{acos}(u_1 \cdot u_2)$
- Twist around u_2 can be arbitrarily chosen



Rotation: Focus Axis Angle - Rodrigues



Applying a rotation (n, θ) to a vector v

$$v = v_{\parallel} + v_{\perp}$$
$$v' = v'_{\parallel} + v'_{\perp}$$
$$\Rightarrow v' = v_{\parallel} + (\cos(\theta)\, v_{\perp} + \sin(\theta)\, n \times v_{\perp})$$

$$v_{\parallel} = (v \cdot n)\, n$$
$$v_{\perp} = v - (v \cdot n)\, n$$

$$v' = (v \cdot n)\, n + \cos(\theta)(v - (v \cdot n)\, n) + \sin(\theta)\, n \times (v - (v \cdot n)\, n)$$

$$\Rightarrow v' = \cos(\theta)\, v + \sin(\theta)\, n \times v + (1 - \cos(\theta))\, (v \cdot n)\, n$$

Rodrigues' rotation Formula

Rotation: Focus Axis Angle as Matrix

$$v' = \cos(\theta)\, v + \sin(\theta)\, n \times v + (1 - \cos(\theta))\, (v \cdot n)\, n = R(n, \theta)\, v$$

$$v' = \cos(\theta) v + \sin(\theta) \begin{pmatrix} n_x \\ n_y \\ n_z \end{pmatrix} \times v + (1 - \cos(\theta)) \begin{pmatrix} n_x & n_y & n_z \end{pmatrix} v \begin{pmatrix} n_x \\ n_y \\ n_z \end{pmatrix}$$

$$v' = \cos(\theta)\, v + \sin(\theta) \underbrace{\begin{pmatrix} 0 & -n_z & n_y \\ n_z & 0 & -n_x \\ -n_y & n_x & 0 \end{pmatrix}}_{\mathrm{K}} v + (1 - \cos(\theta)) \underbrace{\begin{pmatrix} n_x^2 & n_x\, n_y & n_x\, n_z \\ n_x\, n_y & n_y^2 & n_y\, n_z \\ n_x\, n_z & n_y\, n_z & n_z^2 \end{pmatrix}}_{\mathrm{K}^2} v$$

$$v' = \begin{pmatrix} \cos(\theta) + n_x^2(1 - \cos(\theta)) & n_x n_y(1 - \cos(\theta)) - n_z \sin(\theta) & n_x n_z(1 - \cos(\theta)) + n_y \sin(\theta) \\ n_x n_y(1 - \cos(\theta)) + n_z \sin(\theta) & \cos(\theta) + n_y^2(1 - \cos(\theta)) & n_y n_z(1 - \cos(\theta)) - n_x \sin(\theta) \\ n_x n_z(1 - \cos(\theta)) - n_y \sin(\theta) & n_y n_z(1 - \cos(\theta)) + n_x \sin(\theta) & \cos(\theta) + n_z^2(1 - \cos(\theta)) \end{pmatrix} v$$

Rotation: Focus Axis Angle - Summary

Given a rotation (n, θ) - Corresponding rotation matrix

$$R(n,\theta) = \mathrm{I} + \sin(\theta)\,\mathrm{K} + (1-\cos(\theta))\,\mathrm{K}^2$$

$$R(n,\theta) = \begin{pmatrix} \cos(\theta) + n_x^2(1-\cos(\theta)) & n_x n_y(1-\cos(\theta)) - n_z\sin(\theta) & n_x n_z(1-\cos(\theta)) + n_y\sin(\theta) \\ n_x n_y(1-\cos(\theta)) + n_z\sin(\theta) & \cos(\theta) + n_y^2(1-\cos(\theta)) & n_y n_z(1-\cos(\theta)) - n_x\sin(\theta) \\ n_x n_z(1-\cos(\theta)) - n_y\sin(\theta) & n_y n_z(1-\cos(\theta)) + n_x\sin(\theta) & \cos(\theta) + n_z^2(1-\cos(\theta)) \end{pmatrix}$$

Pro

- Concise and general representation
- Expressive parameters in 3D space (axe, angles)
- Efficient rotation and correspondance with matrix

Cons

- No simple composition expression between two rotations.
- No direct interpolation

(well handled by quaternion representation)

Rotation: Focus on Quaternions

Quaternions: generalization of complex numbers.

$$\boxed{q = x\,\mathbf{i} + y\,\mathbf{j} + z\,\mathbf{k} + w}$$

w real part, (x, y, z) imaginary (or pure quaternion) part.

We write in short $q = (x, y, z, w)$

(don't confound with 4D vectors in homogeneous coordinates)

Properties of imaginary basis vectors

$$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = -1$$
$$\mathbf{ij} = -\mathbf{ji} = \mathbf{k}$$
$$\mathbf{jk} = -\mathbf{kj} = \mathbf{i}$$
$$\mathbf{ki} = -\mathbf{ik} = \mathbf{j}$$
$$\mathbf{ijk} = -1$$

- Provides the algebraic properties

Basics operations on quaternions

- Conjugated quaternion $q^\star = (-x, -y, -z, w)$

- Quaternion norm $\|q\| = \sqrt{q\,q^\star} = \sqrt{x^2 + y^2 + z^2 + w^2}$. Unit quaternion satisfies $\|q\| = 1$.

- Quaternion product

$$q_1\ q_2 = (x_1\,\mathbf{i} + y_1\,\mathbf{j} + z_1\,\mathbf{k} + w_1)\ (x_2\,\mathbf{i} + y_2\,\mathbf{j} + z_2\,\mathbf{k} + w_2) = \ \ldots$$

$$q_1\ q_2 = \begin{pmatrix} x_1w_2 + w_1x_2 + y_1z_2 - z_1y_2 \\ y_1w_2 + w_1y_2 + z_1x_2 - x_1z_2 \\ z_1w_2 + w_1z_2 + x_1y_2 - y_1x_2 \\ w_1w_2 - x_1x_2 - y_1y_2 - z_1z_2 \end{pmatrix}$$

Note: Quaternion product is
- associative: $(q_1q_2)q_3 = q_1(q_2q_3) = q_1\,q_2\,q_3$
- **non-commutative**: $q_1\,q_2 \neq q_2\,q_1$

Sometimes interesting to separate *real part* w from *pure quaternion* part $s = (x, y, z)$.
- Shorthand vector form $q = (s, w)$
- Quaternion product in vector form $q_1\ q_2 = (s_1\,w_2 + s_2\,w_1 + s_1 \times s_2\ ,\ w_1\,w_2 - s_1 \cdot s_2)$

Relation between quaternion and rotation

Consider

- a quaternion $q = (s, w)$ of unit norm $\|q\| = 1$.
- a vector $v = (v_x, v_y, v_z)$ assimilated to the pure quaternion $q_v = (v, 0) = (v_x, v_y, v_z, 0)$.

Then

- $q_{v'} = \mathcal{R}_q(v) = q\, q_v\, q^\star$ is a pure quaternion $q_{v'} = (v'_x, v'_y, v'_z, 0)$
- And $v' = (v'_x, v'_y, v'_z)$ is the rotation of vector v around the axis $n = s/\|s\|$, with angle $2\ acos(w)$.

Demonstration

$$\mathcal{R}_q(v) = (s, w)\ (v, 0)\ (-s, w) = \ \dots\ = ((w^2 - s^2)\, v + 2(s \cdot v)\, s + 2w\, (s \times v), 0)$$

As $\|q\| = 1$, we can write $q = (s, w) = (n \sin(\phi), \cos(\phi))$, where $\|n\| = 1$

Then $\mathcal{R}_q(v) = (\underbrace{(\cos^2(\phi) - \sin^2(\phi))}_{\cos(2\phi)}\, v + \underbrace{2 \sin^2(\phi)}_{1-\cos(2\phi)}(n \cdot v)\, n + \underbrace{2\ \cos(\phi) \sin(\phi)}_{\sin(2\phi)}\ n \times v\ ,\ 0)$

$\Rightarrow$ Rodrigues formula for axis n and angle 2ϕ.

The unit quaternion $q = (n\ \sin(\theta/2), \cos(\theta/2))$ represents the rotation of angle θ around the axis n.

Composition of rotations

Consider two rotation (R_1, R_2) associated to their unit quaternions (q_1, q_2).

The product $q_1\, q_2$ represents the composition $R_1 \circ R_2$.

Demonstration

We show that $\mathcal{R}_{q_1\, q_2}(v) = \mathcal{R}_{q_1} \circ \mathcal{R}_{q_2}(v)$.

$$\mathcal{R}_{q_1\, q_2}(v) = (q_1 q_2)\ v\ (q_1 q_2)^\star$$

$$\mathcal{R}_{q_1\, q_2}(v) = (q_1\, q_2)\ v\ (q_2^\star\, q_1^\star), \text{ as } (q_1 q_2)^\star = q_2^\star\, q_1^\star$$

$$\mathcal{R}_{q_1\, q_2}(v) = q_1\ (q_2\ v\ q_2^\star)\ q_1^\star$$

$$\mathcal{R}_{q_1\, q_2}(v) = q_1\, \mathcal{R}_{q_2}(v)\, q_1^\star = \mathcal{R}_{q_1} \circ \mathcal{R}_{q_2}(v)$$

Correspondance quaternion to rotation matrix

The unit quaternion $q = (x, y, z, w)$ represents the rotation given by the matrix

$$R = \begin{pmatrix} 1-2(y^2+z^2) & 2(xy-wz) & 2(xz+wy) \\ 2(xy+wz) & 1-2(x^2+z^2) & 2(yz-wx) \\ 2(xz-wy) & 2(yz+wx) & 1-2(x^2+y^2) \end{pmatrix}$$

Demonstration

$$v' = \mathcal{R}_q(v) = q\, q_v\, q^\star = ((w^2 - s^2)\, v + 2(s \cdot v)\, s + 2w\,(s \times v), 0) \quad \text{with } s = (x, y, z)$$

$$v' = (w^2 - x^2 - y^2 - z^2)v + 2 \begin{pmatrix} x & y & z \end{pmatrix} v \begin{pmatrix} x & y & z \end{pmatrix}^T + 2w \begin{pmatrix} x & y & z \end{pmatrix} \times v$$

$$v' = \left((w^2 - x^2 - y^2 - z^2)\,\mathrm{I} + 2 \begin{pmatrix} x^2 & xy & xz \\ xy & y^2 & yz \\ xz & yz & z^2 \end{pmatrix} + 2w \begin{pmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{pmatrix} \right) v$$

$v' = R\, v$, with $x^2 + y^2 + z^2 + w^2 = 1$

Summary - Correspondance quaternion / matrix-vector

Representation and Operations

	3D space	**Quaternion space**
Vector	$v = (v_x, v_y, v_z)$	$q_v = (v, 0) = (v_x, v_y, v_z, 0)$
Rotation	R (3 $\times$ 3 matrix)	$q = (x, y, z, w)$, $\|q\| = 1$
Apply rotation to vector	Rv	$q\ q_v\ q^\star$
Rotation composition	$R_1\ R_2$	$q_1\ q_2$

Rotation to Quaternion

Rotation of axis n and angle $\theta \Rightarrow q = (n \sin(\theta/2), \cos(\theta/2))$

Quaternion to Rotation

Unit quaternion $q = (x, y, z, w) \Rightarrow R = \begin{pmatrix} 1 - 2(y^2 + z^2) & 2(xy - wz) & 2(xz + wy) \\ 2(xy + wz) & 1 - 2(x^2 + z^2) & 2(yz - wx) \\ 2(xz - wy) & 2(yz + wx) & 1 - 2(x^2 + y^2) \end{pmatrix}$

Interpolation

Rotations

Interpolating rotation

Problem: Given 2 rotations r_1, r_2

- Find rotation $r(t)$ st $r(0) = r_1, r(1) = r_2$, and varies smoothly along t

Matrix representation

Componentwise interpolation not adapted

ex. $t\,R_1 + (1-t)\,R_2$ is not a rotation: introduce scaling/shearing

The correct formulation to interpolate on the manifold would be $R1\,exp(t\log(R_1^T\,R_2))$

But matrix exponential is complex to compute

Euler angle

Can interpolate the 3 angles separately

(+) Leads to a rotation

(-) Doesn't necessarily follows simplest trajectory

Axis-Angle

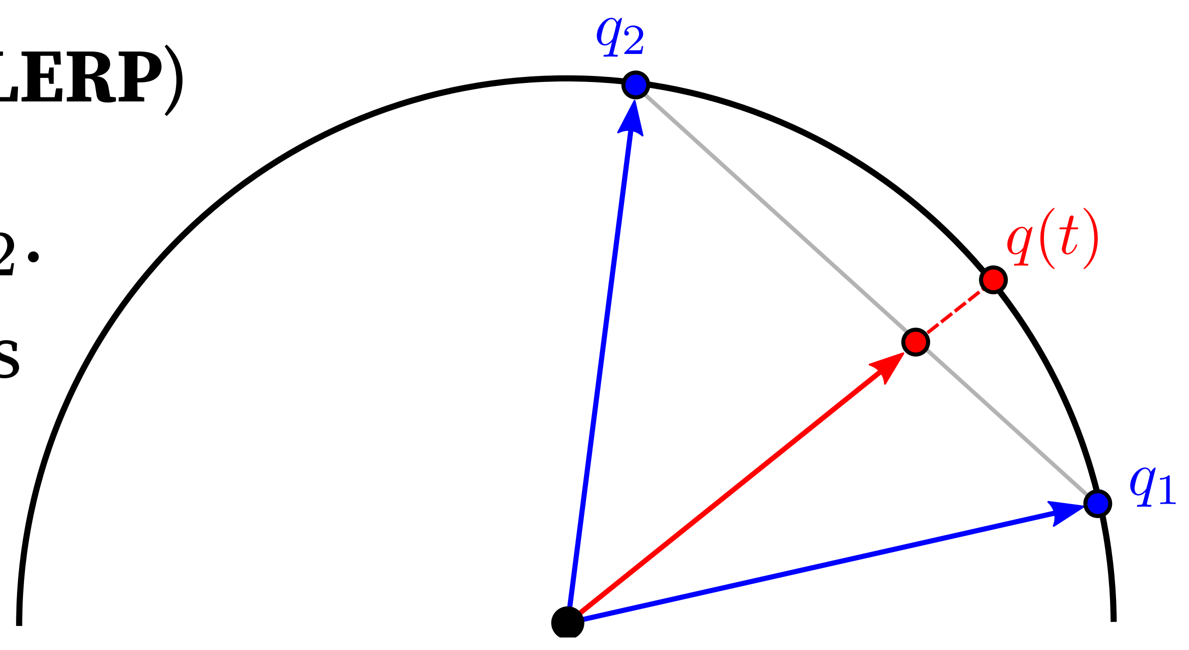
No trivial interpolation scheme with different axes

Quaternion

Well adapted (see later)

Interpolation of Quaternion - LERP

Linear interpolation (with normalization) in quaternion space (**LERP**)



- Consider two rotations with corresponding quaternion q_1, q_2.
- The *linearly interpolated* quaternion at parameter $t \in [0, 1]$ is

$$q(t) = \frac{(1-t)\, q_1 + t\, q_2}{\|(1-t)\, q_1 + t\, q_2\|}$$

(+) Follows a shortest path (great circle on 4D sphere)

(+) Can be generalized to more general parametric curves (Splines, etc)

(-) Angular speed of orientation is not-constant

ex. extreme case of two opposite quaternions

Interpolation of Quaternion - SLERP

Spherical Linear interpolation (**SLERP**)

- Consider two units vectors (arbitrary dimensions) v_1, v_2.
- The *spherical linear interpolation* at parameter $t \in [0, 1]$ is

$$v(t) = \frac{\sin((1-t)\Omega)}{\sin(\Omega)} v_1 + \frac{\sin(t\Omega)}{\sin(\Omega)} v_2 \quad \text{with } \cos(\Omega) = v_1 \cdot v_2$$

Demonstration

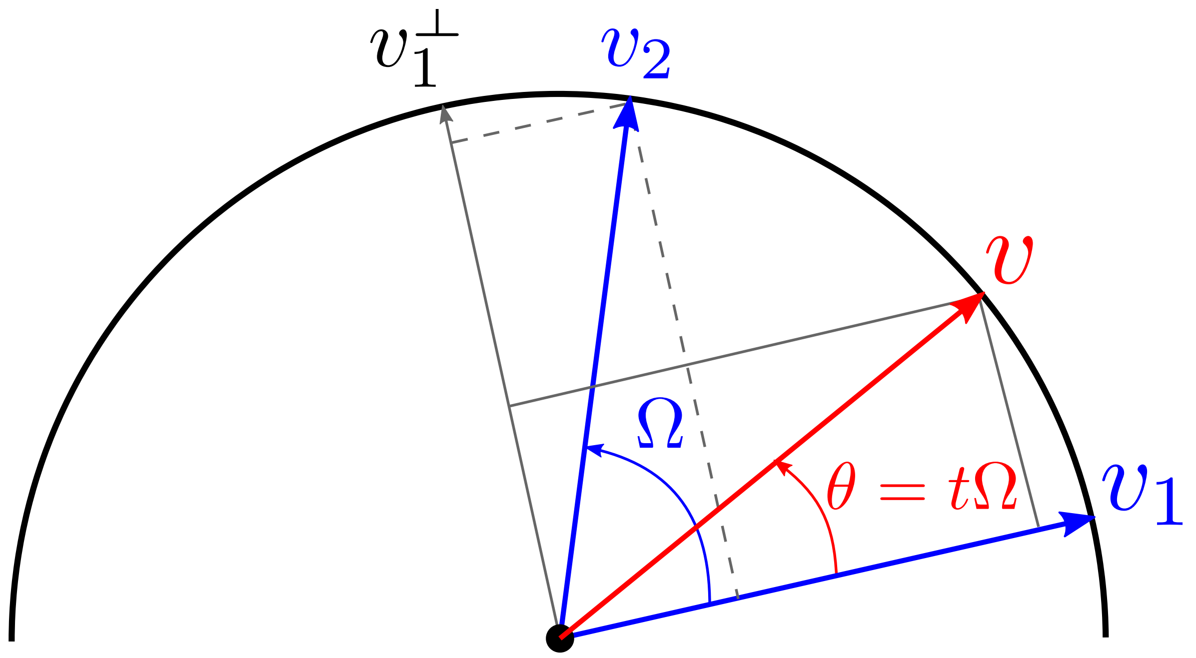
Consider

- Two unit vectors (arbitrary dimensions) v_1, v_2.
- Ω: angle b/w v_1 and v_2
- The interpolated vector v at angle $\theta = \Omega t, t \in [0, 1]$.

$v = v_1 \cos(\theta) + v_1^{\perp} \sin(\theta)$, and $v_1^{\perp} = \frac{v_2 - \cos(\Omega)\, v_1}{\sin(\Omega)}$

$\Rightarrow v = \left(\frac{\sin(\Omega)\ \cos(\theta) - \cos(\Omega)\ \sin(\theta)}{\sin(\Omega)}\right) v_1 + \frac{\sin(\theta)}{\sin(\Omega)} v_2$

$\Rightarrow v = \frac{\sin(\Omega - \theta)}{\sin(\Omega)} v_1 + \frac{\sin(\theta)}{\sin(\Omega)} v_2$

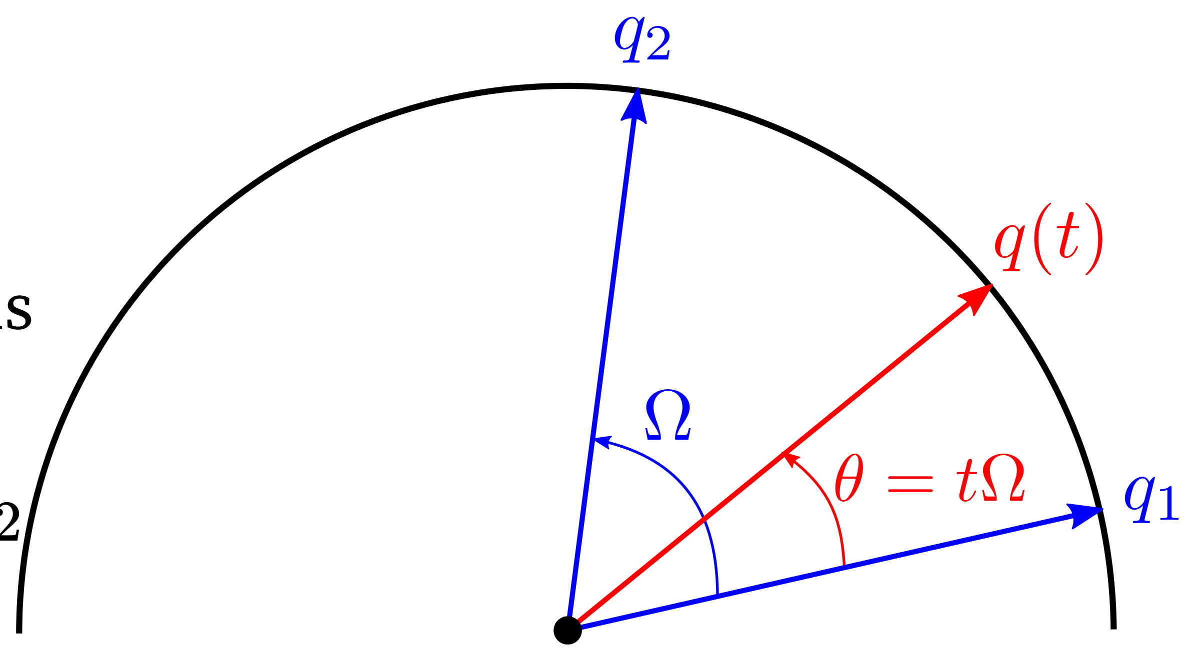


Interpolation of Quaternion - SLERP

Spherical Linear interpolation (**SLERP**)

- Consider two rotations with corresponding quaternion q_1, q_2.
- The *spherical linear interpolated* quaternion at parameter $t \in [0,1]$ is

$$q(t) = \frac{\sin((1-t)\Omega)}{\sin(\Omega)} q_1 + \frac{\sin(t\Omega)}{\sin(\Omega)} q_2 \quad \text{with } \cos(\Omega) = q_1 \cdot q_2$$



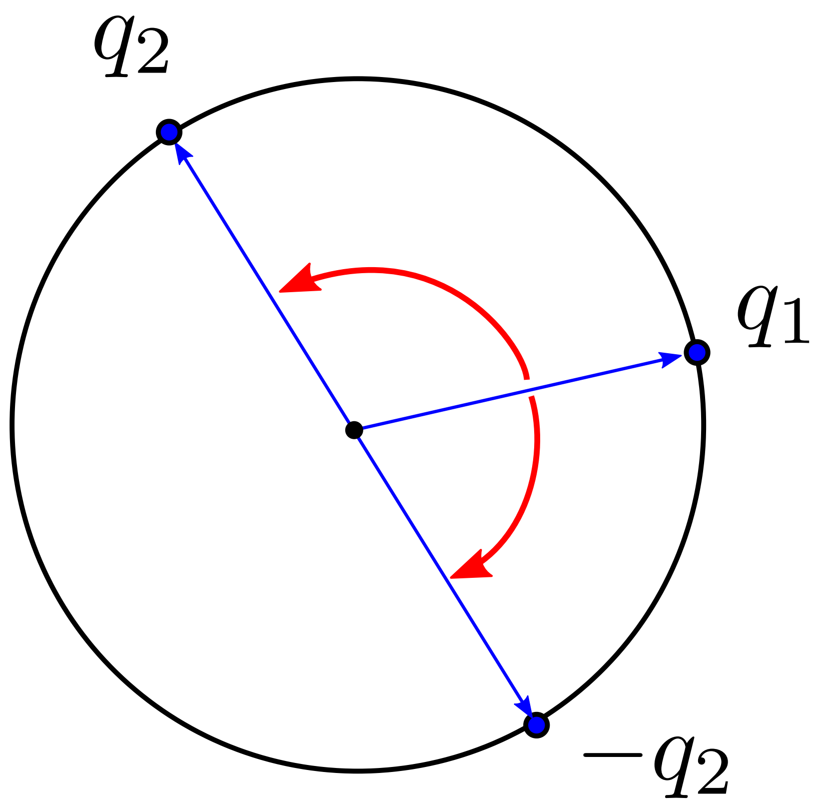
(+) Follows a shortest path (great circle on 4D sphere)

(+) Constant angular speed

(-) Cannot be generalized to more general curves.

Cannot interpolates between more than two quaternions

Care with quaternion negation



$+q$ and $-q$ correspond to the same rotation ($n \to -n, \theta \to 2\pi - \theta$)

*Warning $-q$ **does not** corresponds to the rotation matrix $-R$.*

But to a **different path** when interpolated in the 4D quaternion space.

$\Rightarrow$ path $q_1 \to -q_2$ is shorter than $q_1 \to q_2$ when $q_1 \cdot q_2 < 0$.

In practice we check for the shorter path before applying SLERP.

Algorithm

```
if( dot(q1,q2)<0 )
    q2 = -q2

q(t) = SLERP(q1,q2,t)
```

Interpolation

Affine Transforms

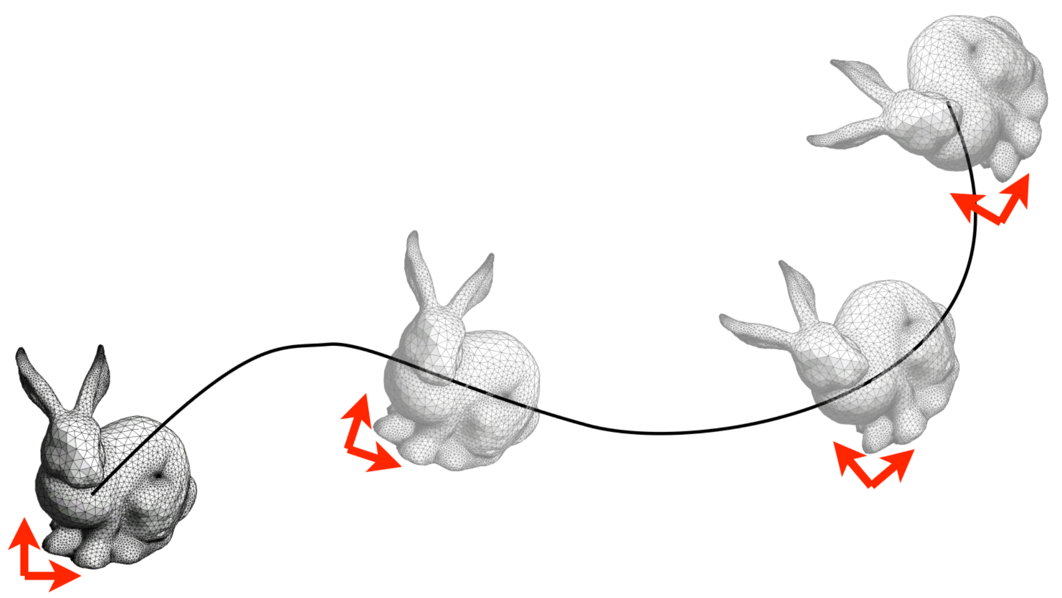
Interpolating rigid motion

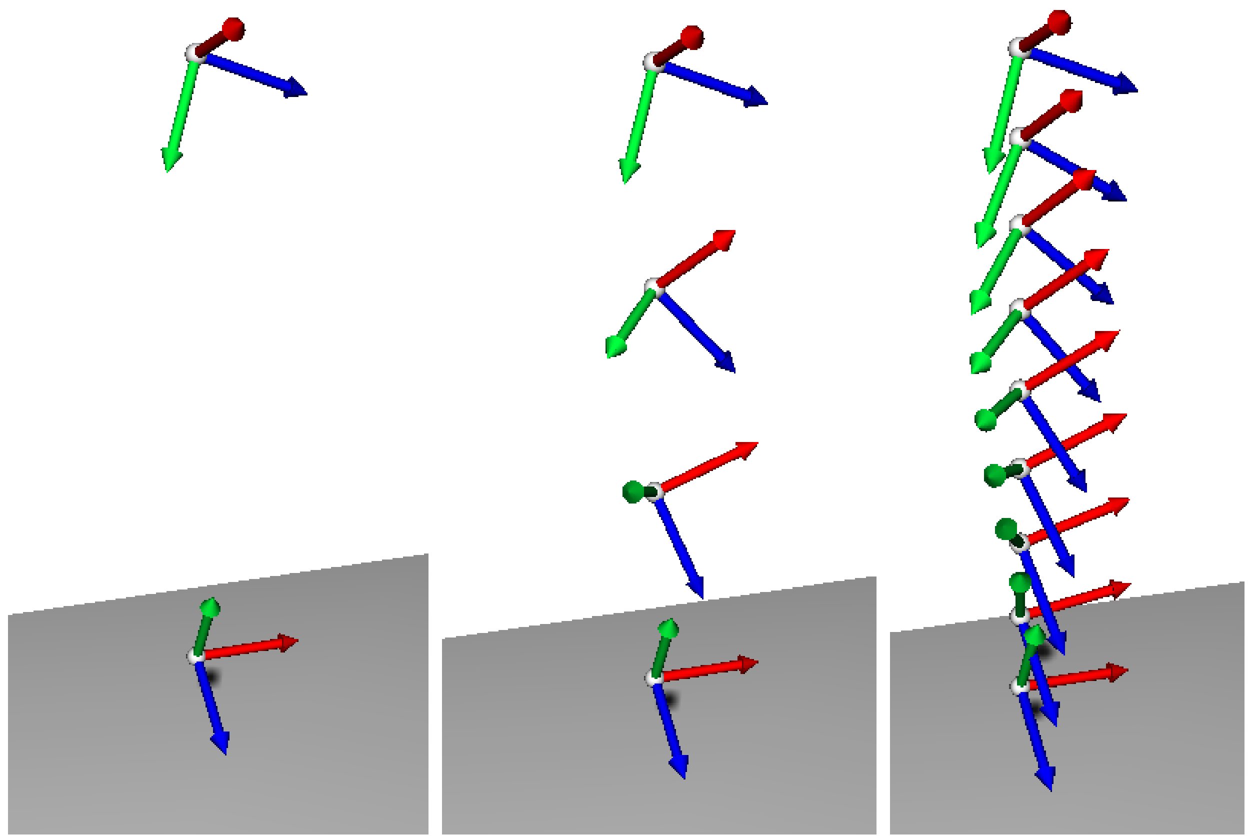
3D objects have orientation r and position p.

$\Rightarrow$ Need to interpolate both, usually handled separately.

Given two key-positions (p_1, r_1) and (p_2, r_2) *in-betweens* computed at time $t \in [0, 1]$ as

- Linear interpolation of positions $p(t) = (1-t)\,p_1 + t\,p_2$
- Interpolate rotation with SLERP on quaternions
    - Convert $(r_1, r_2) \rightarrow (q_1, q_2)$
    - Compute $q(t) = SLERP(q_1, q_2, t)$
    - Convert back $q(t) \rightarrow r(t)$





Interpolating rigid motion - Comparison


Matrix interpolation


Euler angle interpolation


Quaternion interpolation