Notes on Orientation Trajectories

Jon Woolfrey

March 2022

Trajectories Between 2 Orientations

Suppose we wish to generate a trajectory to smoothly transition the orientation of an object over time. The problem is, there are many paradigms for representing orientation. For example:

- $\mathbf{R} \in \mathbb{SO}(3)$: rotation matrices,
- $\phi, \theta, \psi \in \mathbb{R}^3$: roll, pitch, and yaw angles (i.e. Euler angles), and
- $\mathcal{Q} \in \mathbb{H}$: quaternions.

Rotation matrices are immediately ruled out as a choice of representation since they cannot be superimposed:

$$\mathbf{R} + \mathbf{R} \notin \mathbb{SO}(3).$$

Alternatively, we could extract the Euler angles from a given rotation matrix and use those as a state vector:

$$\mathbf{x}(\mathrm{t}) = \begin{bmatrix} \phi(\mathrm{t}) \\ \theta(\mathrm{t}) \\ \psi(\mathrm{t}) \end{bmatrix} \in \mathbb{R}^3.$$

Then we can simply apply a polynomial for linear interpolation:

$$\mathrm{x_i}(\mathrm{t}) = \mathrm{a_i}(\mathrm{t} - \mathrm{t_i})^{\mathrm{n}-1} + \mathrm{b_i}(\mathrm{t} - \mathrm{t_i})^{\mathrm{n}-2} + \mathrm{c_i}(\mathrm{t} - \mathrm{t_i})^{\mathrm{n}-3} + \dots \tag{1}$$

for $\mathrm{i} \in \{1, 2, 3\}$. However, there are two problems with this approach:

1. The Euler angles cannot be determined for certain orientations due to the Gimbal lock phenomenon, and
2. Euler angle interpolation does not take the shortest route between two orientations.

Alternatively, we could use spherical linear interpolation (slerp) over quaternions [Shoemake, 1985]:

$$\mathcal{Q}(\mathrm{t}) = \frac{\sin\left((1-\mathrm{s}(\mathrm{t}))\,\alpha\right)}{\sin(\alpha)}\mathcal{Q}_0 + \frac{\sin\left(\mathrm{s}(\mathrm{t})\alpha\right)}{\sin(\alpha)}\mathcal{Q}_\mathrm{f}$$

where $\mathrm{s}(\mathrm{t}_0) = 0$, $\mathrm{s}(\mathrm{t}_\mathrm{f}) = 1$, and $\alpha = [0, 2\pi]$ is the angle between the orientations. This method is nice because quaternions are computationally efficient and slerp will generate the shortest path. Slerp was designed with animation in mind, hence its time derivatives are not explored. However, to adequately apply feedback control, we require the angular velocity and acceleration vectors $\boldsymbol{\omega}, \dot{\boldsymbol{\omega}} \in \mathbb{R}^3$. [Wittenburg, 2008] provides a proof of how $\dot{\mathcal{Q}}$ relates to $\boldsymbol{\omega}$, and with some effort the acceleration level relationship can be derived. The problem becomes complicated when we want to generate a trajectory across multiple waypoints via a spline.

Instead, given an initial orientation $\mathbf{R}_0$ and final orientation $\mathbf{R}_\mathrm{f}$ (or $\mathcal{Q}_0$ and $\mathcal{Q}_\mathrm{f}$) we can interpolate over the *difference* between them:

$$\mathbf{R}(\mathrm{t}) = \mathbf{R}_0 \Delta\mathbf{R}(\mathrm{t})$$

such that:

$$\mathbf{R}(\mathrm{t}_0) = \mathbf{R}_0 \Longrightarrow \Delta\mathbf{R}(\mathrm{t}_0) = \mathbf{I}$$
$$\mathbf{R}(\mathrm{t}_\mathrm{f}) = \mathbf{R}_\mathrm{f} \Longrightarrow \Delta\mathbf{R}(\mathrm{t}_\mathrm{f}) = \mathbf{R}_0^{-1}\mathbf{R}_\mathrm{f}.$$

Moreover, we can represent this difference using the angle-axis notation:

$$\Delta\mathbf{R}(\mathrm{t}) \rightarrow \Delta\alpha(\mathrm{t}),\ \hat{\mathbf{a}} \in \mathbb{R}^3$$

and assign our state vector as:

$$\mathbf{x}(\mathrm{t}) = \Delta\alpha(\mathrm{t})\hat{\mathbf{a}} \in \mathbb{R}^3.$$

We can then apply Eqn. (1) as our interpolating polynomial using the conditions:

$$\mathbf{x}(\mathrm{t}_0) = \mathbf{0} \qquad \mathbf{x}(\mathrm{t}_\mathrm{f}) = \Delta\alpha(\mathrm{t}_\mathrm{f})\hat{\mathbf{a}}.$$

By using the angle-axis representation we take the shortest route between orientations. In addition, the angular velocity and acceleration naturally follows:

$$\boldsymbol{\omega}(\mathrm{t}) = \dot{\mathbf{x}}(\mathrm{t}) \qquad \dot{\boldsymbol{\omega}}(\mathrm{t}) = \ddot{\mathbf{x}}(\mathrm{t}).$$

We can always recover the angle and axis of rotation from the state vector by computing its norm:

$$\Delta\alpha(\mathrm{t}) = \|\mathbf{x}(\mathrm{t})\| \qquad \hat{\mathbf{a}} = \frac{\mathbf{x}(\mathrm{t})}{\|\mathbf{x}(\mathrm{t})\|}.$$

We can then convert this back to $\mathbb{SO}(3)$ using Rodrigues' rotation formula, or to $\mathbb{H}$.

Trajectories Between Many Orientations

Suppose we have several orientations we wish to interpolate over: $\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_n$. We can apply a single 3rd order polynomial between any two points j and j + 1:

$$x_{i,j}(t) = a_{i,j}(t - t_{i,j})^3 + b_{i,j}(t - t_{i,j})^2 + c_{i,j}(t - t_{i,j}) + d_{i,j} \tag{2}$$

for $i \in \{1, 2, 3\}$ and $j \in \{1, \ldots, n-1\}$. This process of joining together individual polynomials is known as a spline. The solutions for the cubic spline coefficients are given by

$$a_{i,j} = -\frac{2\Delta x_{i,j}}{\Delta t_j^3} + \frac{\dot{x}_{i,j+1} - \dot{x}_{i,j}}{\Delta t_j^2} \tag{3a}$$

$$b_{i,j} = \frac{3\Delta x_{i,j}}{\Delta t_j^2} - \frac{\dot{x}_{i,j+1} + 2\dot{x}_{i,j}}{\Delta t_j} \tag{3b}$$

$$c_{i,j} = \dot{x}_{i,j} \tag{3c}$$

$$d_{i,j} = 0. \tag{3d}$$

which is determined entirely by the displacement and velocities at the start and end. The difference in displacement is conveniently available to us from the choice of our state representation:

$$\Delta\mathbf{x}_j = \Delta\alpha_j(t_{j+1})\hat{\mathbf{a}} = \begin{bmatrix} \Delta x_{1,j} \\ \Delta x_{2,j} \\ \Delta x_{3,j} \end{bmatrix}$$

Since displacement and time are fixed, we must solve for the velocities at each waypoint. To do this, we need to eliminate the coefficients from Eqns. (3a) to (3c). This can be achieved by equating the accelerations:

$$\ddot{x}_{i,j-1}(t_j) = \ddot{x}_{i,j}(t_j). \tag{4a}$$

This ensures a smooth transition between sections of the spline at the velocity level. It follows from Eqns. (2) and (4a) that:

$$3a_{i,j-1}\Delta t_{j-1} + b_{i,j-1} = b_{i,j}. \tag{4b}$$

Then by substituting in Eqn. (3b) this leads to:

This can be put in to matrix/vector form as:

$$\begin{bmatrix} \Delta t_{j-1}^{-1} & 2\left(\Delta t_{j-1}^{-1} + \Delta t_j^{-1}\right) & \Delta t_j^{-1} \end{bmatrix} \begin{bmatrix} \dot{x}_{i,j-1} \\ \dot{x}_{i,j} \\ \dot{x}_{i,j+1} \end{bmatrix} = \begin{bmatrix} 3\Delta t_{j-1}^{-2} & 3\Delta t_j^{-2} \end{bmatrix} \begin{bmatrix} \Delta x_{i,j-1} \\ \Delta x_{i,j} \end{bmatrix}. \tag{4c}$$

Now the issue is that we have 1 equation but 3 unknowns (which extends to n unknowns and n − 2 equations for the full spline). A practical solution is to set the start and and velocities

as zero:

$$\begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ & & \ddots & & \\ 0 & 0 & \cdots & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{x}_{i,1} \\ \dot{x}_{i,2} \\ \vdots \\ \dot{x}_{i,n-1} \\ \dot{x}_{i,n} \end{bmatrix} = \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 \\ & & \ddots & & \\ 0 & 0 & \cdots & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta x_{i,1} \\ \Delta x_{i,2} \\ \vdots \\ \Delta x_{i,n-2} \\ \Delta x_{i,n-1} \end{bmatrix}. \tag{5}$$

In this manner, any object following the trajectory will start and end from rest.

By combining Eqns. (4c) and (5) we can solve a system of equations as follows:

$$\mathbf{A} \begin{bmatrix} \dot{x}_{i,1} \\ \vdots \\ \dot{x}_{i,n} \end{bmatrix} = \mathbf{B} \begin{bmatrix} \Delta x_{i,1} \\ \vdots \\ \Delta x_{i,n-1} \end{bmatrix} \tag{6a}$$

$$\begin{bmatrix} \dot{x}_{i,1} \\ \vdots \\ \dot{x}_{i,n} \end{bmatrix} = \mathbf{A}^{-1}\mathbf{B} \begin{bmatrix} \Delta x_{i,1} \\ \vdots \\ \Delta x_{i,n-1} \end{bmatrix}. \tag{6b}$$

References

[Shoemake, 1985] Shoemake, K. (1985). Animating rotation with quaternion curves. In *12th Annual Conference on Computer Graphics and Interactive Techniques*, pages 245–254.

[Wittenburg, 2008] Wittenburg, J. (2008). *Dynamics of Multibody Systems*. Springer. (c) Springer-Verlag Berlin Heidelberg 2008.