Geometry for computer Vision

Lecture 7b: Rotation interpolation and smoothing

Lecture 7b: Rotation Interpolation and smoothing

- Interpolation of SO(3)
- Smoothing of SO(3)
- SO(3) and SE(3)
- Discussion of SLERP article

Motivation

- Computer Graphics Animations
- After SfM you might want a smoother camera trajectory.
- Video stabilisation.
- Augmented reality.

SO(3)

- SO(3) is the group of 3D rotations (3dof)

$$\mathrm{SO}(3) = \left\{ \mathbf{R} \in \mathbb{R}^{3\times 3} | \mathbf{R}^T\mathbf{R} = \mathbf{I}, \det(\mathbf{R}) = 1 \right\}$$

SO(3)

- SO(3) is the group of 3D rotations (3dof)

$$\mathrm{SO}(3) = \left\{\mathbf{R} \in \mathbb{R}^{3\times 3} | \mathbf{R}^T\mathbf{R} = \mathbf{I}, \det(\mathbf{R}) = 1\right\}$$

- An element in SO(3) can be represented by three elements from the matrix logarithm of $\mathbf{R}$

$$\mathrm{logm}(\mathbf{R}) = \begin{bmatrix} 0 & -n_3 & n_2 \\ n_3 & 0 & -n_1 \\ -n_2 & n_1 & 0 \end{bmatrix}$$

- Or by the 4-elements in a unit quaternion

$$\mathbf{q} = (\cos\frac{\theta}{2}, \sin\frac{\theta}{2}\hat{\mathbf{n}})$$

SO(3)

- SO(3) is the group of 3D rotations (3dof)

$$\mathrm{SO}(3) = \left\{\mathbf{R} \in \mathbb{R}^{3\times 3} | \mathbf{R}^T\mathbf{R} = \mathbf{I}, \det(\mathbf{R}) = 1\right\}$$

- An element in SO(3) can be represented by three elements from the matrix logarithm of **R**

$$\mathrm{logm}(\mathbf{R}) = \begin{bmatrix} 0 & -n_3 & n_2 \\ n_3 & 0 & -n_1 \\ -n_2 & n_1 & 0 \end{bmatrix} \quad \in \mathrm{so}(3)$$

- Or by the 4-elements in a unit quaternion

$$\mathbf{q} = (\cos\frac{\theta}{2}, \sin\frac{\theta}{2}\hat{\mathbf{n}}) \quad \in \mathrm{SU}(2)$$

SLERP

- SLERP (see today's paper) dictates that we should interpolate two rotations by applying parts of the intermediate rotation, followed by the first rotation

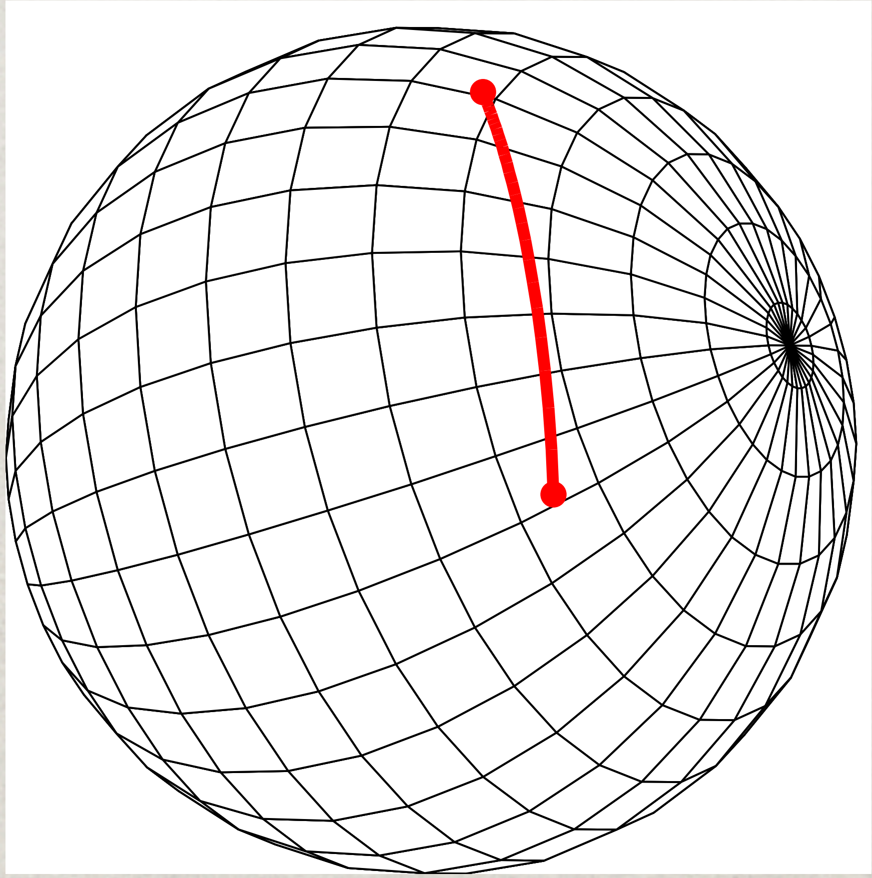
$$\mathtt{SLERP}(\mathbf{q}_1, \mathbf{q}_2, w) = \mathbf{q}_1(\mathbf{q}_1^{-1}\mathbf{q}_2)^w$$

- Or if we use rotation matrices

$$\mathtt{SLERP}(\mathbf{R}_1, \mathbf{R}_2, w) = \mathbf{R}_1\mathtt{exp}(w\mathtt{log}(\mathbf{R}_1^T\mathbf{R}_2))$$

SLERP

- The SLERP construction is a ***geodesic*** on SO(3), i.e. a walk along the shortest path, on the manifold, between the two rotations.

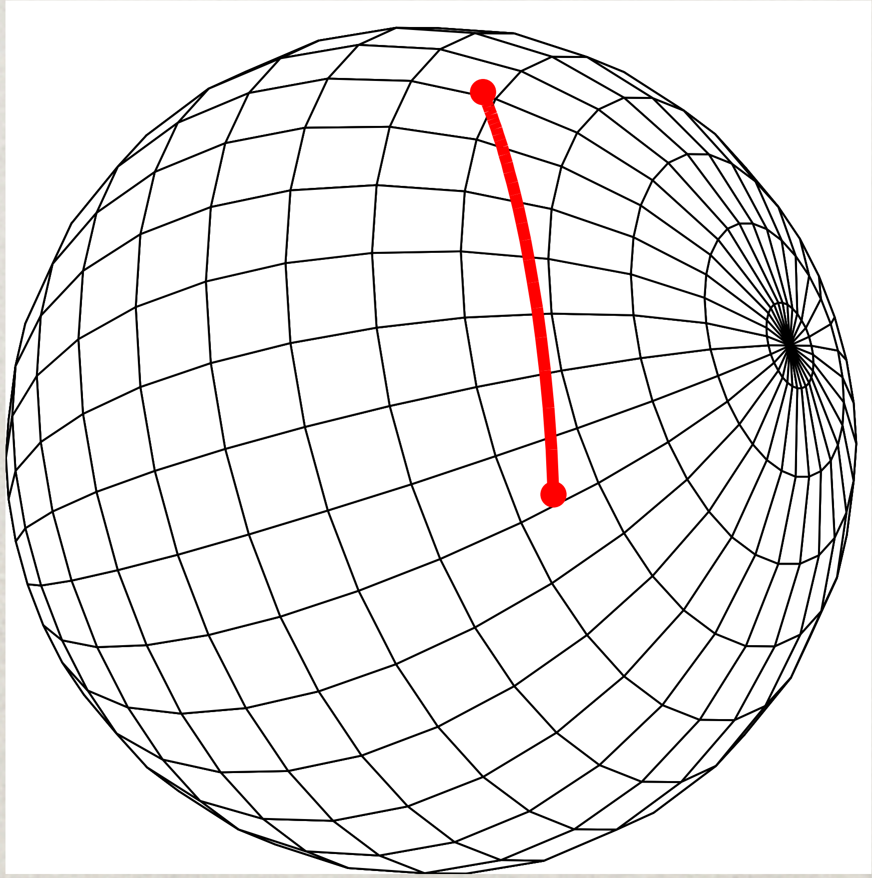


Geodesic on the sphere

SLERP

- The SLERP construction is a ***geodesic*** on SO(3), i.e. a walk along the shortest path, on the manifold, between the two rotations.

- If we use unit quaternions, the geodesic lies on a 4D sphere.



Geodesic on the sphere

Interpolation on SO(3)

- We can interpolate between key rotations on SO(3) using Bézier curves as in today's paper.

- Another alternative is to define cubic splines directly on the rotation group as described in: **Park and Ravani, *Smooth Invariant Interpolation of Rotations*, ACM Transactions on Graphics 1997.**

Interpolation on SO(3)

- A ***natural cubic spline*** on $\mathbb{R}^n$ has the form

$$\mathbf{y}(t) = \mathbf{a}_i\tau^3 + \mathbf{b}_i\tau^2 + \mathbf{c}_i\tau + \mathbf{d}_i\,, \quad \tau = \frac{t - t_i}{t_{i+1} - t_i}$$

Interpolation on SO(3)

- A ***natural cubic spline*** on $\mathbb{R}^n$ has the form

$$\mathbf{y}(t) = \mathbf{a}_i\tau^3 + \mathbf{b}_i\tau^2 + \mathbf{c}_i\tau + \mathbf{d}_i\,, \quad \tau = \frac{t - t_i}{t_{i+1} - t_i}$$

- On SO(3) we instead get the expression

$$\mathbf{R}(t) = \mathbf{R}_{i-1}e^{[\mathbf{a}_i\tau^3 + \mathbf{b}_i\tau^2 + \mathbf{c}_i\tau]_\times}\,, \quad \tau = \frac{t - t_i}{t_{i+1} - t_i}$$

- **b** corresponds to angular acceleration, and **c** is angular the velocity.

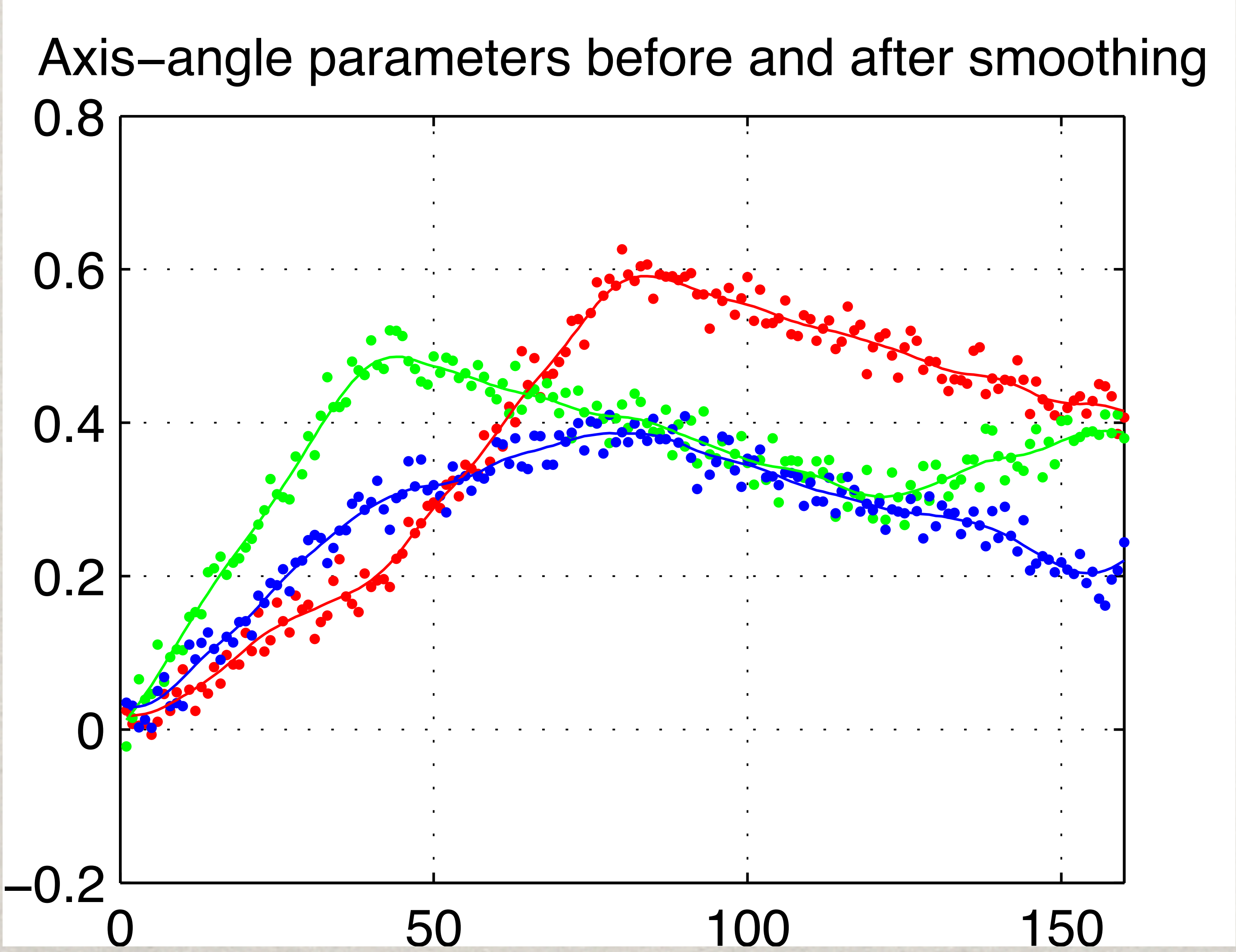
- Initialise $\mathbf{b}_0$ and $\mathbf{c}_0$ by setting them to 0

Interpolation on SO(3)

- $\mathbf{a}_i, \mathbf{b}_i, \mathbf{c}_i$ can be computed recursively, from the previous values: $\mathbf{a}_{i-1}, \mathbf{b}_{i-1}, \mathbf{c}_{i-1}$
- Park and Ravani's scheme is more efficient than the Bézier curves of Shoemake's
- The Spline approximately minimises integrated angular acceleration of the curve.

ROTATION SMOOTHING

- Problem: We have a sequence of noisy rotations, and want a smoother trajectory.



ROTATION SMOOTHING

- For each temporal window, this can be solved by ML as:

$$\mathbf{R}^* = \arg\min_{\mathbf{R}\in SO(3)} \sum_k d_{\text{geo}}(\mathbf{R}, \mathbf{R}_k)^2$$

- Where

$$d_{\text{geo}}(\mathbf{R}_1, \mathbf{R}_2)^2 = \frac{1}{2}||\text{logm}(\mathbf{R}_1^T\mathbf{R}_2)||_{\text{fro}}^2$$

ROTATION SMOOTHING

- For each temporal window, this can be solved by ML as:

$$\mathbf{R}^* = \arg \min_{\mathbf{R} \in SO(3)} \sum_k d_{\text{geo}}(\mathbf{R}, \mathbf{R}_k)^2$$

- Where

$$d_{\text{geo}}(\mathbf{R}_1, \mathbf{R}_2)^2 = \frac{1}{2} ||\text{logm}(\mathbf{R}_1^T \mathbf{R}_2)||_{\text{fro}}^2$$

- Iterative search. Maybe too slow :-(

- There are fast and nearly as good alternatives :-)

ROTATION SMOOTHING

- For a sequence of **unit quaternions**

  $\mathbf{q}_k \ , \quad \mathbf{q}_{k+1} \ , \quad \mathbf{q}_{k+2} \ , \quad \cdots$

  $$\mathbf{q}_k = (\cos\frac{\theta_k}{2}, \sin\frac{\theta_k}{2}\hat{\mathbf{n}}_k)$$

- Note that $\mathbf{q}_k$ and $-\mathbf{q}_k$ represent the same rotation (double folding property)

- We need to first ensure that $\mathbf{q}_k \cdot \mathbf{q}_l > 0$

- Now we can simply average them!

ROTATION SMOOTHING

- If we have a sequence of **unit quaternions**

$$\mathbf{q}_k \ , \quad \mathbf{q}_{k+1} \ , \quad \mathbf{q}_{k+2} \ , \quad \cdots$$

$$\mathbf{q}_k = (\cos\frac{\theta_k}{2}, \sin\frac{\theta_k}{2}\hat{\mathbf{n}}_k)$$

- Apply a temporal convolution, followed by a normalisation to unit length.

$$\tilde{\mathbf{q}}_k = \sum_{l=-2}^{2} w_l \mathbf{q}_{k+l} \ , \qquad \hat{\mathbf{q}}_k = \tilde{\mathbf{q}}_k / \sqrt{\tilde{q}_1^2 + \tilde{q}_2^2 + \tilde{q}_3^2 + \tilde{q}_4^2}$$

ROTATION SMOOTHING

- If we have a sequence of **rotation matrices**

$$\mathbf{R}_k\,,\quad \mathbf{R}_{k+1}\,,\quad \mathbf{R}_{k+2}\,,\quad \ldots$$

We could apply a temporal convolution, followed by an orthogonalisation.

$$\tilde{\mathbf{R}}_k = \sum_{l=-2}^{2} w_l \mathbf{R}_{k+l}$$

$$\mathbf{UDV}^T = \texttt{svd}(\tilde{\mathbf{R}}_k)\,,\quad \hat{\mathbf{R}}_k = \mathbf{UV}^T$$

ROTATION SMOOTHING

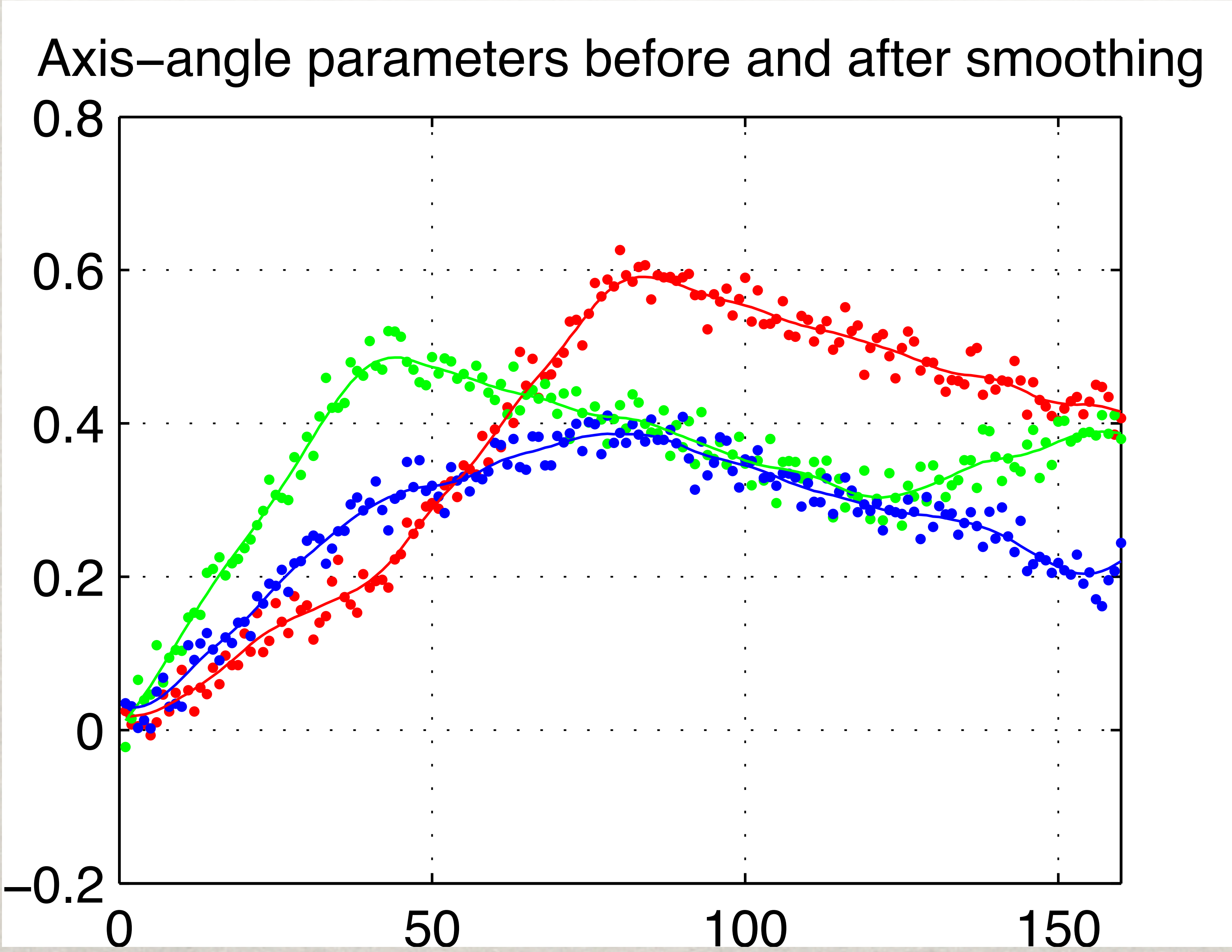
- Both versions can be shown to be 2nd order Taylor approximations of the geodesic distance. **Gramkow, *On Averaging Rotations*, IJCV01**

- Gramkow also compares both against ML. Both are **very accurate** (<5% relative error at 40deg)

- Quaternion variant is slightly closer to the ML solution, and also significantly faster.

ROTATION SMOOTHING

- Result (both methods indistinguishable)



SO(3) and SE(3)

- SO(3) is the group of 3D rotations (3dof)

$$\mathrm{SO}(3) = \left\{\mathbf{R} \in \mathbb{R}^{3\times 3} | \mathbf{R}^T\mathbf{R} = \mathbf{I}, \det(\mathbf{R}) = 1\right\}$$

- SE(3) is the group of Euclidean rigid body transformations (3D rotation+3Dtranslation) (6dof)

$$\mathrm{SE}(3) = \mathrm{SO}(3) \times \mathbb{R}^3$$

- For SE(3) we can similarly define an exponential map and a log map.

SO(3) and SE(3)

- An element $\mathbf{G} \in \mathrm{SE}(3)$ has the matrix form

$$\mathbf{G} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \qquad \mathbf{R} \in \mathrm{SO}(3)\,, \quad \mathbf{t} \in \mathbb{R}^3$$

- It is the exponential of a **twist**

$$\mathbf{G} = \mathtt{exp}(\hat{\xi}\theta) \qquad \hat{\xi} = \begin{bmatrix} \mathrm{logm}(\mathbf{R}) & \mathbf{v} \\ 0 & 0 \end{bmatrix} \qquad \theta \in \mathbb{R}$$

SO(3) and SE(3)

- An element $\mathbf{G} \in \mathrm{SE}(3)$ has the matrix form

$$\mathbf{G} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \qquad \mathbf{R} \in \mathrm{SO}(3)\,, \quad \mathbf{t} \in \mathbb{R}^3$$

- It is the exponential of a **twist**

$$\mathbf{G} = \mathtt{exp}(\hat{\xi}\theta) \qquad \hat{\xi} = \begin{bmatrix} \mathrm{logm}(\mathbf{R}) & \mathbf{v} \\ 0 & 0 \end{bmatrix} \qquad \theta \in \mathbb{R}$$

- One could do smoothing and interpolation of rigid body motions using the geodesic distance on SE(3) (via the log map). **However...**

SO(3) and SE(3)

- It turns out that **physically meaningful** motions do not follow geodesics in SE(3). Rather (if no external force):

  1. The centre of mass moves linearly

  2. Rotation happens about the centre of mass

- Thus we should represent $\mathbf{R}(t)$ in object centered coordinates, and interpolate $\mathbf{R}(t)$ and $\mathbf{t}(t)$ separately.

SO(3) and SE(3)

- A very good treatment of SO(3) and SE(3) can be found in the book:
  Murray et al. ***A Mathematical Introduction to Robotic Manipulation*****, CRC Press. 1994**

- http://www.cds.caltech.edu/~murray/mlswiki/

Discussion

- Discussion of the paper:
Ken Shoemake, *Animating rotation with quaternion curves*, ACM SIGGRAPH'85

FOR NEXT WEEK...

- Forssén and Ringaby, *Rectifying rolling shutter video from hand-held devices*, CVPR'10