# Efficient Derivative Computation for Cumulative B-Splines on Lie Groups

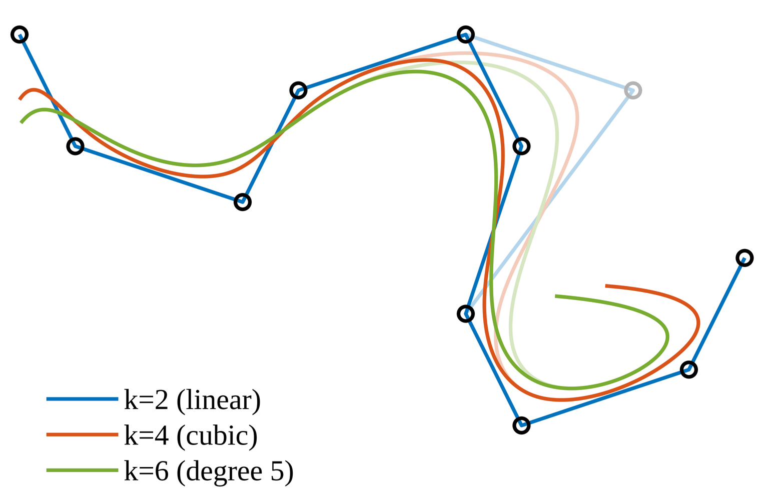
Christiane Sommer*, Vladyslav Usenko*, David Schubert, Nikolaus Demmel and Daniel Cremers

Chair for Computer Vision & Artificial Intelligence, Technical University of Munich

## Background

Continuous-time trajectory representation using B-splines is very useful for several tasks:

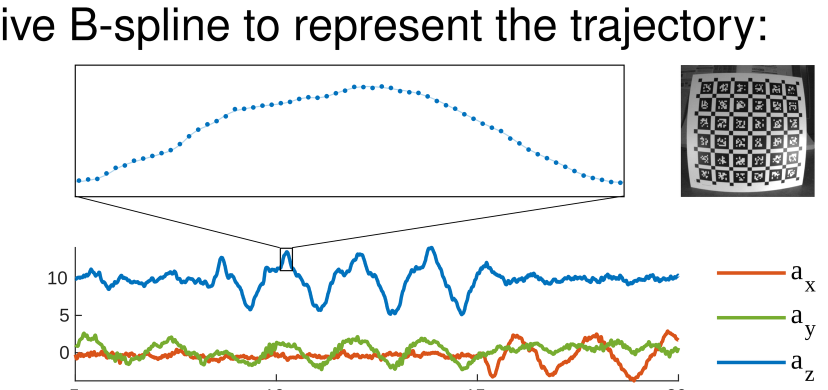
- High-frame-rate sensor calibration
- Fusion of multiple unsynchronized devices
- Smooth trajectory planning




However, current implementations for calibration [1] or odometry estimation [2, 3] are unable to achieve real-time performance.

## Example Application

Camera-IMU calibration using a Lie group cumulative B-spline to represent the trajectory:




Observations of the calibration pattern are combined with accelerometer and gyroscope measurements to estimate the trajectory and calibration parameters jointly. Accelerometer measurements (dots) are overlaid on the continuous estimate generated from the spline trajectory (line) after optimization.

## Acknowledgements

This work was supported by the ERC Consolidator Grant “3D Reloaded”.

## Contributions

In short, our work provides time derivatives and Jacobians for Lie group-values B-splines that can be more efficiently implemented than previous derivatives. In particular, it features

- A simple formulation for the time derivatives of Lie group cumulative B-splines that requires a number of matrix operation which scales linearly with the order $k$ of the spline.
- Simple (linear in $k$) analytic Jacobians of the value and the time derivatives of an $SO(3)$ spline with respect to its knots.
- Faster optimization time compared to the currently available implementations, due to provably lower complexity.

## Time Derivatives (Velocities)

The time derivative $\dot{X}$ is given by the following recurrence relation:

$$\dot{X} = X\boldsymbol{\omega}^{(k)}_\wedge , \tag{7}$$

$$\boldsymbol{\omega}^{(j)} = \mathrm{Adj}_{A_{j-1}^{-1}}\boldsymbol{\omega}^{(j-1)} + \dot{\lambda}_{j-1}\mathbf{d}_{j-1} \in \mathbb{R}^d , \tag{8}$$

$$\boldsymbol{\omega}^{(1)} = \mathbf{0} \in \mathbb{R}^d . \tag{9}$$

$\boldsymbol{\omega}^{(k)}$ is commonly referred to as *velocity*. For $\mathcal{L} = SO(n)$, we also call it *angular velocity*.

## Cumulative B-Spline on Lie Groups

The cumulative B-spline of order $k$ in a Lie group $\mathcal{L}$ with control points $X_0, \cdots, X_N \in \mathcal{L}$ has the form

$$X(u) = X_i \cdot \prod_{j=1}^{k-1} \mathrm{Exp}\left(\lambda_j(u) \cdot \mathbf{d}_j^i\right) , \tag{1}$$

with the generalized difference vector $\mathbf{d}_j^i$

$$\mathbf{d}_j^i = \mathrm{Log}\left(X_{i+j-1}^{-1} X_{i+j}\right) \in \mathbb{R}^d , \tag{2}$$

and $\lambda_j(u)$ implicitly defined by the derivation of B-splines [4, 5, 6]. We define

$$A_j(u) = \mathrm{Exp}\left(\lambda_j(u) \cdot \mathbf{d}_j^i\right) \tag{3}$$

and re-formulate (1) as a recurrence relation:

$$X(u) = X^{(k)}(u) , \tag{4}$$

$$X^{(j)}(u) = X^{(j-1)}(u) A_{j-1}(u) , \tag{5}$$

$$X^{(1)}(u) = X_i . \tag{6}$$

## Second Time Derivatives (Accelerations)

The second derivative of $X$ w.r.t. $u$ can be computed by the following recurrence relation:

$$\ddot{X} = X\left((\boldsymbol{\omega}^{(k)})^2_\wedge + \dot{\boldsymbol{\omega}}^{(k)}_\wedge\right) , \tag{10}$$

where the *(angular) acceleration* $\dot{\boldsymbol{\omega}}^{(k)}$ is recursively defined by

$$\dot{\boldsymbol{\omega}}^{(j)} = \dot{\lambda}_{j-1}\left[\boldsymbol{\omega}^{(j)}_\wedge, D_{j-1}\right]_\vee + \mathrm{Adj}_{A_{j-1}^{-1}}\dot{\boldsymbol{\omega}}^{(j-1)} + \ddot{\lambda}_{j-1}\mathbf{d}_{j-1} , \tag{11}$$

$$\dot{\boldsymbol{\omega}}^{(1)} = \mathbf{0} \in \mathbb{R}^d . \tag{12}$$

## References

[1] S. Lovegrove, A. Patron-Perez, and G. Sibley, “Spline Fusion: A continuous-time representation for visual-inertial fusion with application to rolling shutter cameras.,” in *Proc. British Mach. Vis. Conf.*, p. 93.1–93.12, 2013.

[2] C. Kerl, J. Stückler, and D. Cremers, “Dense continuous-time tracking and mapping with rolling shutter RGB-D cameras,” in *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 2264–2272, Dec 2015.

[3] E. Mueggler, G. Gallego, H. Rebecq, and D. Scaramuzza, “Continuous-time visual-inertial odometry for event cameras,” *IEEE Transactions on Robotics*, vol. 34, pp. 1425–1440, Dec 2018.

[4] M. G. Cox, “The numerical evaluation of B-splines,” *IMA Journal of Applied Mathematics*, 1972.

[5] C. De Boor, “On calculating with B-splines,” *Journal of Approximation theory*, 1972.

[6] K. Qin, “General matrix representations for B-splines,” *The Visual Computer*, vol. 16, no. 3-4, pp. 177–186, 2000.

[7] S. Agarwal, K. Mierle, and Others, “Ceres solver.” `http://ceres-solver.org`.

## Results

- Simulated velocity (*vel.*) and acceleration (*acc.*) measurements are used to estimate the trajectory
- Optimizations are done in Ceres [7]
- Baseline and our derivatives are implemented in the very same framework for maximal fairness
- In all experiments both formulations converged to the same result with the same number of iterations

Optimization time in seconds for $\mathcal{L} = SO(3)$:

| $k$ | Config. | Ours | Baseline | Speedup |
|---|---|---|---|---|
| 4 | acc. | **0.057** | 0.147 | 2.57x |
| 4 | vel. | **0.058** | 0.088 | 1.52x |
| 5 | acc. | **0.081** | 0.280 | 3.45x |
| 5 | vel. | **0.082** | 0.141 | 1.73x |
| 6 | acc. | **0.117** | 0.520 | 4.43x |
| 6 | vel. | **0.111** | 0.217 | 1.95x |

Optimization time in seconds for $\mathcal{L} = SE(3)$:

| $k$ | Config. | Ours | Baseline | Speedup |
|---|---|---|---|---|
| 4 | acc. | **0.277** | 0.587 | 2.12x |
| 4 | vel. | **0.253** | 0.334 | 1.32x |
| 5 | acc. | **0.445** | 1.196 | 2.69x |
| 5 | vel. | **0.405** | 0.581 | 1.43x |
| 6 | acc. | **0.644** | 2.332 | 3.62x |
| 6 | vel. | **0.590** | 0.936 | 1.59x |

## Code

Experiments are available open-source at: `https://gitlab.com/tum-vision/lie-spline-experiments`

## Contact

- Christiane Sommer: `sommerc@in.tum.de`
- Vladyslav Usenko: `vlad.usenko@tum.de`