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 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 \( X \) is given by the following recurrence relation:
\[
X = X \omega^{(1)}, \quad \omega^{(1)} = \lambda_{j} \omega_{j}^{(1)} + \dot{\lambda}_{j} \mathbf{d}_{j-1} \in \mathbb{R}^{3},
\]
where \( \omega^{(1)} \) is commonly referred to as velocity. For \( L = SO(3) \), we also call it angular velocity.

Second Time Derivatives (Accelerations)
The second derivative of \( X \) w.r.t. \( u \) can be computed by the following recurrence relation:
\[
X = X \omega^{(1)} + \omega^{(2)},
\]
where the (angular) acceleration \( \omega^{(2)} \) is recursively defined by
\[
\omega^{(2)} = \lambda_{j} \omega_{j}^{(2)} + \dot{\lambda}_{j} \mathbf{d}_{j-1} + \mathbf{A}_{j} \omega_{j}^{(1)} + \mathbf{d}_{j-1}/2
\]
with the generalized difference vector \( \mathbf{d}_{j} \) and \( \lambda_{j} \) implicitly defined by the derivation of B-splines [4, 5, 6].

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.

In all experiments both formulations converged to the same result with the same number of iterations.

Cumulative B-Spline on Lie Groups
The cumulative B-spline of order \( k \) in a Lie group \( L \) with control points \( X_0, \ldots, X_N \in L \) has the form
\[
X(u) = X_0 + \sum_{j=1}^{k} \Delta \mathbf{A}(u) \mathbf{B}(u) \mathbf{d}_{j-1} \in L,
\]
with the generalized difference vector \( \mathbf{d}_{j} \) and \( \lambda_{j}(u) \) implicitly defined by the derivation of B-splines [4, 5, 6].

We define
\[
\lambda_{j}(u) = \exp(\lambda_{j}(u) \cdot \mathbf{d}_{j}),
\]
and re-formulate (1) as a recurrence relation:
\[
X(u) = X^{(0)}(u),
\]
\[
X^{(1)}(u) = X^{(1)}(u) \lambda_{j}(u), \quad \lambda_{j}(u) = \exp(\lambda_{j}(u) \cdot \mathbf{d}_{j}),
\]
(4)\( K \) and (5)\( K \) and (6)\( K \) then optimize (1) as a recurrence relation:

\[
X^{(1)}(u) = X^{(1)}(u) \lambda_{j}(u), \quad \lambda_{j}(u) = \exp(\lambda_{j}(u) \cdot \mathbf{d}_{j}),
\]
(3)

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 \( L = SO(3) \):

<table>
<thead>
<tr>
<th>( k ) Config.</th>
<th>Ours</th>
<th>Baseline</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 acc.</td>
<td>0.057</td>
<td>0.147</td>
<td>2.57x</td>
</tr>
<tr>
<td>4 vel.</td>
<td>0.058</td>
<td>0.088</td>
<td>1.52x</td>
</tr>
<tr>
<td>5 acc.</td>
<td>0.081</td>
<td>0.260</td>
<td>3.45x</td>
</tr>
<tr>
<td>5 vel.</td>
<td>0.082</td>
<td>0.141</td>
<td>1.73x</td>
</tr>
<tr>
<td>6 acc.</td>
<td>0.117</td>
<td>0.520</td>
<td>4.43x</td>
</tr>
<tr>
<td>6 vel.</td>
<td>0.111</td>
<td>0.217</td>
<td>1.95x</td>
</tr>
</tbody>
</table>

Optimization time in seconds for \( L = SE(3) \):

<table>
<thead>
<tr>
<th>( k ) Config.</th>
<th>Ours</th>
<th>Baseline</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 acc.</td>
<td>0.277</td>
<td>0.587</td>
<td>2.12x</td>
</tr>
<tr>
<td>4 vel.</td>
<td>0.253</td>
<td>0.334</td>
<td>1.32x</td>
</tr>
<tr>
<td>5 acc.</td>
<td>0.445</td>
<td>1.196</td>
<td>2.69x</td>
</tr>
<tr>
<td>5 vel.</td>
<td>0.405</td>
<td>0.581</td>
<td>1.43x</td>
</tr>
<tr>
<td>6 acc.</td>
<td>0.644</td>
<td>2.332</td>
<td>3.62x</td>
</tr>
<tr>
<td>6 vel.</td>
<td>0.590</td>
<td>0.936</td>
<td>1.55x</td>
</tr>
</tbody>
</table>

References

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

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

International Conference on Computer Vision and Pattern Recognition (CVPR) 2020