SPLINE FUSION:
A CONTINUOUS-TIME REPRESENTATION FOR VISUAL-INERTIAL FUSION WITH APPLICATION TO ROLLING SHUTTER CAMERAS

Steven Lovegrove, Alonso Patron-Perez and Gabe Sibley.
Robot Perception Group, George Washington University.
**Motivation:** Flexible SLAM / Calibration

Visual-Inertial Simultaneous Localization and Mapping

Calibration

N-Camera

Unsynchronized, high-rate devices
TRADITIONAL VISUAL SLAM / SfM

- Minimize landmark reprojection error across images
- Landmark positions and camera poses are estimation parameters
- Least Squares Minimization / EKF / UKF / ...
What about Rolling Shutter Cameras?

Each line exposed at different instance in time.
Rolling Shutters are common, but usually not modeled.
Can introduce significant bias, particularly during fast motion.
High-Rate / Unsyncronized Devices?

Estimating pose for each line / measurement may lead to under-constrained parameters, requiring motion model. Handling unsynchronized devices within discrete time framework implicitly requires motion model. Motion models are common in filtering, but not in BA.
The pose of the camera / rig / robot can be evaluated at any moment in time, and hence predictions and observed errors can be formed at any instance in time.

Supports high-rate and unsynchronized devices easily.
Rolling Shutter SLAM


Visual-inertial Fusion / Calibration


How do you interpolate poses?

We want:

• Local Control - for efficient optimization
• C-2 Continuity - enable us to predict IMU
• Approximation of minimum torque trajectories

We cannot just interpolate in any 6 DoF parametrization, in general we cannot guarantee reasonable trajectories.
How to generalize SLERP (Shoemake ‘85) to higher order smoothness? Quaternion Bézier curve or recursive SQUAD not necessarily C-2 continuous and do not have simple closed-form 2nd derivatives (Kim et al. ‘95).
The pose at time $t$ can be expressed as a weighted average of neighboring control poses expressed using this parametrization.

**CUMULATIVE B-SPLINES**


Linear interpolation: \( \mathbf{p}_{i,j} \in \mathbb{R}^N, t \in \mathbb{R} \)

\[
\text{lerp}(\mathbf{p}_i, \mathbf{p}_j, t) = \mathbf{p}_i(1 - t) + \mathbf{p}_j t
\]

Weighted average form

Cumulative form

\[
\mathbf{p}_i + (\mathbf{p}_j - \mathbf{p}_i)t = \Delta_i
\]

B-Splines:

\[
\mathbf{p}(t) = \sum_{i=0}^{n} \mathbf{p}_i B_{i,k}(t)
\]

Weighted average form

Cumulative form

Cumulative B-Splines:

\[
\mathbf{p}(t) = \mathbf{p}_0 \tilde{B}_{0,k}(t) + \sum_{i=1}^{n} (\mathbf{p}_i - \mathbf{p}_{i-1}) \tilde{B}_{i,k}(t)
\]

Cumulative form

![Graph of B,tilde_i,k(t)](image)

![Graph of B_i,k(t)](image)
Cumulative B-Splines

\[ p(t) = p_0 \tilde{B}_{0,k}(t) + \sum_{i=1}^{n} (p_i - p_{i-1}) \tilde{B}_{i,k}(t) \]

Cumulative B-Splines \( \in SE^3 \):

\[ T_{w,s}(t) = \exp(\tilde{B}_{0,k}(t) \log(T_{w,0})) \prod_{i=1}^{n} \exp(\tilde{B}_{i,k}(t) \log(T_{w,i-1}^{-1} T_{w,i})) = T_{w,0} = \Omega_i \]

Using the cumulative form, we need only take local differences on the manifold of \( \in SE^3 \) using \( \log \).

These differences \( \in se^3 \) represent rotational velocities, and it’s these that get blended.
\[ \tilde{B}(u) = C \begin{bmatrix} 1 \\ u \\ u^2 \\ u^3 \end{bmatrix}, \quad \dot{\tilde{B}}(u) = \frac{1}{\Delta t} C \begin{bmatrix} 0 \\ 1 \\ 2u \\ 3u^2 \end{bmatrix}, \quad \ddot{\tilde{B}}(u) = \frac{1}{\Delta t^2} C \begin{bmatrix} 0 \\ 0 \\ 2 \\ 6u \end{bmatrix}, \quad C = \frac{1}{6} \begin{bmatrix} 6 & 0 & 0 & 0 \\ 5 & 3 & -3 & 1 \\ 1 & 3 & 3 & -2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \]

\[ T_{w,s}(u) = T_{w,i-1} \prod_{j=1}^{3} \exp \left( \tilde{B}(u)_j \Omega_{i+j} \right), \]

\[ \dot{T}_{w,s}(u) = T_{w,i-1} \left( \dot{A}_0 A_1 A_2 + A_0 \dot{A}_1 A_2 + A_0 A_1 \dot{A}_2 \right), \]

\[ \ddot{T}_{w,s}(u) = T_{w,i-1} \left( \dot{A}_0 A_1 A_2 + A_0 \dot{A}_1 A_2 + A_0 A_1 \dot{A}_2 + 2 \left( \dot{A}_0 \dot{A}_1 A_2 + A_0 \dot{A}_1 \dot{A}_2 + A_0 \dot{A}_1 \dot{A}_2 \right) \right), \]

\[ A_j = \exp \left( \Omega_{i+j} \tilde{B}(u)_j \right), \quad \dot{A}_j = A_j \Omega_{i+j} \tilde{B}(u)_j, \]

\[ \ddot{A}_j = \dot{A}_j \Omega_{i+j} \tilde{B}(u)_j + A_j \Omega_{i+j} \ddot{B}(u)_j \]
Objective Function

Camera Prediction

\[ \mathbf{p}_b = \mathcal{W}(\mathbf{p}_a; \mathbf{T}_{b,a}, \rho) = \pi \left( [\mathbf{K}_b \mid 0] \mathbf{T}_{b,a} [\mathbf{K}_a^{-1} [\mathbf{p}_a^1] \mid \rho] \right) \]

IMU Prediction

Gyro\((u)\) = \[ \mathbf{R}_{w,s}^T(u) \cdot \dot{\mathbf{R}}_{w,s}(u) + \text{bias}, \]
Accel\((u)\) = \[ \mathbf{R}_{w,s}^T(u) \cdot (\ddot{s}_w(u) + g_w) + \text{bias} \]

Objective Function

\[ E(\theta) = \sum_{\hat{\mathbf{p}}_m} \left( \hat{\mathbf{p}}_m - \mathcal{W}(\mathbf{p}_r; \mathbf{T}_{c,s} \mathbf{T}_{w,s}(u_m)^{-1} \mathbf{T}_{w,s}(u_r) \mathbf{T}_{s,c}, \rho) \right)^2 + \]
\[ \sum_{\hat{\omega}_m} \left( \hat{\omega}_m - \text{Gyro}(u_m) \right)^2 + \sum_{\hat{\mathbf{a}}_m} \left( \hat{\mathbf{a}}_m - \text{Accel}(u_m) \right)^2 \]
SIMULATED VISUAL-INERTIAL SLAM
Rolling Shutter and IMU

Recover calibration and scale

True Distance: 25.45cm
Global Shutter: 25.31cm (0.55% err)
Rolling Shutter: 25.37cm (0.31% err)

Unknown landmarks
Known data association
MONOCULAR ROLLING SHUTTER SLAM
Linearization of projection

\[ p_b(t + \Delta t) = \mathcal{W}(p_a; T_{b,a}(t), \rho) + \Delta t \frac{d \mathcal{W}(p_a; T_{b,a}(t), \rho)}{dt} \]

Solve for time offset

\[ y_b(t + \Delta t) = \frac{h(t + \Delta t - s)}{e - s} \]
\[ \Delta t = -\frac{h.t_0 + s.(y_b(t) - h) - e.y_b(t)}{(s - e) \frac{d \mathcal{W}_y(p_a; T_{b,a}(t), \rho)}{dt} + h} \]

\~ 2-3 iterations
IMPROVED MONOCULAR ODOMETRY

- ground truth
- global shutter
- rolling shutter
**SPLINE FUSION**

Summary:
- Flexible continuous-time framework for SLAM / calibration
- Applicable to rolling-shutter cameras
- Can easily support inertial measurements
- And unsynchronized devices

For the future:
- Speed system up - runs at just few FPS right now
- Integrate with better front end
- Improve initialization
- Variable control pose placement

THANKS!

ANY QUESTIONS?

GWU, Robot Perception Group

Lovegrove, Patron-Perez, Sibley