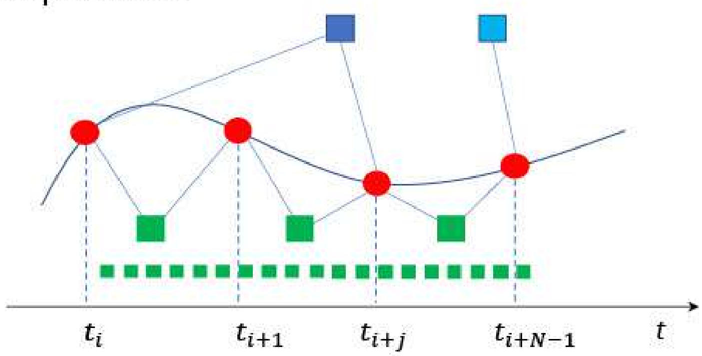
B-SPLINE CONTINUOUS-TIME OPTIMIZATION

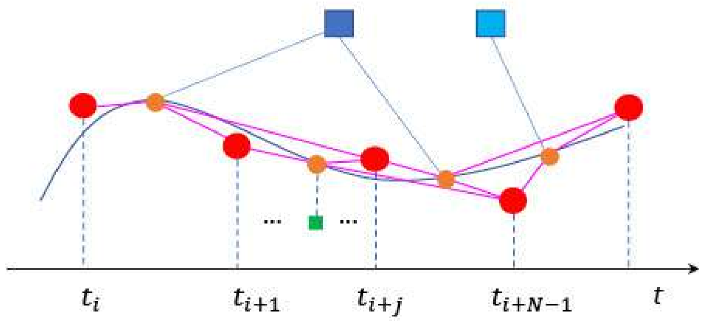
- Continuous-time optimization via B-spline interpolation:

  - **KEY TAKEOUT**:

    A pose $\mathbf{T}_t$ at time $t \in [t_i, t_{i+1})$ **can be linearly interpolated** from N control points $\mathbf{T}_i, \mathbf{T}_{i+1}, \dots \mathbf{T}_{i+N-1}$ via some time dependent weights.

  - **ADVANTAGES:**
    - Can integrate arbitrary number of sensors (multiple IMUs, multiple monocular VIOs...)
    - Can estimate *time delay.*
    - Can directly compensate for motion distortion in the MAP optimization.





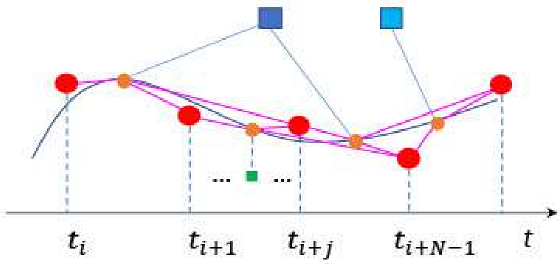
B-SPLINE CONTINUOUS-TIME OPTIMIZATION

- Continuous-time optimization via B-spline interpolation:
  - B-spline is defined by:
    - Its *order* N (or *degree* $N-1$), knot length Δt,
    - $K \geq N$ *knots* (*also called control points*):

$$\{\mathbf{T}_i \triangleq (\mathbf{R}_i, \mathbf{p}_i)\}_{i=0}^{K-1}, \mathbf{R}_i \in \mathrm{SO3}, \mathbf{p}_i \in \mathbb{R}^3$$

$$\mathbf{p}(s(t)) = \mathbf{p}_i + \sum_{j=1}^{N-1} \lambda_j(s(t))\,\mathbf{p}_{i+j} = \mathbf{p}_i + \sum_{j=1}^{N-1} \tilde{\lambda}_j(s(t))\,(\mathbf{p}_{i+j} - \mathbf{p}_{i+j-1})$$

$$\mathbf{R}(s(t)) = \mathbf{R}_i \prod_{j=1}^{N-1} \mathrm{Exp}\left(\tilde{\lambda}_j(s(t))\overbrace{\mathrm{Log}(\mathbf{q}_{i+j-1}^{-1} \circ \mathbf{q}_{i+j})}^{d_j}\right) = \mathbf{R}_i \prod_{j=1}^{N-1} A_j$$



$$\underbrace{(1, \lambda_1, \lambda_2, \dots, \lambda_{N-1})}_{N \times 1} \triangleq \boldsymbol{\lambda}(\mathrm{s}) = \underbrace{\mathbf{B}(N)}_{N \times N}\underbrace{(1, s, s^2, \dots, s^{N-1})}_{N \times 1}$$

$$\underbrace{(1, \tilde{\lambda}_1, \tilde{\lambda}_2, \dots, \tilde{\lambda}_{N-1})}_{N \times 1} \triangleq \boldsymbol{\lambda}(\mathrm{s}) = \underbrace{\tilde{\mathbf{B}}(N)}_{N \times N}\underbrace{(1, s, s^2, \dots, s^{N-1})}_{N \times 1}$$

$$s \triangleq \frac{(t - t_i)}{\Delta t}, \forall t \in [t_i, t_{i+1}),$$

B-SPLINE CONTINUOUS-TIME OPTIMIZATION

- Continuous-time optimization via B-spline interpolation:
  - B-spline is defined by:
    - Its *order* N (or *degree* $N-1$), knot length Δt,
    - $K \geq N$ *knots* (*also called control points*):

$$\{\mathbf{T}_i \triangleq (\mathbf{R}_i, \mathbf{p}_i)\}_{i=0}^{K-1}, \mathbf{R}_i \in \mathrm{SO3}, \mathbf{p}_i \in \mathbb{R}^3$$

$$\mathbf{p}(s(t)) = \mathbf{p}_i + \sum_{j=1}^{N-1} \lambda_j(s(t))\,\mathbf{p}_{i+j} = \mathbf{p}_i + \sum_{j=1}^{N-1} \tilde{\lambda}_j(s(t))\,(\mathbf{p}_{i+j} - \mathbf{p}_{i+j-1})$$

$$\mathbf{R}(s(t)) = \mathbf{R}_i \prod_{j=1}^{N-1} \mathrm{Exp}\Big(\tilde{\lambda}_j(s(t)) \overbrace{\mathrm{Log}\big(\mathbf{q}_{i+j-1}^{-1} \circ \mathbf{q}_{i+j}\big)}^{d_j}\Big) = \mathbf{R}_i \prod_{j=1}^{N-1} A_j$$

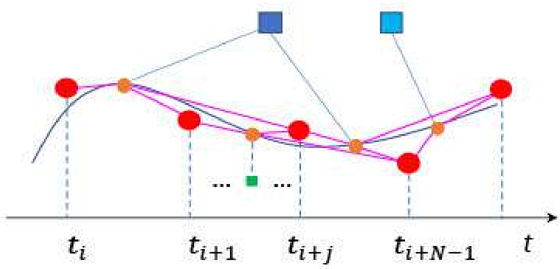
$$\underbrace{(1, \lambda_1, \lambda_2, \ldots, \lambda_{N-1})}_{N \times 1} \triangleq \boldsymbol{\lambda}(\mathrm{s}) = \underbrace{\mathbf{B}(N)}_{N \times N} \underbrace{(1, s, s^2, \ldots, s^{N-1})}_{N \times 1}$$

$$\underbrace{(1, \tilde{\lambda}_1, \tilde{\lambda}_2, \ldots, \tilde{\lambda}_{N-1})}_{N \times 1} \triangleq \boldsymbol{\lambda}(\mathrm{s}) = \underbrace{\tilde{\mathbf{B}}(N)}_{N \times N} \underbrace{(1, s, s^2, \ldots, s^{N-1})}_{N \times 1}$$

$$s \triangleq \frac{(t - t_i)}{\Delta t}, \forall t \in [t_i, t_{i+1}),$$

- **KEY TAKEOUT**:

  A pose $\mathbf{T}_t$ at time $t \in [t_i, t_{i+1})$ can be **linearly interpolated** from N control points $\mathbf{T}_i, \mathbf{T}_{i+1}, \ldots \mathbf{T}_{i+N-1}$ via some time dependent weights.



B-SPLINE CONTINUOUS-TIME OPTIMIZATION

- Gauss-Newton B-SPLINE Optimization on Manifold:

  Need to calculate the gradient → Jacobian.

  For 1st order observations, the Jacobian is:

Calculated as in Discrete Time case

$$J_{\mathbf{T}_j}^{\mathcal{Z}} = \frac{\partial h(\mathbf{T}_t, \breve{\mathcal{Z}})}{\partial \mathbf{T}_j} = \boxed{\frac{\partial h(\mathbf{T}_t, \breve{\mathcal{Z}})}{\partial \mathbf{T}_t}} \boxed{\frac{\partial \mathbf{T}_t}{\partial \mathbf{T}_j}}$$

- **KEY TAKEOUT**:

  Calculate the Jacobian as normal, then multiple it with $\frac{\partial \mathbf{T}_t}{\partial \mathbf{T}_j}$

$$\boxed{\frac{\partial \mathbf{p}_t}{\partial \mathbf{p}_j} = \lambda_j}$$

$$\boxed{\frac{\partial \mathbf{R}_t}{\partial \mathbf{R}_j} = \tilde{\lambda}_j \left(\prod_{j+1}^{m=N-1} A_m^{\top} \right) J_r(\tilde{\lambda}_j d_j) J_r^{-1}(d_j) - \tilde{\lambda}_{j+1} \left(\prod_{j+2}^{m=N-1} A_m^{\top} \right) J_r(\tilde{\lambda}_{j+1} d_{j+1}) J_r^{-1}(-d_{j+1})}$$

B-SPLINE CONTINUOUS-TIME OPTIMIZATION

Previous slide

- Gauss-Newton B-SPLINE Optimization on Manifold:

Acceleration factor:

$$J^{a}_{\mathbf{R}_j} = \frac{\partial}{\partial \mathbf{R}_j}\left[\mathbf{R}_t^{-1}(\ddot{\boldsymbol{p}}_t + \mathbf{g}) - \breve{a}\right] = \lfloor \mathbf{R}_t^{-1}\ddot{\boldsymbol{p}}_t \rfloor_\times \boxed{\frac{\partial \mathbf{R}_t}{\partial \mathbf{R}_j}}$$

$$J^{a}_{\mathbf{p}_j} = \frac{\partial}{\partial \mathbf{p}_j}\left[\mathbf{R}_t^{-1}(\ddot{\boldsymbol{p}}_t + \mathrm{g}) - \breve{a}\right] = \mathbf{R}_t^{-1} \boxed{\frac{\partial \ddot{\mathbf{p}}_t}{\partial \mathbf{p}_j}}$$

$$\boxed{\frac{\partial \ddot{\mathbf{p}}_t}{\partial \mathbf{p}_j} = \ddot{\lambda}_j = \frac{\lambda_j}{\Delta t^2}\frac{ds^j}{ds^2}}$$

B-SPLINE CONTINUOUS-TIME OPTIMIZATION

- Gauss-Newton B-SPLINE Optimization on Manifold:

Special factor based on IMU:

$$J^{\omega}_{\mathbf{R}_j} = \frac{\partial}{\partial \mathbf{R}_j}(\boldsymbol{\omega}_t - \breve{\omega}) = \frac{\partial \boldsymbol{\omega}_t}{\partial \mathbf{R}_j},$$

$$\omega_t = \omega^{(N)},$$

$$\omega^{(j)} = A_{j-1}^{\top}\omega^{(j-1)} + \dot{\tilde{\lambda}}_{j-1} d_{j-1}$$

$$\frac{\partial \omega_t}{\partial \mathbf{R}_j} = \frac{\partial \omega^{(j)}}{\partial \mathbf{d}_j} J_r^{-1}(d_j) - \frac{\partial \omega^{(j+1)}}{\partial \mathbf{d}_{j+1}} J_r^{-1}(-d_{j+1})$$

$$\frac{\partial \omega^{(j)}}{\partial \mathbf{d}_j} = \left(\prod_{j+1}^{m=N-1} A_m^{\top}\right)\left(\tilde{\lambda}_j A_j^{\top} \lfloor \omega^{(j)} \rfloor_{\times} J_r(-\tilde{\lambda}_j d_j) + \dot{\tilde{\lambda}}_j I\right)$$

B-SPLINE CONTINUOUS-TIME OPTIMIZATION

- Verified with Ceres Automatic derivatives
  - Exactly the same as automatic derivatives.
  - 80% faster
  - More concise and clearly written code ☺ .