Spline Based Sensor Fusion

Patron-Perez, Alonso, Steven Lovegrove, and Gabe Sibley. "A spline-based trajectory representation for sensor fusion and rolling shutter cameras." International Journal of Computer Vision 113.3 (2015): 208-219.

Lu Yu

Imorpheus.ai

Contributions

- Easily handles multiple sensors that send unsynchronized timestamped signals
- Time continuous trajectory
- C^2 smooth trajectory
- Rigid body motion without singularity
- Deals with rolling shutter camera naturally

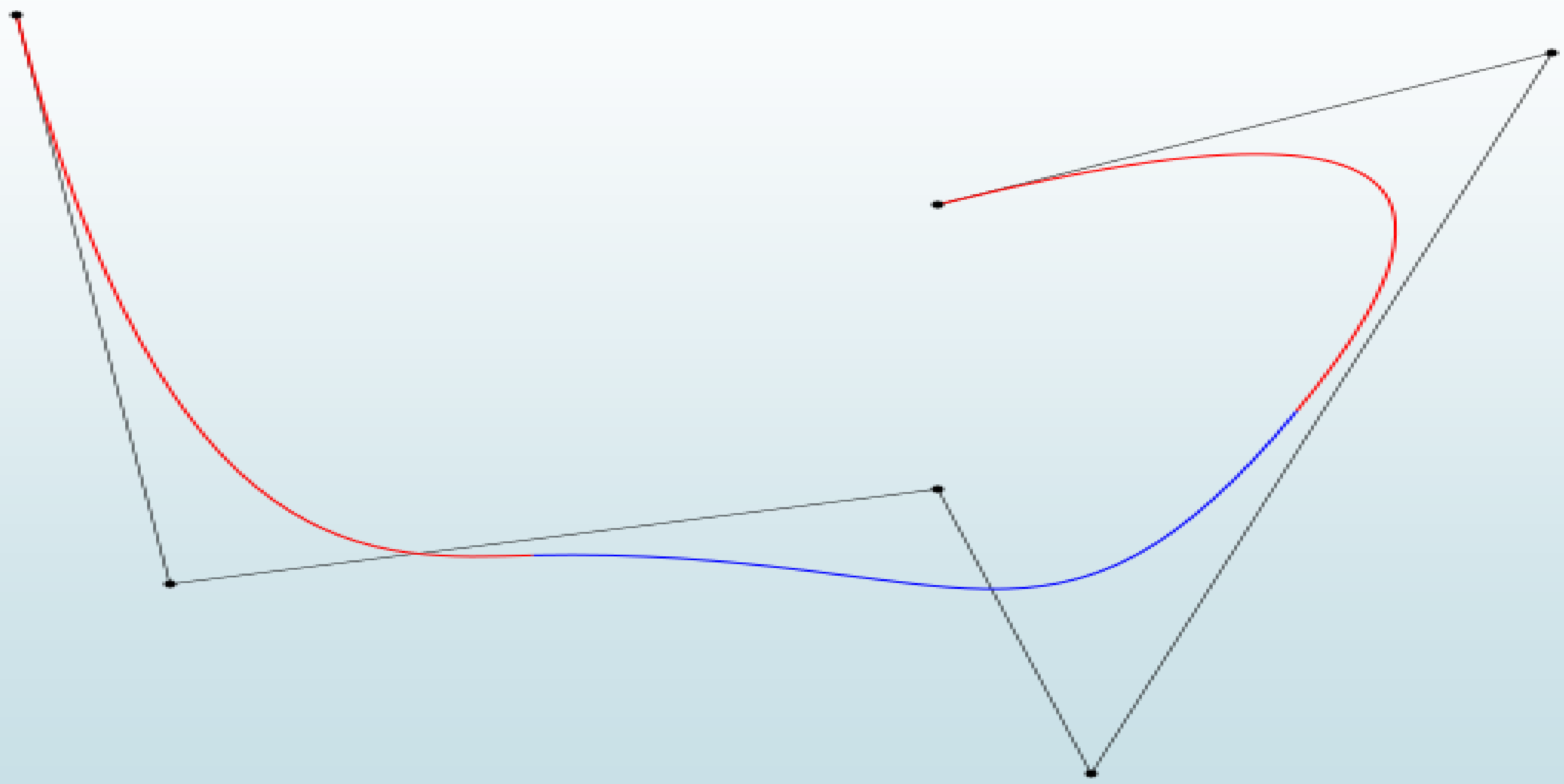
Technical outline

- Camera state represented in 3D homogenous coordinates (4D vector)
- Pose transformation matrix in Lie group SE(3)
- Logarithm of pose matrix in Lie algebra se(3)
- Cumulative cubic B-spline for trajectory fitting, using elements in SE(3) and se(3)

B-spline

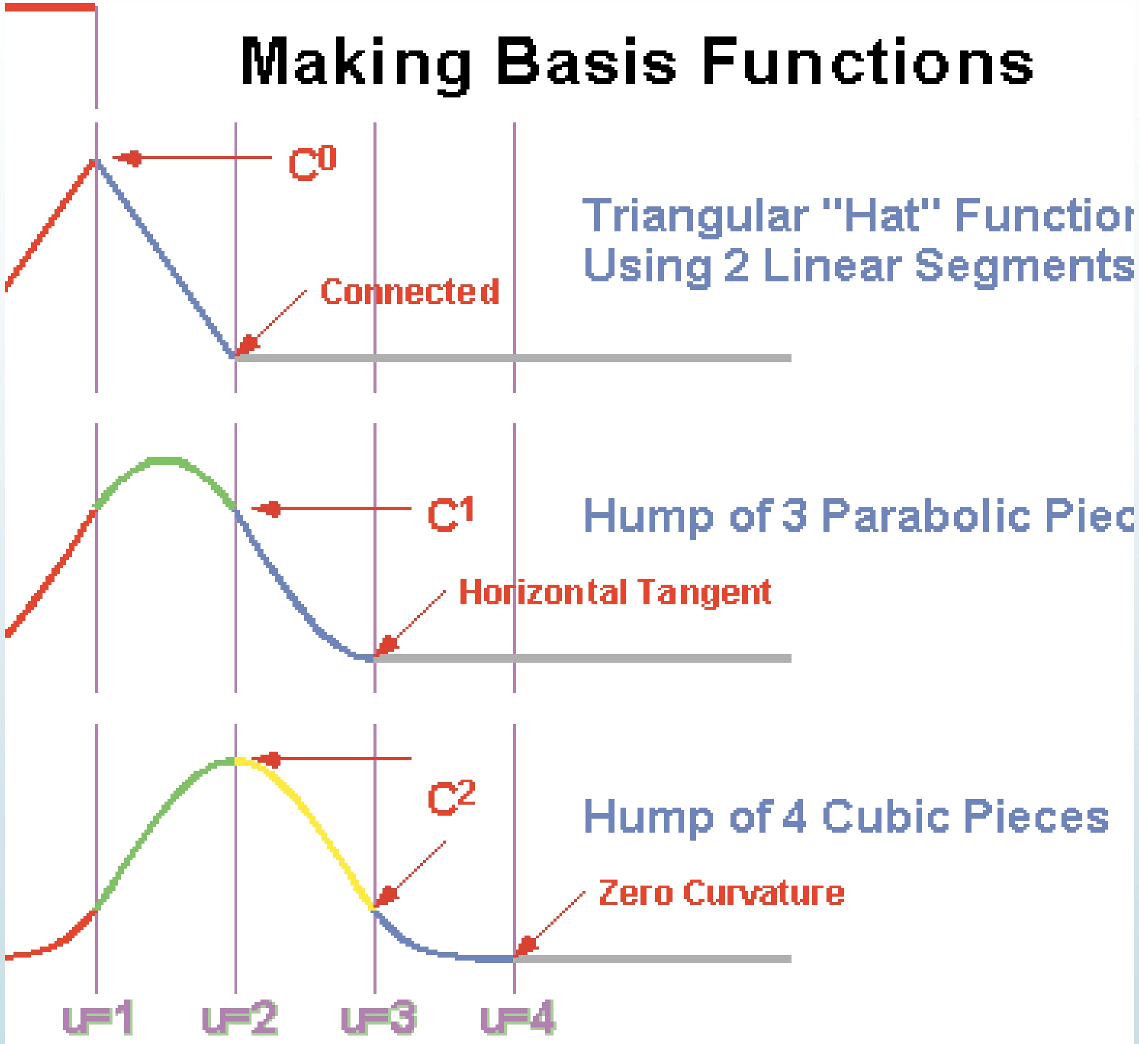
- Curve (in any dimension) parametrized by scalar t with a sequence t_i known as knots
- Piecewise polynomial function (basis function) that can be pre-calculated
- Shape affected by control points, which give value (can be weighted) at knots
- Fast evaluation by De Boor's algorithm

B-spline example



B-spline basis functions

- Recursively defined
- For order k basis, polynomial is of degree k-1 and curve has C^{k-2} smoothness at connections



B-spline equations

- Order k and n+1 control points $\mathbf{p}(t) = \sum_{i=0}^{n} \mathbf{p}_i B_{i,k}(t)$
- Cumulative form $\mathbf{p}(t) = \mathbf{p}_0 \tilde{B}_{0,k}(t) + \sum_{i=1}^{n} (\mathbf{p}_i - \mathbf{p}_{i-1}) \tilde{B}_{i,k}(t)$
- Basis function

$$B_{i,0}(x) := \begin{cases} 1 & \text{if} \quad t_i \leq x < t_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

$$B_{i,k}(x) := \frac{x - t_i}{t_{i+k} - t_i} B_{i,k-1}(x) + \frac{t_{i+k+1} - x}{t_{i+k+1} - t_{i+1}} B_{i+1,k-1}(x)$$

Cubic uniform B-spline

- Uniformly distributed knots $t_0, \dots, t_n$ with interval Δt
- Reparametrize by $s(t) = (t - t_0)/\Delta t$ and $u(t) = s(t) - s(t_i) = s(t) - i$ for $s(t_i) \leq s(t) < s(t_{i+1})$
- Matrix representation of cumulative B-spline

$$\tilde{\mathbf{B}}(u) = \mathbf{C}\begin{bmatrix} 1 \\ u \\ u^2 \\ u^3 \end{bmatrix}, \quad \dot{\tilde{\mathbf{B}}}(u) = \frac{1}{\Delta t}\mathbf{C}\begin{bmatrix} 0 \\ 1 \\ 2u \\ 3u^2 \end{bmatrix},$$

$$\ddot{\tilde{\mathbf{B}}}(u) = \frac{1}{\Delta t^2}\mathbf{C}\begin{bmatrix} 0 \\ 0 \\ 2 \\ 6u \end{bmatrix}, \quad \mathbf{C} = \frac{1}{6}\begin{bmatrix} 6 & 0 & 0 & 0 \\ 5 & 3 & -3 & 1 \\ 1 & 3 & 3 & -2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

SE(3) and SO(3)

Under 3D homogeneous coordinates, pose transformations (from a to b) are represented by 4x4 matrix, where SE(3) is the special Euclidean group, SO(3) is the special orthogonal group and a_b is the translation from a to b.

$$\mathbf{T}_{b,a} = \begin{bmatrix} \mathbf{R}_{b,a} & \mathbf{a}_b \\ \mathbf{0}^T & 1 \end{bmatrix}, \quad \mathbf{T}_{b,a} \in \mathbb{SE}3, \quad \mathbf{R}_{b,a} \in \mathbb{SO}3$$

Why Lie group

- In robotics and computer vision, transformations must be inverted, differentiated and interpolated. Lie groups and Lie algebras form mathematical basis for this purpose.
- A Lie group is a topological group that is also a smooth manifold.
- Every Lie group has an associated Lie algebra, which is the tangent space (vector space) around the identity element of the group.

Matrix exponential and logarithm

- Matrix exponential is defined by power series

$$e^{\mathbf{A}} \equiv \sum_{n=0}^{\infty} \frac{\mathbf{A}^n}{n!}.$$

- Matrix logarithm ("differentiation") is the inverse of exponential (multi-valued)

if exp(A) = B, log(B) = A

Lie group and Lie algebra

- For matrix A in Lie group SE(3), log(A) is in its associated Lie algebra se(3).
- For matrix B in Lie algebra se(3), exp(B) is in its associated Lie group SE(3).
- Both have closed form solutions.

Exponential map for se(3)

$$G_1 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \quad G_2 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \quad G_3 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix},$$

$$G_4 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \quad G_5 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \quad G_6 = \begin{pmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\begin{aligned} \begin{pmatrix} \mathbf{u} & \boldsymbol{\omega} \end{pmatrix}^T &\in \mathbb{R}^6 \\ \mathbf{u}_1 G_1 + \mathbf{u}_2 G_2 + \mathbf{u}_3 G_3 + \boldsymbol{\omega}_1 G_4 + \boldsymbol{\omega}_2 G_5 + \boldsymbol{\omega}_3 G_6 &\in \mathrm{se}(3) \end{aligned}$$

Exponential map for se(3)

$$
\begin{aligned}
\boldsymbol{\delta} = \begin{pmatrix} \mathbf{u} & \boldsymbol{\omega} \end{pmatrix} &\in \mathrm{se}(3) \\
\exp(\boldsymbol{\delta}) &= \exp\left(\begin{array}{c|c} \boldsymbol{\omega}_\times & \mathbf{u} \\ \hline \mathbf{0} & 0 \end{array}\right)
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{u}, \boldsymbol{\omega} &\in \mathbb{R}^3 \\
\theta &= \sqrt{\boldsymbol{\omega}^T \boldsymbol{\omega}} \\
A &= \frac{\sin\theta}{\theta} \\
B &= \frac{1-\cos\theta}{\theta^2} \\
C &= \frac{1-A}{\theta^2} \\
\mathbf{R} &= \mathbf{I} + A\boldsymbol{\omega}_\times + B\boldsymbol{\omega}_\times^2 \\
\mathbf{V} &= \mathbf{I} + B\boldsymbol{\omega}_\times + C\boldsymbol{\omega}_\times^2 \\
\exp\begin{pmatrix} \mathbf{u} \\ \boldsymbol{\omega} \end{pmatrix} &= \left(\begin{array}{c|c} \mathbf{R} & \mathbf{V}\mathbf{u} \\ \hline \mathbf{0} & 1 \end{array}\right)
\end{aligned}
$$

Ego state estimation

- As analogue to position and velocity, we can differentiate transformation matrix by taking matrix logarithm

$$\mathbf{v} = \frac{1}{\Delta t}(\mathbf{p}_b - \mathbf{p}_a)$$

$$\boldsymbol{\Omega} = \frac{1}{\Delta t}\log\left(\mathbf{T}_{b,a}\right), \quad \boldsymbol{\Omega} \in \mathbb{R}^{4\times 4}$$

- Trajectory can be expressed as

$$\mathbf{p}(t) = \mathbf{p}_a + t\,\mathbf{v}_{b,a}$$

$$\begin{aligned}\mathbf{T}_w(t) &= \mathbf{T}_{w,a}\exp(t\log(\mathbf{T}_{w,b}^{-1}\mathbf{T}_{w,a})) \\ &= \mathbf{T}_{w,a}\exp(t\,\boldsymbol{\Omega}_{b,a})\end{aligned}$$

Ego state estimation

- Applying cumulative cubic B-spline and its matrix form

$$\mathbf{T}_{w,s}(t) = \exp\left(\tilde{B}_{0,k}(t)\log(\mathbf{T}_{w,0})\right)\prod_{i=1}^{n}\exp\left(\tilde{B}_{i,k}(t)\boldsymbol{\Omega}_i\right)$$

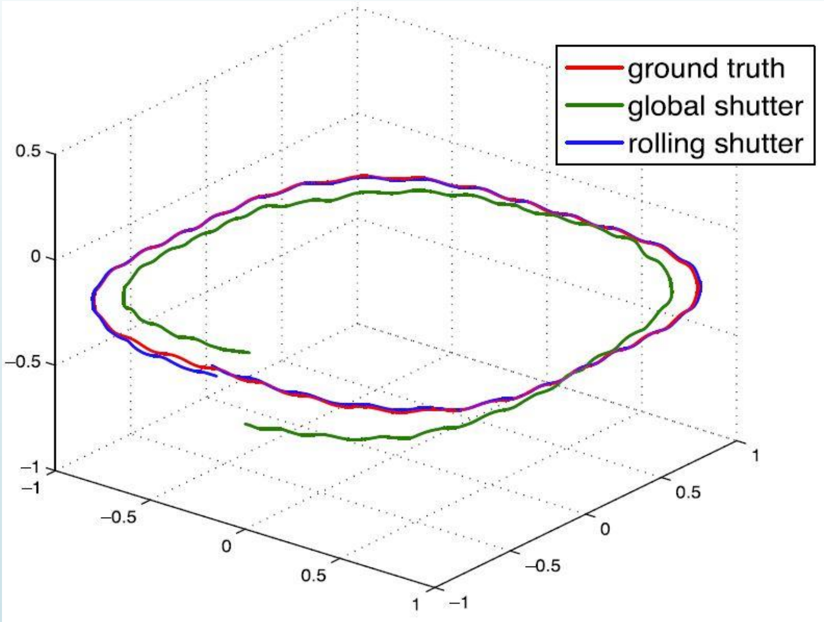
$$\mathbf{T}_{w,s}(u) = \mathbf{T}_{w,i-1}\prod_{j=1}^{3}\exp\left(\tilde{\mathbf{B}}(u)_j\,\boldsymbol{\Omega}_{i+j}\right)$$

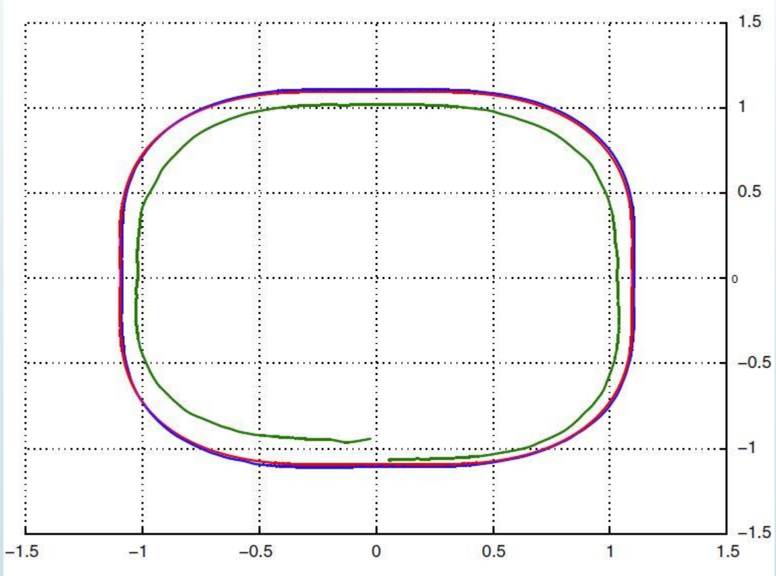
- The position-velocity analogue is

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

Experiment

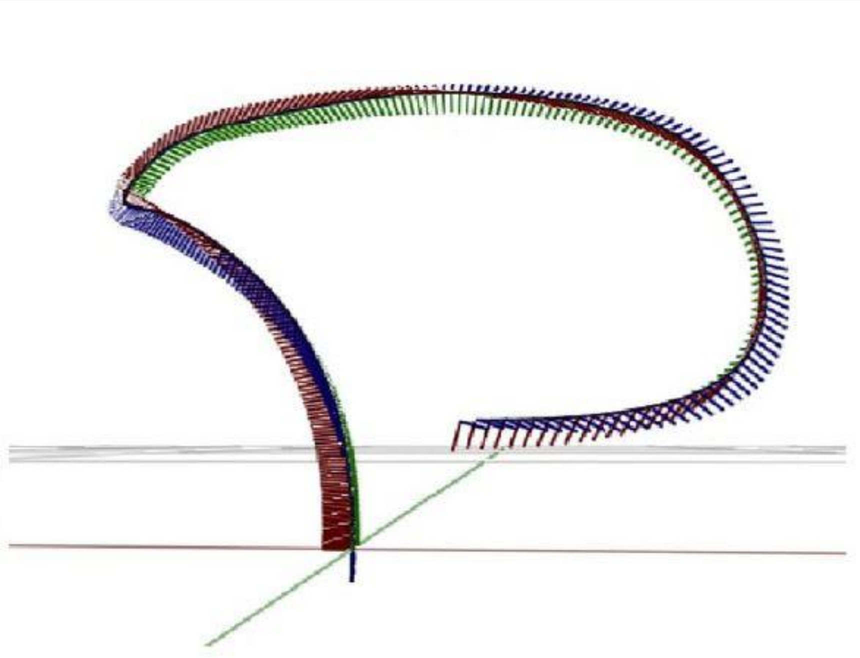
Results from simulated monocular rolling shutter camera

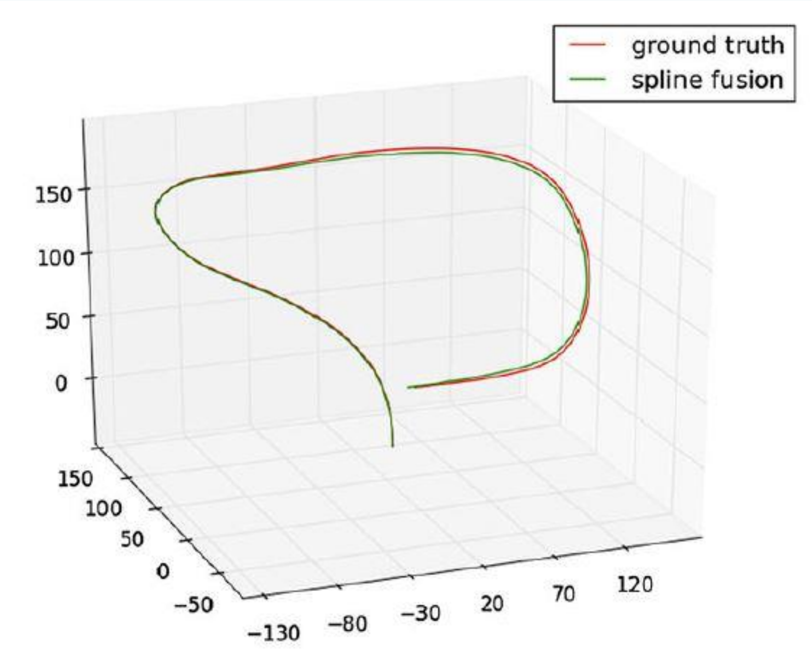




Experiment

Results from Tsubuka dataset, 3D trajectory in office environment





Thank you

Lu Yu

Imorpheus.ai