HyperSLAM: A Generic and Modular Approach to Sensor Fusion and Simultaneous Localization And Mapping in Continuous-Time

*Hyper*SLAM: A Generic and Modular Approach to Sensor Fusion and Simultaneous Localization And Mapping in Continuous-Time

David Hug and Margarita Chli
Vision for Robotics Lab, ETH Zürich
www.v4rl.ethz.ch

Abstract

Within recent years, Continuous-Time Simultaneous Localization And Mapping (CTSLAM) formalisms have become subject to increased attention from the scientific community due to their vast potential in facilitating motion-corrected feature reprojection and direct unsynchronized multi-rate sensor fusion. They also hold the promise of yielding better estimates in traditional sensor setups (e.g. visual, inertial) when compared to conventional discrete-time approaches. Related works mostly rely on cubic, $\mathcal{C}^2$-continuous, uniform cumulative B-Splines to exemplify and demonstrate the benefits inherent to continuous-time representations. However, as this type of splines gives rise to continuous trajectories by blending uniformly distributed $\mathbb{SE}_3$ transformations in time, it is prone to under- or overparametrize underlying motions with varying volatility and prohibits dynamic trajectory refinement or sparsification by design. In light of this, we propose employing a more generalized and efficient non-uniform split interpolation method in $\mathbb{R} \times \mathbb{SU}_2 \times \mathbb{R}^3$ and commence with development of 'HyperSLAM', a generic and modular CTSLAM framework. The efficacy of our approach is exemplified in proof-of-concept simulations based on a visual, monocular setup.

1. Introduction

Simultaneous Localization And Mapping (SLAM), in its most elementary and abstracted formulation, addresses the question of simultaneous ego-motion estimation and scene reconstruction based on arbitrary sensory input. Due to their affordability, relatively small size and information-rich output, monocular, frame-based cameras, operating in the visible spectrum, have been established as a solid and suitable choice in applications requiring SLAM with minimal sensory input, such as in aerial navigation [1]. Within this context, it has become the norm [2] in SLAM literature to pose the underlying optimization over a discretized set of camera poses in $\mathbb{SE}_3$ (e.g. whenever a new camera frame arrives). There are, however, known shortcomings of purely monocular SLAM, which generally, cannot be easily overcome without using additional sensing modalities. Specifically, motion blur and highly dynamic scenes detriment the trajectory estimation significantly, while any estimates (i.e. of camera poses and landmarks) are only recoverable up to scale. Hence, to address these issues it is often essential to incorporate auxiliary sensors (e.g. stereo visual(-inertial) setups) to augment the performance and reliability of the SLAM pipeline, especially when SLAM estimates are used to automate the navigation of mobile platforms [3, 4]. This, however, implies increased complexity with respect to sensor fusion and synchronization issues between the sensors.

Regardless of potentially detrimental effects of preintegrating arriving measurements (e.g. incoming data from Inertial Measurement Units (IMUs) or accumulated event-frames from Dynamic Vision Sensors (DVS)), most SLAM frameworks opt for a discrete-time SLAM formulation, due to its simplicity mostly. However, the assumption of such discrete-time paradigm becomes a bottleneck when adding more sensors with variable data rates to the sensor suite requiring more elaborate, generic and asynchronous fusion. The traditional alternative of assigning every measurement its own state would cause the size of the optimization problem to explode. Furthermore, some sensory updates do not contain sufficient information to completely determine a pose in $\mathbb{SE}_3$ by themselves (e.g. a single event stemming from an event-based camera). Within recent years, the concept of formulating the SLAM problem as a spline-based continuous-time non-linear least squares optimization gave rise to an elegant way to bound the state size, whilst allowing completely asynchronous integration of sensory input. In particular, the formalism of uniform cumulative B-Splines, which has become the go-to representation of such problems in the literature, gives rise to a continuous-time representation, which can be used to query the estimated trajectory $T(t) \in \mathbb{SE}_3$ at any possible instance in time t. This representation was demonstrated to be particularly effective in compensating for rolling-shutter effects [5] as well as improving the accuracy of event-based approaches [6, 7].

Despite leveraging the benefits of spline-based methods to significantly compress the problem's state size, many approaches in the literature fail to achieve real-time performance as the underlying representation exposes several disadvantages. For instance, some methods pose the problem as a batch optimization, which is unsuitable for online operation. Furthermore, a few shortcomings can be directly accredited to the selection of cumulative B-Splines to query the continuous-time trajectory. Notably, it demands the recomputation of many incremental terms and matrix expressions upon evaluation of the trajectory. Additionally, uniform B-Splines do not permit dynamic refinement or sparsification of the underlying trajectory representation and, hence, are prone to over- or underparametrization. Driven by these observations and the need for real-time capable continuous-time frameworks for SLAM, we propose the use of a split representation [8] that facilitates faster evaluations, exposes tractable derivatives and allows to dynamically refine our trajectory representation according to external requirements. Concisely, the contributions and augmentations of this paper are the following:

- A fast non-uniform representation in $\mathbb{R} \times \mathbb{SU}_2 \times \mathbb{R}^3$ specifically designed to address CTSLAM problems.
- Required analytic derivatives to efficiently optimize non-uniform spline bases temporally.
- Development of '*Hyper*SLAM', a generic and modular continuous-time framework.
- Demonstration of the efficacy of the proposed system in monocular simulated proof-of-concept experiments.

2. Related Work

Over the years, the mathematical construct of (cumulative) B-Splines has proven to be a very powerful concept applicable to many scientific disciplines achieving continuous inter- or extrapolation. In particular, the popularity of B-Splines is a result of several, desirable inherent characteristics, such as having local support, a high level of smoothness (i.e. $\mathcal{C}^2$-continuity) and a compact mathematical representation [9]. However, it has only been within recent years that they achieved to broadly claim their well-deserved place within the field of Robotics and Computer Vision. Featuring in various works, they were shown to be a well-suited approach whenever states need to be represented in a non-discretized, continuous fashion.

Notably, [10] leveraged the use of B-Splines to formulate the SLAM problem in continuous-time as a Maximum (*a posteriori*) Likelihood Estimation (MLE), ultimately aiming at a reduction in overall state size of multi-sensor setups. Similarly, in [6] they were utilized to more tightly integrate the highly asynchronous event-based camera output into a Non-Linear Least Squares (NLLS) minimization to achieve ego-motion estimation. The approach was augmented to incorporate inertial measurements [7] later on. Related paradigms, specifically a hierarchical wavelet decomposition, have been explored in [11], where the main novelties lied with introducing means to adaptively refine trajectory estimates by superimposing wavelets at different scales atop a basic, coarse representation.

As explored in [5], B-Splines also turn out to be a suitable choice to effectively account and compensate for rolling shutter effects commonly detrimental to SLAM pipelines. The preconditions to allow efficient loop closing for CT-SLAM problems in large scale environments were investigated in [12], where it was verified that moving to a relative representation is advantageous. Furthermore, a continuous formulation also proves to be of great value in the context of LIDARs aboard vehicles [13], where it was deployed to achieve a refined alignment of LIDAR measurements.

More recent jointly probabilistic and continuous approaches based on Gaussian processes were presented in [14]. A recurring shortcoming of previous implementations, however, is the circumstance that they do not achieve real-time performance in generic cases, imposing a limit to their applicability. It was only recently, that [15] addressed the computational complexity by formulating derivatives in a recursive manner, which greatly alleviates performance issues with B-Splines in $\mathbb{SE}_3$.

In response to these inherent challenges and limitations of B-Splines, here we further explore the limitations and capabilities of alternative continuous-time representations [8] towards a generic and expandable spline-based CTSLAM framework allowing straight-forward and swift integration of synchronized, or even unsynchronized, sensor suites. Furthermore, the absence of any open-source continuous-time pipelines specifically tailored to perform SLAM emboldens us to pursue the development of a publicly available generic CTSLAM framework in the long run.

3. Methodology

3.1. Preliminaries

Following previous works [5–7, 13], here, in accordance with Eq. (1), we denote transformations converting homogeneous points $X_a \in \mathbb{R}^4$ in frame a to their equivalents X_b in frame b, such that $X_b \sim T_a^b X_a$ with $T_a^b \in \mathbb{SE}_3$.

$$T_a^b = \begin{bmatrix} R_a^b & t_a^b \\ 0 & 1 \end{bmatrix} \in \mathbb{R}^{4\times 4}, \text{ with } R_a^b \in \mathbb{SO}_3,\ t_a^b \in \mathbb{R}^3 \quad (1)$$

$$\hat{\xi} = \begin{bmatrix} \hat{\omega} & v \\ 0 & 0 \end{bmatrix}, \text{ with } \hat{\omega}^\top = -\hat{\omega} \in \mathbb{R}^{3\times 3},\ v \in \mathbb{R}^3 \quad (2)$$

As is well established, every transformation T_a^b in $\mathbb{SE}_3$ has a corresponding representation $\xi_a^b \in \mathbb{R}^{4\times 4}$ (i.e. twists) in its Lie algebra $\mathfrak{se}_3$. Conversion between these representations can be achieved by means of appropriately defining exponential and logarithmic maps as detailed in [2]. Thus, omitting all sub- and superscripts for clarity, $\hat{\xi} = \log(T)$ and $T = \exp(\hat{\xi})$, where the structure of $\hat{\xi}$ is as in Eq. (2). Owing to the special structure of $\hat{\xi}$, it can be brought into vector form $\xi = [\omega; v] \in \mathbb{R}^6$ (i.e. twist coordinates), where ω is found by converting the underlying skew-symmetric matrix $\hat{\omega}$ to its vector form. Specifically, $\hat{\omega}$ fulfills the condition that $\hat{\omega}x = \omega \times x$, for any $x \in \mathbb{R}^3$.

3.2. Cumulative B-Splines in $\mathbb{SE}_3$

Due to the local support property of cubic, cumulative B-Splines, it is sufficient to only consider the placement of four spline bases to fully determine the trajectory at any given time. Specifically, a pose query at any time $t \in [t_i, t_{i+1})$ (i.e. a segment of the spline) only relies on the bases located at times $\{t_{i-1}, t_i, t_{i+1}, t_{i+2}\}$ and the absolute pose of the spline $T_s^w(t)$, where w denotes the world and s the spline coordinate frame, can then be computed:

$$T_s^w(t) = T_{i-1}^w \prod_{j=i}^{i+2} \delta T_j \tag{3}$$

$$\text{with } \delta T_j = \exp\left(\lambda_{j-i}(t)\delta\hat{\xi}_j\right), \ \Lambda(t) = MU(t), \tag{4}$$

$$M = \frac{1}{6}\begin{bmatrix} 5 & 3 & -3 & 1 \\ 1 & 3 & 3 & -2 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \ U(t) = \begin{bmatrix} 1 \\ u(t) \\ u(t)^2 \\ u(t)^3 \end{bmatrix} \text{ and} \tag{5}$$

$$\delta\hat{\xi}_j := \log(\exp(\hat{\xi}_w^{j-1})\exp(\hat{\xi}_j^w)) = \log(T_j^{j-1}) \in \mathfrak{se}_3. \tag{6}$$

Above, the expression $\lambda_n(t)$ refers to the n-th zero-based element within $\Lambda(t)$ (i.e. the blending vector). Thus far, uniform B-Splines are considered, implying time intervals $\Delta t = t_{i+1} - t_i$ of equal duration for all valid values of i and a fixed mixing matrix M. The expression $u(t) = (t - t_i)/\Delta t$ in Eq. (5) is the normalized time associated with a particular spline segment.

3.3. Issues and Disadvantages of Cumulative B-Spline Representations

To our best knowledge, continuous-time SLAM representations based on cumulative B-Splines relying on Lie groups were first proposed in [5] and were, *de facto*, used as the standard representation in various works [5–7, 13] ever since. Nonetheless, [10] presented a very efficient approach to represent a continuous-time trajectory by means of using superposition-based B-Splines beforehand. However, as pointed out in [5], the representation in [10] exposes several disadvantages with respect to inherent singularities at π and non-geodesic interpolations in $\mathbb{SO}_3$ due to its underlying Cayley-Gibbs-Rodrigues parametrization. Contrary, the cumulative formulation in [5] follows geodesics in $\mathbb{SE}_3$, but its

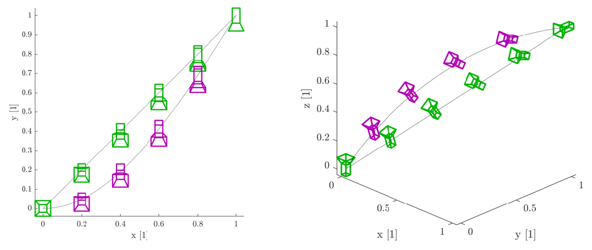


Figure 1: Qualitative comparison of two examples of trajectory interpolations in $\mathbb{SE}_3$ (in magenta) and $\mathbb{SU}_2 \times \mathbb{R}_3$ (in green). Although the transformations at the beginning and end of the two trajectories are identical, the interpolation in $\mathbb{SE}_3$ does not follow an intuitive shortest path, but geodesics in $\mathbb{SE}_3$ without any physical relevance.

evaluation is considerably more expensive owing to the necessity of mapping back and forth between the manifold $\mathbb{SE}_3$ and the tangent space $\mathfrak{se}_3$ as well as due to requiring concatenated matrix multiplications. In particular, the evaluation of δT_j in Eq. (6) involves many arithmetic and trigonometric operations, which even its closed-form expression, the Baker-Campbell-Hausdorff formula, which avoids explicit computation of exponentials and logarithms, [16] for elements in $\mathfrak{so}_3$, $\mathfrak{se}_3$ respectively, is not able to alleviate due numerous internal trigonometric evaluations.

Furthermore, by consideration of the closed-form $\mathfrak{se}_3$ matrix exponential, the rotation matrix $R(\omega)$ and the expression $V(\omega)$ in Eq. (7), as defined in [2], one may observe that the translation and angular components become inherently coupled as exemplified in Fig. (1). Clearly, said coupling further increases the computational effort required upon spline evaluations, but more importantly, gives rise to complexer Jacobians whose non-block-diagonal nature necessitates undesirable updates of v on changes in ω.

$$\exp(\hat{\xi}) = \begin{bmatrix} R(\omega) & V(\omega)\,v \\ 0 & 1 \end{bmatrix} \tag{7}$$

3.4. Split Representation of Uniform B-Splines

In order to address the aforementioned shortcomings of the prevalently used representation in Eq. (3), and in order to facilitate the extension to non-uniform B-Splines, we adopt the split-interpolation-based method of [8] often deployed in computer graphics with the main difference that we augment each base with a notion of time. Thus, our spline bases $\mathfrak{B}$, effectively, can be seen as poses in $\mathbb{SE}_3^t := \mathbb{R} \times \mathbb{SU}_2 \times \mathbb{R}^3$, where $\mathbb{R}$ denotes the temporal component t_i of base $b_i \in \mathfrak{B}$ and the remaining component $\mathbb{SU}_2 \times \mathbb{R}^3$ comprises a unit quaternion and a translational part $\{q_i, x_i\} \in \mathbb{R}^7$ as in [17]. Despite their non-minimality, operations on elements in the space $\mathbb{SE}_3^t$ are extremely fast, the representation itself remains singularity-free and it also serves the purpose of decoupling translational and angular components. Owing to the fact that geodesics in $\mathbb{SE}_3$, unlike geodesics in $\mathbb{R}^3$ and $\mathbb{SO}_3$, do not have any tangible physical meaning in any case, a decoupled representation grants a more natural and intuitive interpretation of the interpolated transformation $T_s^w(t)$ (see Fig. (1)), which, analogously to Eq. (3), is defined as:

$$T_s^w(t) = \begin{bmatrix} R(q_s^w(t)) & x_s^w(t) \\ 0 & 1 \end{bmatrix} \in \mathbb{R}^{4\times 4}, \tag{8}$$

where $q_s^w(t) \in \mathbb{SU}^2$ is a unit quaternion and $x_s^w(t) \in \mathbb{R}^3$ is a translation vector. Both expressions can be evaluated independently as:

$$q_s^w(t) = q_{i-1}^w \prod_{j=i}^{i+2} \Gamma_j^i \in \mathbb{SU}_2, \tag{9}$$

$$\text{where } \Gamma_j^i := (\delta q_j)^{\lambda_{j-i}(t)} \text{ and } \delta q_j = q_w^{j-1} q_j^w \tag{10}$$

$$x_s^w(t) = x_{i-1}^w + \sum_{j=i}^{i+2} \Upsilon_j^i \in \mathbb{R}_3, \tag{11}$$

$$\text{where } \Upsilon_j^i := \lambda_{j-i}(t)\, \delta x_j \text{ and } \delta x_j = x_j^w - x_{j-1}^w. \tag{12}$$

Above, $\lambda_{j-i}(t)$ is consistent with its prior definition from Eq. (4). As concluded in [8, 15], split representations are a preferential choice for real-time systems due to increased efficiency. In [18], it was also shown that they are physically better suited and that they offer a faster convergence rate. Thus, we adopt a split representation $\mathbb{SE}_3^t$ (i.e. timed split transformations) into our framework.

3.5. Generalization to Non-Uniform B-Splines

Another limitation of uniform B-Splines is the necessity for spline bases $b_i = \{t_i, q_i, x_i\}$ to be spaced equidistantly in time. However, this may be an undesirable property in cases, where uniform B-Splines either under- or overfit the incoming data. For example, a static position of infinite duration can be represented by a limited number of bases in the non-uniform formulation, whilst, in the case of uniform B-Splines, the number of bases tends towards infinity over time. Conversely, the undergone trajectory might be too complex to properly be captured by equidistant bases with interval Δt, while it could locally be captured perfectly by a base spacing of $\Delta t^* < \Delta t$. Hence, we extend existing approaches by moving to a non-uniform spline, which is able to address this issue by reformulating the interpolation matrices $\Lambda(t)$ and $M(\{t_{i-2}, \ldots, t_{i+3}\})$ in Eq. (4, 5) according to Eq. (13), where time-dependency is implicitly assumed. In particular, the matrix $M_3^i(t)$ is the cumulative, cubic version of the generic, recursively defined mixing matrix found in [9] with appropriately chosen entries. As one observes, the spline evaluation now depends on the time associated with bases $\{k_{i-2}, \ldots, k_{i+3}\}$, implying an additional dependency on bases k_{i-2} and k_{i+3}. Owing to the introduced time-dependencies, it also becomes evident why representing the non-uniform B-Spline as poses in $\mathbb{SE}_3^t$ is a preferential option, especially with respect to storing parameters in contiguous memory locations, as well as generally reducing interdependencies between (individual) parameter blocks. In Eq. (13), $U_3^i(t)$ is a generalized, cubic stacked polynomial vector of normalized times with entries $u_n^i(t) = (t - t_i)^n/(t_{i+1} - t_i)^n$ analogous to Eq. (5).

$$\Lambda_3^i = M_3^i U_3^i = \begin{bmatrix} 1-\mu_0 & 3\mu_0 & -3\mu_0 & \mu_0 \\ \mu_{10} & 3\mu_{11} & 3\mu_{12} & \mu_{13} \\ 0 & 0 & 0 & \mu_2 \end{bmatrix} U_3^i \tag{13}$$

This formulation permits dynamic trajectory refinement where required, offers a significant reduction in state size for static trajectories, or trajectories with slow dynamics and can swiftly be augmented to any order. All aforementioned aspects are advantageous to aid avoiding either redundancy or overparametrization of the underlying path within the NLLS optimization and, ultimately, enable us to formulate a universally applicable spline container that is completely agnostic to its underlying bases, to its order, and to whether it is of uniform or non-uniform kind.

3.6. Abstract System Description

Subsequent sections target the treatment of important, conceptual aspects related to the design of *Hyper*SLAM and are presented to provide a theoretical introspective into the aspired system layout. Thus, they also address potential incorporation of sensors that surpass the scope of the presented experimental parts of this work.

Throughout the development of *Hyper*SLAM, we adhere to several fundamental, axiomatic policies: generality, modularity, expandability and intuitiveness. Thus, all formulations ought to be independent of traits inherent to any specific sensory setup, algorithmic approach or system architecture. Conceptually, our system design follows principles of established SLAM frameworks [19, 20] and combines them with a rethought systematic top-down approach as to how components should be designed hierarchically and interconnected intuitively.

Thus, we commence by posing the SLAM problem in its most abstract form. Specifically, each SLAM system s from a collection of systems $\mathcal{S}$ (e.g. as in multi-agent SLAM) processes its data on clusters of available computational nodes $\mathcal{N}_s$ (e.g. CPU cores). Each node n_s, in turn, is responsible for preprocessing a (sub)set of sensory measurements $\mathcal{M}_{\mathcal{D}_s}$, originating from a (sub)collection of devices $\mathcal{D}_s$ associated to system s. Every such measurement gets processed in a SLAM frontend $f_s \in \mathcal{F}_s$ according to its type (e.g. event-based, visual, visual-inertial) and then submitted to some

SLAM backend(s) $\mathcal{B}_s \in \mathcal{B}$ (e.g. for redundancy purposes), which can be operating in the discrete- or continuous-time realm. Lastly, each backend can provide feedback and information to some frontend and/or update properties of some devices. These considerations directly provide general guidelines for devising an intuitive modular system structure and inheritance hierarchy. In the interest of brevity, we, however, omit an in-depth description of the inner workings at this point.

3.7. Least Squares Problem Formulation

In analogy to [5, 6], we pose the CTSLAM problem in the backend as a NLLS optimization for the modified case of non-uniform cumulative B-Splines as presented in section 3.5. In the following, let $\hat{\mathcal{M}}_\mathcal{D}$ be the model-based collection of predicted measurements $\hat{m}^n_\mathcal{D}$ for a set of connected devices $\mathcal{D}$ and let $\hat{\mathcal{A}}$ denote the collection of estimated auxiliary states $\hat{a}^k_\mathcal{D}$. Furthermore, let $\mathcal{M}_\mathcal{D}$ be the set of observed measurements $m^n_\mathcal{D}$, each associated with its own timestamp. Hence, assuming a generative data model, we aim to minimize the accumulated residual error $f(\hat{\mathcal{M}}_\mathcal{D}(\hat{\mathcal{A}}_\mathcal{D}), \hat{\mathcal{A}})$, as in Eq. (15), with respect to $\hat{\mathcal{M}}_\mathcal{D}$ and $\hat{\mathcal{A}}$. Hence, in contrast to analogous discrete-time approaches, here, we aim to optimized the spline bases themselves instead of the conventional device poses observed at discrete times. Trivially, $\hat{\mathcal{A}}_d \subseteq \hat{\mathcal{A}}_\mathcal{D} \subseteq \hat{\mathcal{A}}$ must hold.

$$f^* = \operatorname{argmin} f(\hat{\mathcal{M}}_\mathcal{D}(\hat{\mathcal{A}}_\mathcal{D}), \hat{\mathcal{A}}) \tag{14}$$

$$\text{with} \quad f(\hat{\mathcal{M}}_\mathcal{D}(\hat{\mathcal{A}}_\mathcal{D}), \hat{\mathcal{A}}) = \sum_{d \in \mathcal{D}} \sum_{m^n_d \in \mathcal{M}_d} \|\bar{m}^n_d\|_{\Sigma_d}, \tag{15}$$

$$\text{where} \quad \bar{m}^n_d = r_d(\hat{m}^n_d(\hat{\mathcal{A}}_d), m^n_d) \tag{16}$$

Individual observation residuals $\bar{m}^n_d$ are weighted by the information matrix Σ_d according to their physical device specifications and $r_d(\cdot)$ denotes an arbitrary error measure between the estimates $\hat{m}^n_d$ and true observations m^n_d.

3.7.1 Camera Model

We define the distance measure $r_\mathcal{C}(\cdot)$ for cameras $\mathcal{C} \in \mathcal{D}$ to be the on-unit-sphere angular distance between the estimated $\hat{m}^n_\mathcal{C} \in \mathbb{R}^3$ and observed $m^n_\mathcal{C} \in \mathbb{R}^3$ landmark bearing vector as formalized in Eq. (18). Assuming known intrinsics, the set of auxiliary states $\hat{\mathcal{A}}_c$ for a camera $c \in \mathcal{C}$ consists of a subset of estimated homogeneous landmarks and the related spline bases at the occurrence time of m^n_c.

$$x = \langle \hat{m}^n_\mathcal{C}, m^n_\mathcal{C} \rangle, \quad y = \sqrt{1 - x^2} \tag{17}$$

$$r^n_\mathcal{C} := \angle(\hat{m}^n_\mathcal{C}, m^n_\mathcal{C}) = \operatorname{arctan}_2(y, x) \tag{18}$$

Contrary to classical approaches, where one would define $r_\mathcal{C}$ to be the distance $(\delta u, \delta v) \in \mathbb{R}^2$ on the normalized image plane, the aforementioned measure $r_\mathcal{C} \in \mathbb{R}$ is agnostic to the specific camera model. Furthermore, its associated cost, in contrast to the on-image plane distance, is directionally invariant under rotations, admits a global minimum as illustrated in Fig. (2), and even correctly handles landmarks that are observed behind the camera. The predicted camera measurements $\hat{m}^n_\mathcal{C}$ are computed by initially bringing the corresponding, estimated homogeneous landmarks $\hat{l}^k_w \in \hat{\mathcal{A}}_\mathcal{C}$ into the camera frame c, resulting in $\hat{l}^k_c \in \mathcal{A}_c$ and subsequent projection onto the unit-sphere. Assuming a static transformation T^s_c, this is done using:

$$\hat{m}^k_c = \hat{l}^k_c / \|\hat{l}^k_c\|, \text{ where } \hat{l}^k_c = T^c_s T^s_w(t_m) \hat{l}^k_w. \tag{19}$$

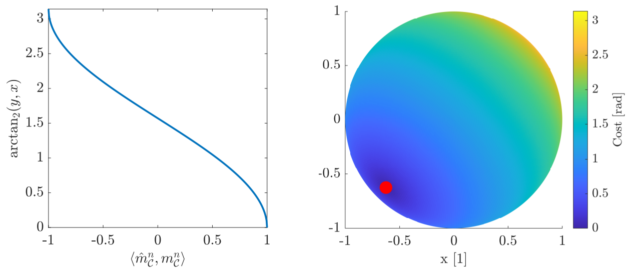


Figure 2: Left: Cost associated with distance measure $r_\mathcal{C}(\cdot)$ stated in Eq. (18). Right: Overlay of a unit sphere with measure function $r_\mathcal{C}(\cdot)$, where the red dot points to the direction of an observation with the lowest associated cost.

3.7.2 Inertial Measurement Unit Model

For future reference as well as for the sake of completeness, here, we also explicitly state the employed IMU model. In analogy to the previous section, one defines residuals $\bar{m}^n_\mathcal{I}$ for the inertial measurements $m^n_\mathcal{I} = [\omega, a]^n_\mathcal{I} \in \mathbb{R}^6$, where ω and a denote the instantaneous angular velocity and the linear acceleration respectively. Assuming a rigid sensor setup, the set of auxiliary states for IMUs contains their predicted biases $\hat{b}^\omega_\mathcal{I} \in \mathbb{R}^3$ and $\hat{b}^a_\mathcal{I} \in \mathbb{R}^3$ ($\in \mathcal{A}_\mathcal{I}$), the jointly estimated direction of gravity $\hat{g}_w \in \mathbb{R}^3$ ($\in \mathcal{A}$) and the relevant spline knots for the time of observation. Furthermore, by virtue of the underlying $\mathcal{C}^2$-continuity of cubic B-Splines, analytic expressions for the IMU predictions $\hat{m}^n_\mathcal{I} = [\hat{\omega}, \hat{a}]^n_\mathcal{I}$ are available in closed-form [15], which may then be compared to observed IMU measurements $m^n_\mathcal{I}$ via an adequately chosen distance measures $r_\mathcal{I}(\cdot)$, such as $r_\mathcal{I}(\hat{m}_\mathcal{I}, m_\mathcal{I}) = \hat{m}_\mathcal{I} - m_\mathcal{I}$, or a more elaborate one.

3.7.3 Other Sensor Models

As stated in preceding sections, formulating the optimization problem in continuous-time renders the problematic requirement for temporal alignment, or preintegration of measurements, obsolete. This, in turn, and irregardless of data rates, sensor heterogeneity and synchronization, directly allows a unified treatment of measurements within the optimization itself. Hence, integrating data stemming from unsychronized sensors, such as event-based cameras, LIDARs, (RTK-)GPS receivers or ultrawideband modules, becomes straight-forward by appropriately incorporating them into Eq. (14).

3.8. Analytic Derivatives

3.8.1 Temporal Spline Derivatives

We formulate the temporal spline derivatives in a recursive manner, as presented in [15], to support B-Splines of any degree k as well as reducing the implementational complexity. Thus, the angular $\omega_s(t) \in \mathbb{R}^3$ and translational velocity $v_s(t) \in \mathbb{R}^3$ at time $t \in [t_i, t_{i+1})$, expressed in the spline frame s, are evaluated by use of Eqs. (20)-(23). In Eq. (21), the rotational increment $\delta\omega_n \in \mathfrak{su}_2$ denotes a pure quaternion with zero scalar part $\widetilde{\delta\omega}_n$ and vector component $\overrightarrow{\delta\omega}_n$. This is obtained by taking the quaternion logarithm of δq_n as defined in Eq. (10).

$$\omega_s(t) = 2\omega_{i+k-1} \tag{20}$$

$$\omega_n = (\delta q_{n-1})^{-1}\,\omega_{n-1} + \dot{\lambda}^i_{n-1}\overrightarrow{\delta\omega}_{n-1}, \tag{21}$$

$$\text{with} \quad \omega_i = 0 \tag{22}$$

$$v_s(t) = q^s_w \sum_{j=i}^{i+k-1} \dot{\lambda}^i_{j-i}\,\delta x_j \tag{23}$$

Above, the coefficient λ^i_{i-j} represents a specific entry in the k-th degree, segment-dependent interpolation vector Λ^i_k. It is worth pointing out that the expression $\omega_s(t)$ in Eq. (20) is also equivalent to the real part of the pure unit quaternion $2q^s_w(t)\dot{q}^w_s(t)$. In analogy, the evaluation of the angular $\alpha_s(t)$ and translational acceleration $a_s(t)$ follows the same principle and $\alpha_s(t)$ is, mathematically speaking, the vector component of the expression $2q^s_w(t)\ddot{q}^s_w(t)$. The formulae for $\alpha_s(t)$ and $a_s(t)$ are provided below for some time $t \in [t_i, t_{i+1})$ within a spline segment:

$$\alpha_s(t) = 2\alpha_{i+k-1} \tag{24}$$

$$\alpha_n = (\delta q_{n-1})^{-1}\alpha_{n-1} + \left(\dot{\lambda}^i_{n-1}\omega_n \times + \ddot{\lambda}^i_{n-1}\right)\overrightarrow{\delta\omega}_{n-1}, \tag{25}$$

$$\text{with} \quad \alpha_i = 0 \tag{26}$$

$$a_s(t) = q^s_w \sum_{j=i}^{i+k-1} \ddot{\lambda}^i_{j-i}\delta x_j. \tag{27}$$

3.8.2 Temporal and Spacial Spline Base Derivatives

Temporal: In principle, derivatives with respect to base times t_m are computed by using Eqs. (20)-(22) under replacement of $\dot{\lambda}^i_{j-1}$ by its corresponding partial derivative $\partial_{t_m}\lambda^i_{j-1}(t)$. Obtaining the analytic Jacobians of the Λ^i_k matrix with respect to the temporal components of its bases t_m quickly becomes impractical for higher order splines. Fortunately, they may also be expressed recursively by exploiting properties of the k-th degree recurrence formula for mixing matrices Λ^i_k from [9]. We explicitly state them in Eqs. (28)-(32) below, where $S_k \in \mathbb{R}^{k\times k}$ is an upper triangular matrix containing ones.

$$\Lambda^i_k = [0, S_k]\,\tilde{M}^i_k U^i_k(t) \tag{28}$$

$$\text{with} \quad \tilde{M}^i_n = \Pi^i_{n-1}\tilde{M}^i_{n-1}, \quad \tilde{M}^i_0 = 1 \tag{29}$$

$$\text{and} \quad \Pi^i_{n-1} = \left[A^i_{n-1}, 0\right] + \left[0, B^i_{n-1}\right] \tag{30}$$

Hence, it holds that:

$$\partial_{t_m}\tilde{M}^i_n = \partial_{t_m}\Pi_{n-1}\tilde{M}^i_{n-1} + \Pi_{n-1}\partial_{t_m}\tilde{M}^i_{n-1} \tag{31}$$

$$\text{where} \quad \partial_{t_m}\tilde{M}^i_0 = 0. \tag{32}$$

A^i_{n-1} and B^i_{n-1} from above are defined as

$$A_{n-1} = \begin{bmatrix} 1-\alpha^i_0 & \cdots & 0 \\ \alpha^i_0 & \ddots & \vdots \\ 0 & \ddots & 1-\alpha^i_{n-1} \\ 0 & \cdots & \alpha^i_{n-1} \end{bmatrix} \tag{33}$$

$$B_{n-1} = \begin{bmatrix} -\beta^i_0 & \cdots & 0 \\ \beta^i_0 & \ddots & \vdots \\ 0 & \ddots & -\beta^i_{n-1} \\ 0 & \cdots & \beta^i_{n-1} \end{bmatrix} \tag{34}$$

$$\alpha^i_j = \frac{t_i - t_{i+j-n+1}}{t_{i+j+1} - t_{i+j-n+1}}, \quad \beta^i_j = \frac{t_{i+1} - t_i}{t_{i+j+1} - t_{i+j-n+1}}. \tag{35}$$

One may observe that the derivatives of α^i_j and β^i_j with respect to base time t_m can now automatically be evaluated analytically. Specifically, under consideration of the product rule, each term within α^i_j and β^i_j either becomes a Kronecker delta or vanishes, which then ultimately permits analytic evaluation of $\partial_{t_m}\tilde{M}^i_k$ and $\partial_{t_m}\Lambda^i_k$ as desired.

Angular: The rotational derivatives, expressed as a recurrence formula, are directly adopted from [15]. Hence, we forgo restating them at this point, which is largely due to the requirement for introduction of additional theoretical preliminaries.

Translational: The expression $x^w_s(t)$ in Eq. (11) only contains commutative and additive terms, thus, its derivatives with respect to the translational part of its bases are

$$\partial_{x_i} x_s^w(t) = (1 - \lambda_0^i)\mathbf{I}_3, \text{ where } j = 0 \tag{36}$$

$$\partial_{x_{i+j}} x_s^w(t) = (\lambda_{j-i}^i - \lambda_{j-i-1}^i)\mathbf{I}_3, \text{ with } 0 < j < k. \tag{37}$$

Other: Remaining derivatives, such as angular and linear velocities Jacobians with respect to base parameters can be derived in analogous fashion. For the sake of brevity we refer the reader to [15].

3.9. Stereo Triangulation

We incorporate a modified, bearing-vector-based version of the inverse depth weighted midpoint algorithm, an efficient and well suited triangulation method for most cases, from [21] to initially estimate landmark positions $\hat{l}_w^k$. In the following, let m_a and m_b be undistorted landmark bearing vectors as observed by camera a and b respectively. Under the assumption of static transformations T_a^s and T_b^s, an estimate for the landmark location $\hat{l}_w$ can be retrieved by use of Eqs. (38) through (40).

$$\hat{l}_w = T_a^w(t)\hat{l}_a = T_s^w(t)T_a^s\hat{l}_a \tag{38}$$

$$\hat{l}_a = \frac{\|\beta\|}{\|\beta\| + \|\gamma\|}\left[x_b^a + \frac{\|\gamma\|}{\|\alpha\|}(m_a + R_b^a m_b)\right] \tag{39}$$

$$\alpha = R_b^a m_b \times m_a,\ \beta = R_b^a m_b \times x_b^a,\ \gamma = m_a \times x_b^a \tag{40}$$

4. EXPERIMENTS

4.1. System Specifications

The proposed containers and framework were implemented in C++. They provide an extension to the Sophus library [22] and make use of Ceres [23] for optimization-related aspects. Furthermore, all experiments were carried out on a single CPU core (i7-8750H CPU @ 2.20 GHz) on a notebook running Ubuntu 20.04.

4.2. Simulation Setup

In order to outline the scaffolding of the proposed framework and involved, novel spline formulations in a monocular setup, we simulate trajectories in artificial environments of visual landmarks. Specifically, we randomly create n landmarks ($n = 30$ here), which are contained in the interior of a spherical shell with inner radius r_i and outer radius r_a around the initial starting location of the trajectory. Here, we use $r_i = 1\ m$ and $r_a = 10\ m$ as a realistic range for this shell. Then, we create random splines, from which we observe the constructed landmarks with a simulated monocular camera at $20\ Hz$ and create corresponding ground-truth measurements. We proceed by randomly perturbing the measured bearing vectors up to some maximum to account for detection noise. In particular, we select a maximum disturbance angle of $\pm 0.006\ rad$, which roughly corresponds to a ± 2 pixel error in the classical sense (assuming a sensor width of twice the focal length and reasonable pixel density). In the backend, we initialize the landmarks using rough guesses stemming from these heavily perturbed positions of the ground-truth landmarks.

4.3. Results

In the following, we present three different archetypes of simulated trajectories with different properties. These are illustrated in Fig. (3) along with their resulting translational and rotational errors with respect to time.

4.3.1 Trajectory 1: Aggressive Trajectory

Firstly, we consider an intentionally aggressive trajectory, as shown in Fig. (3, a), in order to evaluate the general stability and robustness of the optimization. For instance, the aforementioned trajectory covers an approximate distance of $30\ m$ within around $10\ s$, which can be considered to be a very rapid movement for aerial vehicles. Nonetheless, the achieved estimates remain accurate. Specifically, a translational RMSE of $0.0014\ m$, and a rotational RMSE of $0.0006\ rad$ are achieved.

4.3.2 Trajectory 2: Smooth Markov Trajectory

In the second setup, we evaluate the same metrics for the case of a Markov spline, whose change in velocity is limited to some amount within a given time span. This results in a more realistic and smother trajectory that resembles more typical trajectories. Similar to the previous setup, the proposed approach is shown to achieve good estimates and RMSE of $0.0038\ m$ and $0.0014\ rad$, respectively. Nonetheless, it is surprising to observe, that the more confined, aggressive Trajectory 1 results in a better estimate. This, however, can be attributed to the fact that the trajectory generally moves away from the simulated landmarks, which, in turn, may deteriorate their position estimates as well as yielding weaker constraints for the optimization. The covered distance in this setup is approximately $14\ m$.

4.3.3 Trajectory 3: Hybrid Trajectory

Lastly, we combine aspects of the previous two experiments. In particular, we combine a smooth Markov spline and overlay it with rapid movements. This results in a trajectory that, depending on the degree of perturbation, aims to capture improperly following a prescribed trajectory due to external disturbances. Here, we intentionally select a large disturbance to emphasise the stability of the optimization, which results in a RMSE of $0.0046\ m$, $0.00081\ rad$, respectively, over a distance of roughly $34\ m$.

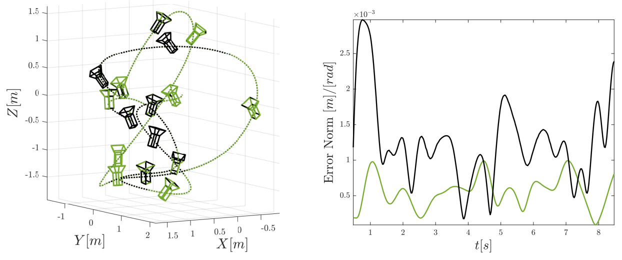
Established, state-of-the-art monocular discrete-time SLAM frameworks [19, 20] reportedly achieve translational

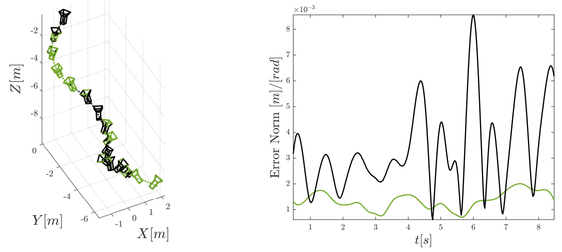
RMSEs anywhere between $0.004\ m$ and $0.119\ m$, depending on the presented experimental setup. In comparison, the estimates of the proposed *Hyper*SLAM framework in these proof-of-concept simulated experiments, indicate that exploiting the continuous-time properties of the proposed spline representation has great potential in effective trajectory estimation.

At this point, it is worth emphasizing, that the error plots in Fig. (3), do not consider some leading and trailing spline segments as they are notorious for not properly capturing the trajectory due to unequal amount of measurements compared to the remaining segments (i.e. they tend to be under-constrained). Throughout the previous setups, we did not yet allow for base times t_i to be optimized and instead, these were placed equidistantly every $0.5\ s$ for simplicity. This generally results in an optimization time of around $5\ s$ on a single core, which can be further optimized in the future.

5. CONCLUSIONS AND FUTURE WORK

In this work, we presented a way to universally extend uniform spline-based continuous-time representations to their non-uniform variants of any degree k, implemented appropriate container extensions to Sophus [22] and posed the underlying, modified NLLS optimization for the augmented case. Furthermore, we verified and demonstrated the efficacy, efficiency and applicability of the framework and the developed $\mathbb{SE}_3^t$ parametrizations for the case of simulated, monocular setup for different archetypes of trajectories. Future work will investigate the applicability of *Hyper*SLAM in real world experiments, explore the effect of dynamic trajectory refinement on performance and accuracy measures, as well as the expansion of the proposed pipeline to include other sensor modalities and configurations with the outlook of publicly releasing our code for the underlying non-uniform spline containers and the *Hyper*SLAM framework.


(a) Aggressive Trajectory


(b) Smooth Markov Trajectory

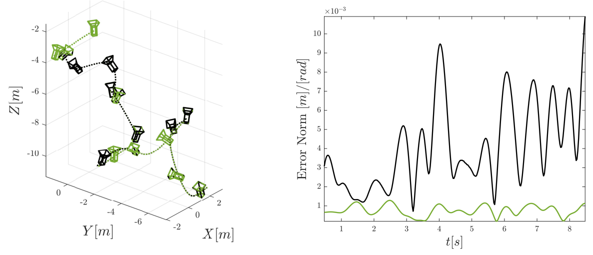

(c) Hybrid Trajectory

Figure 3: **Left Column:** Simulated, randomly generated ground-truth trajectories (in black) and their corresponding trajectory estimates after alignment (in green) following an optimization of the received messages in the backend. **Right Column:** Norm of the translation in meters (in black), and rotation errors in radians (in green) resulting from the trajectory estimates on the left. The rotation error is defined as the angle between the estimated and the ground-truth quaternions.

Acknowledgements

This research was supported by the Swiss National Science Foundation (SNSF, Agreement no. PP00P2157585), NCCR Robotics and the ETH Mobility Initiative together with the Swiss Federal Railways (SBB).

References

[1] P. Schmuck and M. Chli, "CCM-SLAM: Robust and efficient centralized collaborative monocular simultaneous localization and mapping for robotic teams," *Journal of Field Robotics*, vol. 36, no. 4, pp. 763–781, jun 2019. 1

[2] E. Eade, "Lie Groups for 2D and 3D Transformations," Tech. Rep., may 2017. [Online]. Available: http://www.ethaneade.org/lie.pdf 1, 3

[3] G. Bresson, Z. Alsayed, L. Yu, and S. Glaser, "Simultaneous Localization and Mapping: A Survey of Current Trends in Autonomous Driving," pp. 194–220, sep 2017. 1

[4] L. Teixeira, I. Alzugaray, and M. Chli, "Autonomous Aerial Inspection Using Visual-Inertial Robust Localization and Mapping," 2018, pp. 191–204. 1

[5] 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 *BMVC 2013 - Electronic Proceedings of the British Machine Vision Con-*

ference 2013. British Machine Vision Association, BMVA, 2013. 1, 2, 3, 5

[6] E. Mueggler, G. Gallego, and D. Scaramuzza, "Continuous-time trajectory estimation for event-based vision sensors," in *Robotics: Science and Systems*, vol. 11. MIT Press Journals, 2015. 1, 2, 5

[7] E. Mueggler, G. Gallego, H. Rebecq, and D. Scaramuzza, "Continuous-Time Visual-Inertial Odometry for Event Cameras," *IEEE Transactions on Robotics*, vol. 34, no. 6, pp. 1425–1440, dec 2018. 1, 2, 3

[8] A. Haarbach, T. Birdal, and S. Ilic, "Survey of higher order rigid body motion interpolation methods for keyframe animation and continuous-time trajectory estimation," in *Proceedings - 2018 International Conference on 3D Vision, 3DV 2018*. Institute of Electrical and Electronics Engineers Inc., oct 2018, pp. 381–389. 2, 3, 4

[9] K. Qin, "General matrix representations for B-splines," in *Proceedings - Pacific Conference on Computer Graphics and Applications*. IEEE Computer Society, 1998, pp. 37–43. 2, 4, 6

[10] P. Furgale, T. D. Barfoot, and G. Sibley, "Continuous-time batch estimation using temporal basis functions," in *Proceedings - IEEE International Conference on Robotics and Automation*. Institute of Electrical and Electronics Engineers Inc., 2012, pp. 2088–2095. 2, 3

[11] S. Anderson and T. D. Barfoot, "Towards relative continuous-time SLAM," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2013, pp. 1033–1040. 2

[12] S. Anderson, F. Dellaert, and T. D. Barfoot, "A hierarchical wavelet decomposition for continuous-time SLAM," in *Proceedings - IEEE International Conference on Robotics and Automation*. Institute of Electrical and Electronics Engineers Inc., sep 2014, pp. 373–380. 2

[13] D. Droeschel and S. Behnke, "Efficient continuous-time SLAM for 3D lidar-based online mapping," in *Proceedings - IEEE International Conference on Robotics and Automation*. Institute of Electrical and Electronics Engineers Inc., sep 2018, pp. 5000–5007. 2, 3

[14] Z. Zhang and D. Scaramuzza, "Rethinking Trajectory Evaluation for SLAM: a Probabilistic, Continuous-Time Approach," Tech. Rep., 2019. [Online]. Available: http://rpg.ifi.uzh.ch. 2

[15] C. Sommer, V. Usenko, D. Schubert, N. Demmel, and D. Cremers, "Efficient Derivative Computation for Cumulative B-Splines on Lie Groups." Institute of Electrical and Electronics Engineers (IEEE), aug 2020, pp. 11 145–11 153. 2, 4, 5, 6, 7

[16] D. Condurache and I. A. Ciureanu, "Closed Form of the Baker-Campbell-Hausdorff Formula for the Lie Algebra of Rigid Body Displacements," in *Computational Methods in Applied Sciences*. Springer, aug 2020, vol. 53, pp. 307–314. [Online]. Available: https://doi.org/10.1007/978-3-030-23132-3{_}37 3

[17] N. Dantam, "Quaternion Computation," Georgia Institute of Technology, Atlanta, Tech. Rep., oct 2014. [Online]. Available: http://www.neil.dantam.name/note/dantam-quaternion.pdf 3

[18] H. Ovrén and P. E. Forssén, "Trajectory representation and landmark projection for continuous-time structure from motion," *International Journal of Robotics Research*, vol. 38, no. 6, pp. 686–701, may 2019. 4

[19] T. Qin, P. Li, and S. Shen, "VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, aug 2018. 4, 7

[20] R. Mur-Artal and J. D. Tardos, "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, oct 2017. 4, 7

[21] S. H. Lee and J. Civera, "Triangulation: Why Optimize?" *30th British Machine Vision Conference 2019, BMVC 2019*, jul 2019. [Online]. Available: http://arxiv.org/abs/1907.11917 7

[22] H. Strasdat, "GitHub - strasdat/Sophus: C++ implementation of Lie Groups using Eigen." [Online]. Available: https://github.com/strasdat/Sophus 7, 8

[23] S. Agarwal and K. Mierle, "Ceres Solver — A Large Scale Non-linear Optimization Library." [Online]. Available: http://ceres-solver.org/ 7