This paper has been accepted for publication in *IEEE Robotics and Automation Letters*.

DOI: 10.1109/LRA.2022.3180427
IEEE *Xplore*: https://ieeexplore.ieee.org/document/9790325

Jacobian Computation for Cumulative B-Splines on SE(3) and Application to Continuous-Time Object Tracking

Javier Tirado[1] and Javier Civera[1]

***Abstract*—In this paper we propose a method that estimates the $SE(3)$ continuous trajectories (orientation and translation) of the dynamic rigid objects present in a scene, from multiple RGB-D views. Specifically, we fit the object trajectories to cumulative B-Splines curves, which allow us to interpolate, at any intermediate time stamp, not only their poses but also their linear and angular velocities and accelerations. Additionally, we derive in this work the analytical $SE(3)$ Jacobians needed by the optimization, being applicable to any other approach that uses this type of curves. To the best of our knowledge this is the first work that proposes 6-DoF continuous-time object tracking, which we endorse with significant computational cost reduction thanks to our analytical derivations. We evaluate our proposal in synthetic data and in a public benchmark, showing competitive results in localization and significant improvements in velocity estimation in comparison to discrete-time approaches.**

***Index Terms*—Computer Vision for Automation, Visual Tracking, Kinematics.**

I. INTRODUCTION

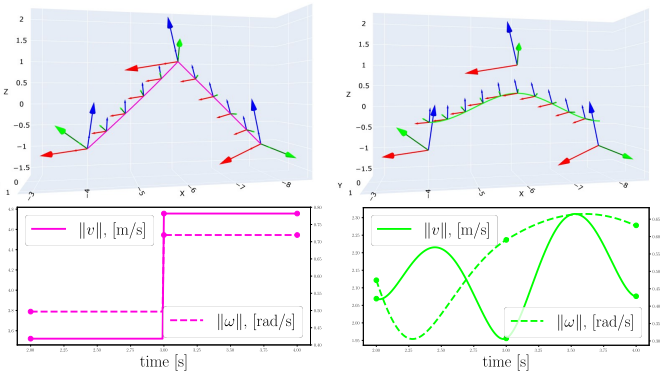
UNDERSTANDING the dynamic behavior of the moving elements present in a scene can become crucial in several robotic applications. Specifically, in SLAM [1] and SfM [2], it is well known that models unaware of the non-rigidity of the scene lead to poor performances regarding robustness and accuracy in localization and map reconstruction tasks [3]. Dynamic scenes are indeed acknowledged as a research challenge and several works have addressed such problem by developing systems that first detect, and subsequently remove from the map, the dynamic regions of an image [4], [5], [6].

On the other hand, the challenge of estimating the unconstrained motions of dynamic objects (instead of removing them) has recently gained attention. Acknowledging the kinematics of moving objects is expected to benefit the reasoning of the agents over the scene, increasing their capabilities for decision making. Moreover, this information can be used to enhance AR/VR experiences or serve as an alternative to expensive motion-capture set-ups. In this paper we focus on rigid objects with a free 6-degrees-of-freedom (6-DoF) motion.

Manuscript received: January, 25, 2022; Revised March, 23, 2022; Accepted May, 21, 2022. This paper was recommended for publication by Editor C. Cadena upon evaluation of the Associate Editor and Reviewers' comments. This work was funded in part by the Spanish Government under Grant PGC2018-096367-B-I00, and in part by the Aragon Government under Grant DGA FSE-T45 20R.

[1]Javier Tirado and Javier Civera are with I3A, Universidad de Zaragoza, Spain jtiradogarin@gmail.com, jcivera@unizar.es

Digital Object Identifier (DOI): see top of this page.



(a) Discrete-time (b) Continuous-time

Fig. 1: Example of linear and angular speed profiles and interpolated poses for discrete-time (a) and continuous-time (b) formulations. In discrete time, the common assumption of constant velocity, leads to discontinuities in time domain. As can be seen, this does not happen with the continuous-time formulation. We also ensure time continuity in acceleration.

With these goals, we find in the literature several approaches that aim to estimate the position and orientation of the objects ($SE(3)$ poses) [7], [8], [9]. However, relevant kinematic magnitudes such as the linear and angular velocities are not considered in their models. Thereby the estimates do not need to explicitly follow physically feasible motions. More recent works [10], [11] assume that kinematics between consecutive images are similar. Imposing this constraint, they are able to not only estimate the $SE(3)$ pose but also velocities [10] and accelerations [11].

All these works operate on discrete time, returning estimations at fixed time stamps. Thereby the estimated kinematics do not need to be continuous in time ($\mathbb{C}^2$ continuity), something that intuitively should happen in real object motions. A simple example is shown in Fig. 1. As long as the time between consecutive estimations is sufficiently small, discrete-time estimations might be accurate. However, this may not happen in real scenarios (e.g. due to temporal occlusions, low frame rates or fast dynamics).

In this work we address this by fitting the dynamic object trajectories to cubic cumulative B-Spline curves [12], a type of curve that in our proposal is defined by a series of $SE(3)$ control points distributed over time, which are interpolated to compute continuous poses. Such curve, has been previously used to estimate sensor ego-motion [13], but

its application to 6-DoF object tracking remains unexplored. Moreover, we show for the first time the $SE(3)$ Jacobians of the pose with respect to the control points, thus significantly reducing its associated computational cost. In summary, our contributions are: 1) A RGB-D system able to estimate the 6-DoF trajectories of objects present in a scene, their related angular and linear velocities and accelerations, presenting all of them continuity with respect to time, and 2) the analytical derivation of the Jacobians of the interpolated pose with respect to the $SE(3)$ control points of a cubic B-Spline curve, applicable to any work that uses this type of curve.

II. RELATED WORK

A. Object motion estimation

One of the first works that aimed to estimate the motion of dynamic entities in the scene was [14], in which the technique of *Factorization* was introduced. To this end, several assumptions were made. Only one object was expected to be present and also an orthographic camera model was used, thus simplifying the computations at the expense of not considering the perspective projection of a real camera [15]. Follow-up works tackled these limitations. In [16] the perspective camera model was introduced, and [17], [18] incorporated the motion estimation of multiple objects. Both aspects were addressed in [19]. However, the major part of these methods share some limitations: the difficulty of making them work sequentially, the need for specific motion assumptions, or high computational costs [3].

In the field of *SLAM* the first work that incorporated the motion estimation of objects in its pipeline was [20] (extended in [21]), in which a Bayesian framework was proposed. Their results showed improvements with respect to only estimating the localization of the sensor. The same conclusion is reached in [22], in which the 3D localization of the dynamic points are estimated. These works, as well as more recent ones [23], [24], [25], [26] are focused on objects whose movement is constrained to a plane, making them appropriate to environments like autonomous driving.

More recently, several works aim to estimate free 6-DoF motions of objects present in the scene. [8], [10] propose to first detect dynamic objects by using deep learning image segmentation techniques [27], to subsequently jointly estimate the object motions and the ego-motion of the sensor. In [9] a multi-level probabilistic association and a Conditional Random Field are added to ensure a correct data association between 3D points. Similarly [7], [11] propose a clustering approach based on the 3D motion of points to associate them to each object. In our work we follow a similar deep learning-based strategy using SiamMask [28].

B. Continuous-time tracking

For a general view of interpolation methods for tracking, we refer the reader to the thorough survey of Haarbach et al. [29]. In this work, we focus on cumulative B-Splines, which were originally defined in [12] for the graphics field. As highlighted in [30], B-Splines present interesting properties for robotics/vision tasks, which were leveraged for the first time in [13] to estimate the trajectory of a rolling shutter camera for calibration and visual-inertial SLAM. Later works extended their use for ego-motion estimation with other sensors: [31] for RGB-D, [32] for event cameras, [33] for 3D laser-range scanners and more recently in [34] for a multi-camera set-up. We also find applications in SfM [35], in which is also shown that $SE(3)$ B-Splines are preferred over the split $SO(3) \times \mathbb{R}^3$ representation when force and torque are related, which holds in general for rigid object motions [36].

In all previous works, the Jacobians were computed either with automatic differentiation (mainly the implementation of [37]) or with numerical differentiation. As noted by some authors [32], the analytical derivation of the Jacobians is a critical step to drastically reduce the execution time. Recently in [38] the Jacobians for the $SO(3)$ cumulative B-Splines were derived. In this work, we derive them for $SE(3)$.

Related to our work, [23] implements a continuous-time estimation of the object motions by using splines. However, a planar motion assumption ($SE(2)$) is made. To the best of our knowledge, our work is the first to apply continuous-time object tracking for 6-DoF.

III. BACKGROUND

A. SE(3) Lie Group

A reference frame $\{o\}$, that is attached to an object, can be expressed with respect to a global reference frame $\{w\}$ with a *transformation matrix* $\mathbf{T}_{wo} \in SE(3)$:

$$\mathbf{T}_{wo} = \begin{bmatrix} \mathbf{R}_{wo} & \mathbf{t}_{wo} \\ \mathbf{0}^\top & 1 \end{bmatrix}, \tag{1}$$

where $\mathbf{R}_{wo} \in SO(3)$ and $\mathbf{t} \in \mathbb{R}^3$ encode respectively, the orientation and translation of $\{o\}$ with respect to $\{w\}$. $SE(3)$ is both a group and a smooth manifold, implying that at each point, $\mathbf{T} \in SE(3)$, exists a unique tangent space called Lie Algebra or $se(3)$, which can be defined locally at $\mathbf{T}$, and at the identity $\mathbf{I}$ [39]. An element $\boldsymbol{\tau}^\wedge \in se(3)$ has the form:

$$\boldsymbol{\tau}^\wedge = \begin{bmatrix} \mathbf{v} \\ \boldsymbol{\omega} \end{bmatrix}^\wedge = \begin{bmatrix} \boldsymbol{\omega}^\wedge & \mathbf{v} \\ \mathbf{0}^\top & 0 \end{bmatrix}, \tag{2}$$

where $\mathbf{v} \in \mathbb{R}^3$ and $\boldsymbol{\omega}^\wedge$ is the anti-symmetric matrix related to $\boldsymbol{\omega} \in \mathbb{R}^3$. $(\cdot)^\wedge$ is the *hat* operator, used for a convenient vectorization. An element $\boldsymbol{\tau}^\wedge \in se(3)$ is mapped to $SE(3)$ and vice versa via the *exponential* and *logarithm* mappings:

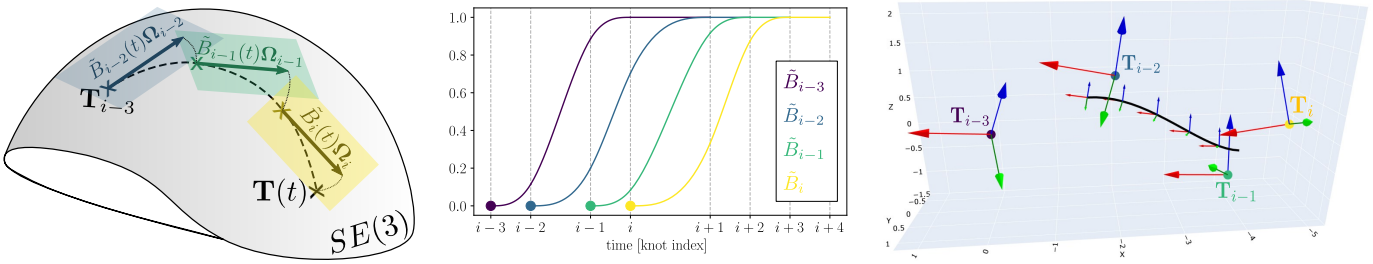
$$\text{Exp} \quad : \quad \mathbb{R}^6 \quad \mapsto SE(3) \quad ; \quad \boldsymbol{\tau} \mapsto \mathbf{T} = \text{Exp}(\boldsymbol{\tau}), \tag{3}$$

$$\text{Log} \quad : \quad SE(3) \mapsto \mathbb{R}^6 \quad ; \quad \mathbf{T} \mapsto \boldsymbol{\tau} = \text{Log}(\mathbf{T}), \tag{4}$$

where we have used the capitalized notation of [39].

This way, we can compose a transformation matrix $\mathbf{T}_{wo}$ with another parameterized in the local tangent space: $\mathbf{T}_{wo}\,\text{Exp}(\boldsymbol{\tau}_o)$, or in the tangent space defined at the identity: $\text{Exp}(\boldsymbol{\tau}_w)\mathbf{T}_{wo}$. The equivalence between the two is given by the *Adjoint matrix* $\text{Ad}_{\mathbf{T}_{wo}} \in \mathbb{R}^{6\times6}$: $\boldsymbol{\tau}_w = \text{Ad}_{\mathbf{T}_{wo}}\boldsymbol{\tau}_o$. As a result (used in Sec. IV-B), we have that:

$$\text{Exp}(\boldsymbol{\tau}_w)\mathbf{T}_{wo} = \mathbf{T}_{wo}\text{Exp}(\text{Ad}_{\mathbf{T}_{wo}^{-1}}\boldsymbol{\tau}_w) = \mathbf{T}_{wo}\text{Exp}(\boldsymbol{\tau}_o). \tag{5}$$



(a) (b) (c)

Fig. 2: Interpolation of a pose $\mathbf{T}(t) \in SE(3)$ with $t \in [t_i, t_{i+1})$ and control points $\{\mathbf{T}_{i-3} .. \mathbf{T}_i\}$. (a) Visualization of Eq. 12 in the $SE(3)$ manifold. The successive compositions, $\text{Exp}(\tilde{B}_j(t)\mathbf{\Omega}_j)$ with $j \in \{i-3 .. i\}$, are parameterized over the successive local tangent spaces. (b) Cumulative basis functions with influence at $t \in [t_i, t_{i+1})$. (c) 4 exemplar control points (bigger reference frames) and resultant interpolation at this time span. For clarity, only a few interpolated frames are shown.

The elements of $se(3)$ can also be related to the kinematics of the coordinate system $\{o\}$ [36]. Using Newton's notation for differentiation with respect to time:

$$\boldsymbol{\tau}_o^\wedge = \begin{bmatrix} \mathbf{v}_o \\ \boldsymbol{\omega}_o \end{bmatrix}^\wedge = \mathbf{T}_{wo}^{-1}\dot{\mathbf{T}}_{wo} = \begin{bmatrix} \mathbf{R}_{wo}^\top \dot{\mathbf{R}}_{wo} & \mathbf{R}_{wo}^\top \dot{\mathbf{t}}_{wo} \\ \mathbf{0}^\top & 0 \end{bmatrix}, \quad (6)$$

contains the linear $\mathbf{v}_o$ and angular $\boldsymbol{\omega}_o$ velocities of $\{o\}$ expressed in a coordinate system that is fixed and instantaneously coincident with $\{o\}$. The linear and angular accelerations are obtained time-differentiating again Eq. 6. These quantities can be transformed to $\{w\}$ via $\mathbf{R}_{wo}$.

Because of its importance in the Jacobian derivations (Sec. IV-B), we introduce the *left Jacobian* $\mathbf{J}_l$ of $SE(3)$ [39]:

$$\mathbf{J}_l(\boldsymbol{\tau}) = \left. \frac{\partial \text{Log}(\text{Exp}(\boldsymbol{\tau} + \delta\boldsymbol{\tau})\text{Exp}(\boldsymbol{\tau})^{-1})}{\partial \delta\boldsymbol{\tau}} \right|_{\delta\boldsymbol{\tau}=\mathbf{0}}. \quad (7)$$

It maps variations of $\boldsymbol{\tau}$ to variations expressed in the tangent space at the identity and composed with the current pose. Two results that will be used are that, for small $\boldsymbol{\xi} \in \mathbb{R}^6$:

$$\text{Exp}(\boldsymbol{\tau} + \boldsymbol{\xi}) \approx \text{Exp}(\mathbf{J}_l(\boldsymbol{\tau})\boldsymbol{\xi})\text{Exp}(\boldsymbol{\tau}), \quad (8)$$

$$\text{Log}(\text{Exp}(\boldsymbol{\xi})\text{Exp}(\boldsymbol{\tau})) \approx \boldsymbol{\tau} + \mathbf{J}_l^{-1}(\boldsymbol{\tau})\boldsymbol{\xi}. \quad (9)$$

Closed-form expressions for: $\text{Exp}(\boldsymbol{\tau}), \text{Log}(\mathbf{T}), \text{Ad}_\mathbf{T}$ and $\mathbf{J}_l(\boldsymbol{\tau})$, with $\boldsymbol{\tau}^\wedge \in se(3), \mathbf{T} \in SE(3)$, can be found in [40].

B. Cumulative B-Splines

A pose $\mathbf{T}(t) \in SE(3)$ at time t, interpolated with a cumulative B-Spline of $n+1$ control points is given by [13]:

$$\mathbf{T}(t) = \text{Exp}(\tilde{B}_{0,k}(t)\text{Log}(\mathbf{T}_0)) \prod_{i=1}^{n} \text{Exp}(\tilde{B}_{i,k}(t)\mathbf{\Omega}_i), \quad (10)$$

where $\tilde{B}_{i,k}(t)$ is the $i-th$ scalar *cumulative basis* function, a $\mathbb{C}^{k-2}$ continuous polynomial of degree $k-1$. Each one is related to a time stamp t_i (*knot*), satisfying:

$$\tilde{B}_{i,k}(t) = \begin{cases} 0 & \text{if } t \leq t_i \\ \sum_{j=i}^{i+k} B_{j,k}(t) & \text{if } t \in (t_i, t_{i+k-1}) \\ 1 & \text{if } t \geq t_{i+k-1} \end{cases} \quad (11)$$

where the terms $B_{j,k}(t)$ are the standard B-Spline basis functions obtained with the de Boor-Cox formula [41]. Eq. 11 conditions are visualized in Fig. 2b.

For $i > 0$ each cumulative basis weighs the relative difference between the *control points* $\mathbf{T}_{i-1}, \mathbf{T}_i \in SE(3)$ i.e. $\mathbf{\Omega}_i = \text{Log}(\mathbf{T}_{i-1}^{-1}\mathbf{T}_i)$. These control points are the variables to be estimated. Since we are interested in a curve with $\mathbb{C}^2$ continuity, $k = 4$ (*cubic* cumulative B-Spline) is chosen, thereby $\tilde{B}_{i,4}(t) = 1$ if $t \geq t_{i+3}$ (Eq. 11). Using this fact, Eq. 10 for $t \in [t_i, t_{i+1})$ can be simplified to[1]:

$$\mathbf{T}(t) = \mathbf{T}_{i-3} \prod_{j=1}^{3} \text{Exp}(\tilde{B}_{i-3+j}(t)\mathbf{\Omega}_{i-3+j}). \quad (12)$$

Time derivatives $\dot{\mathbf{T}}$ and $\ddot{\mathbf{T}}$, are given in [13], [38]. Fig. 2a shows a visual conceptualization of Eq. 12 in the manifold of $SE(3)$. Additionally, in Fig. 2c we show 4 exemplar control points and the resultant interpolation using the cumulative basis functions of Fig. 2b.

In our implementation, to compute the value of a cumulative basis function at time $t \in [t_i, t_{i+1})$, we combine the cumulative definition of Eq. 11 with the matrix representation of the standard basis functions derived in [42]:

$$\tilde{\mathbf{B}} = \begin{bmatrix} \tilde{B}_{i-3}(t) & \tilde{B}_{i-2}(t) & \tilde{B}_{i-1}(t) & \tilde{B}_i(t) \end{bmatrix} = \mathbf{u}^\top \tilde{\mathbf{M}},$$
$$\tilde{\mathbf{M}} = \begin{bmatrix} 1 & 1-m_{00} & m_{02} & 0 \\ 0 & 3m_{00} & m_{12} & 0 \\ 0 & -3m_{00} & m_{22} & 0 \\ 0 & m_{00} & m_{32}+m_{33} & m_{33} \end{bmatrix}, \quad (13)$$

where $\mathbf{u} = \begin{bmatrix} 1 & u & u^2 & u^3 \end{bmatrix}^\top$, with $u = \frac{t-t_i}{t_{i+1}-t_i} \in [0, 1)$. The terms m_{ij} correspond to the ones defined in [42, Sec. 3.2]. $\tilde{\mathbf{M}}$ is the cumulative reformulation without assuming the common constraint [13], [32] of constant time intervals between control points. This can benefit applications that require more flexibility in their placement.

IV. CONTINUOUS-TIME OBJECT TRACKING

At each time stamp t, our approach receives as inputs a RGB-D image, its estimated pose $\mathbf{T}_{wc} \in SE(3)$, and instance segmentation masks without inter-image associations (we used COLMAP [2] and SiamMask [28] in our experiments). The outputs are the continuous-time trajectories $\mathbf{T}_{wo}(t)$ of each observed object, parameterized with the control points of a cumulative B-Spline (Sec. III-B). Since our formulation is

[1]From now on, we drop out the subscript $k = 4$ to avoid clutter.

independent for each object, we use one common subscript. An additional optional output is the set of optimized 3D objects' points in the object frame $\{o\}$. Two main blocks conform the proposal: the front-end (Sec. IV-D) and the optimization back-end (Sec. IV-A). The front-end is in charge of 1) object correspondences between frames, 2) initialization of new trajectories, and 3) providing enough object feature tracks to optimize its trajectory. Special care is taken in the filtering of outliers. The back-end receives this information and optimizes the set of control points, $\mathcal{T}$, of each object's continuous-time trajectory and optionally a set of sparse objects' points $\mathcal{P}$. To this end, a robustified Gauss-Newton algorithm is used.

A. Optimization

To avoid an unconstrained increase in computational cost, we adopt a sliding window optimization. The set of 3D object point observations $\mathbf{p}_c \in \mathbb{R}^3$ expressed in $\{c\}$ within this temporal window is denoted by $\mathcal{Z}$. We adopt an object-centric parameterization [10], i.e. we relate each observation $\mathbf{p}_c \in \mathcal{Z}$ to its correspondent point $\mathbf{p}_o \in \mathcal{P}$ in the object frame $\{o\}$. If an object point $\mathbf{p}_o$ is observed in different images, multiple observations in $\mathcal{Z}$ are related to it.

We define the estimated 3D location error, $\mathbf{e}_{\mathbf{p}_c} \in \mathcal{E}$ (with $\mathcal{E}$ as the set of errors within the temporal window) as:

$$\mathbf{e}_{\mathbf{p}_c} = \mathbf{p}_c - \text{proj}\left(\mathbf{T}_{wc}^{-1}\,\mathbf{T}_{wo}(t)\,\tilde{\mathbf{p}}_o\right), \tag{14}$$

where $\tilde{\mathbf{p}}_o \in \mathbb{P}^3$ is the homogeneous representation of $\mathbf{p}_o$, $\mathbf{T}_{wo}(t)$ is the interpolated pose (with Eq. 12) at the time stamp t at which $\mathbf{p}_c$ was observed from a camera with estimated pose $\mathbf{T}_{wc}$. proj : $\mathbb{P}^3 \mapsto \mathbb{R}^3$ performs the homogeneous to Cartesian coordinates mapping.

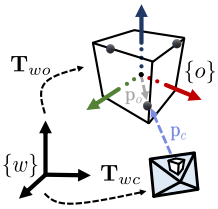


Fig. 3: Notation.

We denote as $\mathcal{X}$ the set of parameters that influences $\mathcal{E}$. If only the control points of the trajectories are optimized, then $\mathcal{X} \equiv \mathcal{T}$. If the objects' points are also optimized then $\mathcal{X} \equiv \{\mathcal{T}, \mathcal{P}\}$. We refer to these optimizations as *Spline BA* and *Local BA*. Then we seek to minimize $E(\mathcal{X})$:

$$E(\mathcal{X}) = \frac{1}{2}\sum_{\mathbf{e}_{\mathbf{p}_c}\in\mathcal{E}} \rho\left(\mathbf{e}_{\mathbf{p}_c}^\top \boldsymbol{\Sigma}_{\mathbf{p}_c}^{-1}\mathbf{e}_{\mathbf{p}_c}\right). \tag{15}$$

Assuming observations to be independent and perturbed with zero-mean Gaussian noise with covariance $\boldsymbol{\Sigma}_{\mathbf{p}_c}$ (without prior knowledge, we set it to the identity), minimizing Eq. 15 leads to maximizing the likelihood $\mathcal{L}(\mathcal{X}|\mathcal{Z})$ [43]. The Huber loss $\rho(\cdot)$ [44] is used to reduce the influence of outliers.

To minimize Eq. 15 we iteratively perform updates $\Delta\mathbf{x}$ on the parameters $\mathbf{x}$ (vectorized $\mathcal{X}$) by solving the normal equations: $\mathbf{H}\Delta\mathbf{x} = -\mathbf{g}$, where $\mathbf{H}$ and $\mathbf{g}$ are the (approximate) Hessian and gradient of $\mathbf{e}$ (vectorized $\mathcal{E}$) with respect to $\mathbf{x}$. For each observation they are computed as:

$$\mathbf{H}_{\mathbf{e}_{\mathbf{p}_c}} = \rho'\mathbf{J}_{\mathbf{e}_{\mathbf{p}_c}}^\top \boldsymbol{\Sigma}_{\mathbf{p}_c}^{-1}\mathbf{J}_{\mathbf{e}_{\mathbf{p}_c}}, \quad \mathbf{g}_{\mathbf{e}_{\mathbf{p}_c}} = \rho'\mathbf{e}_{\mathbf{p}_c}^\top \boldsymbol{\Sigma}_{\mathbf{p}_c}^{-1}\mathbf{J}_{\mathbf{e}_{\mathbf{p}_c}}, \tag{16}$$

where ρ' is the Huber loss derivative at $\mathbf{e}_{\mathbf{p}_c}^\top \boldsymbol{\Sigma}_{\mathbf{p}_c}^{-1}\mathbf{e}_{\mathbf{p}_c}$, and $\mathbf{J}_{\mathbf{e}_{\mathbf{p}_c}} = \partial\mathbf{e}_{\mathbf{p}_c}/\partial\mathbf{x}$ is the Jacobian of the observation error.

B. Jacobians Derivation

To compute each $\mathbf{J}_{\mathbf{e}_{\mathbf{p}_c}}$ we need to differentiate $\mathbf{e}_{\mathbf{p}_c}$ with respect to the control points. To this end, as it is common when dealing with $SE(3)$ poses [40], we parameterize each control point $\mathbf{T}_i$ with a perturbation $\boldsymbol{\xi}_i^\wedge \in se(3)$, so that:

$$\left.\frac{\partial\mathbf{e}_{\mathbf{p}_c}(\text{Exp}(\boldsymbol{\xi}_i)\mathbf{T}_i)}{\partial\boldsymbol{\xi}_i}\right|_{\boldsymbol{\xi}_i=\mathbf{0}} = \frac{\partial\mathbf{e}_{\mathbf{p}_c}(\text{Exp}(\boldsymbol{\xi}_i)\mathbf{T}_i)}{\partial\boldsymbol{\xi}_i}(\mathbf{0}), \tag{17}$$

is used to update $\mathbf{T}_i \leftarrow \text{Exp}(\boldsymbol{\xi}_i)\mathbf{T}_i$ with the value of $\boldsymbol{\xi}_i \in \mathcal{X}$ computed at each iteration of the optimization process.

It is of special interest the derivative of an interpolated pose $\mathbf{T}(t)$ (Eq. 12) w.r.t. the control points as it has not been addressed yet in the literature and would imply significant computational savings [32]. In this section we derive them for its $se(3)$ form and for its 12-dimensional vectorized form of the $SE(3)$ object[2]. The notation used is the following (for convenience, we particularize Eq. 12 with $i = 3$):

$$\mathbf{T}(t) = \text{Exp}(\mathbf{a}_0)\mathbf{T}_0\mathbf{A}_1(t)\mathbf{A}_2(t)\mathbf{A}_3(t), \tag{18}$$
$$\mathbf{A}_j(t) = \text{Exp}(\mathbf{a}_j(t)), \tag{19}$$
$$\mathbf{a}_j(t) = \tilde{B}_j(t)\boldsymbol{\Omega}_j, \tag{20}$$
$$\mathbf{a}_0 = \boldsymbol{\xi}_0, \tag{21}$$
$$\boldsymbol{\Omega}_j = \text{Log}\left((\text{Exp}(\boldsymbol{\xi}_{j-1})\mathbf{T}_{j-1})^{-1}\text{Exp}(\boldsymbol{\xi}_j)\mathbf{T}_j\right), \tag{22}$$

with $j \in \{1, 2, 3\}$ and the perturbations evaluated at $\mathbf{0}$.

Focusing first on the 12-d vectorized form, $\mathbf{T}_{\text{vec}}$ (see Eq. 24), and using the multi-variable chain rule, we have:

$$\boldsymbol{\xi}_{j\in\{0,1,2\}} : \frac{\partial\mathbf{T}_{\text{vec}}}{\partial\boldsymbol{\xi}_j}(\mathbf{0}) = \frac{\partial\mathbf{T}_{\text{vec}}}{\partial\mathbf{a}_j}\frac{\partial\mathbf{a}_j}{\partial\boldsymbol{\xi}_j}(\mathbf{0}) + \frac{\partial\mathbf{T}_{\text{vec}}}{\partial\mathbf{a}_{j+1}}\frac{\partial\mathbf{a}_{j+1}}{\partial\boldsymbol{\xi}_j}(\mathbf{0}),$$
$$\boldsymbol{\xi}_3 : \frac{\partial\mathbf{T}_{\text{vec}}}{\partial\boldsymbol{\xi}_3}(\mathbf{0}) = \frac{\partial\mathbf{T}_{\text{vec}}}{\partial\mathbf{a}_3}\frac{\partial\mathbf{a}_3}{\partial\boldsymbol{\xi}_3}(\mathbf{0}), \tag{23}$$

where $\mathbf{T}_{\text{vec}}$ vectorizes $\mathbf{T}$ to represent it as a 1D vector:

$$\mathbf{T}_{\text{vec}} = \begin{bmatrix}(\mathbf{R}^{c1})^\top & (\mathbf{R}^{c2})^\top & (\mathbf{R}^{c3})^\top & \mathbf{t}^\top\end{bmatrix}^\top, \tag{24}$$

with $\mathbf{R}^{ci}$ as the $i-th$ column of $\mathbf{R}$ (rotation in $\mathbf{T}$). The last row is ignored since it would add meaningless computations. With these considerations (and definitions of Eqs. 28-31):

$$\frac{\partial\mathbf{T}_{\text{vec}}}{\partial\mathbf{a}_j} = \frac{\partial(\mathbf{P}_j\text{Exp}(\mathbf{a}_j+\boldsymbol{\tau}_j)\mathbf{N}_j)_{\text{vec}}}{\partial\boldsymbol{\tau}_j}(\mathbf{0}), \tag{25}$$
$$\overset{8}{=} \frac{\partial}{\partial\boldsymbol{\tau}_j}(\mathbf{P}_j\underbrace{\text{Exp}(\mathbf{J}_l(\mathbf{a}_j)\boldsymbol{\tau}_j)}_{\mathbf{C}(\boldsymbol{\tau}_j)}\underbrace{\text{Exp}(\mathbf{a}_j)\mathbf{N}_j}_{\mathbf{N}'_j})_{\text{vec}}(\mathbf{0}), \tag{26}$$
$$= \left.\frac{\partial(\mathbf{P}_j\mathbf{C}(\boldsymbol{\tau}_j)\mathbf{N}'_j)_{\text{vec}}}{\partial\mathbf{C}(\boldsymbol{\tau}_j)_{\text{vec}}}\frac{\partial\mathbf{C}(\boldsymbol{\tau}_j)_{\text{vec}}}{\partial\mathbf{J}_l(\mathbf{a}_j)\boldsymbol{\tau}_j}\frac{\partial\mathbf{J}_l(\mathbf{a}_j)\boldsymbol{\tau}_j}{\partial\boldsymbol{\tau}_j}\right|_{\boldsymbol{\tau}_j=\mathbf{0}}, \tag{27}$$

with all the derivatives of Eq. 27 evaluated at $\boldsymbol{\tau}_j = \mathbf{0}$, and:

$$j=0 \rightarrow \mathbf{P}_0 = \mathbf{I}_{4\times4}, \quad \mathbf{N}'_0 = \mathbf{T}, \tag{28}$$
$$j=1 \rightarrow \mathbf{P}_1 = \mathbf{T}_0, \quad \mathbf{N}'_1 = \mathbf{A}_1\mathbf{A}_2\mathbf{A}_3, \tag{29}$$
$$j=2 \rightarrow \mathbf{P}_2 = \mathbf{T}_0\mathbf{A}_1, \quad \mathbf{N}'_2 = \mathbf{A}_2\mathbf{A}_3, \tag{30}$$
$$j=3 \rightarrow \mathbf{P}_3 = \mathbf{T}_0\mathbf{A}_1\mathbf{A}_2, \quad \mathbf{N}'_3 = \mathbf{A}_3. \tag{31}$$

[2]Both forms can fit a wide range of cost functions via the chain rule. In Sec. V-A we show their benefits in our continuous-time tracking problem.

The right-most term of Eq. 27 is straightforward:

$$\frac{\partial \mathbf{J}_l(\mathbf{a}_j)\boldsymbol{\tau}_j}{\partial \boldsymbol{\tau}_j}(\mathbf{0}) = \mathbf{J}_l(\mathbf{a}_j), \tag{32}$$

Note that $\mathbf{J}_l(\mathbf{a}_0)|_{\boldsymbol{\xi}_0=\mathbf{0}} = \mathbf{I}_{6\times 6}$. For the left-most term of Eq. 27, denoting the Kronecker product as $\otimes$, and the rotation matrix of $\mathbf{P}_j$ as $\mathbf{R}_{\mathbf{P}_j}$, from [45, Eq. 11-12]:

$$\frac{\partial (\mathbf{P}_j \mathbf{C}(\boldsymbol{\tau}_j)\mathbf{N}'_j)_{\text{vec}}}{\partial \mathbf{C}(\boldsymbol{\tau}_j)_{\text{vec}}}(\mathbf{0}) = (\mathbf{N}'_j)^\top \otimes \mathbf{R}_{\mathbf{P}_j}, \tag{33}$$

The middle term of Eq. 27 is given by the generators of $SE(3)$, $\{\mathbf{G}_i\}_{i=1}^6$ [46, Eq. A.1], since they map, at the identity, infinitesimal variations in the dimensions of an element $\boldsymbol{\tau}^\wedge \in se(3)$ to variations in $SE(3)$:

$$\frac{\partial \text{Exp}(\mathbf{J}_l(\mathbf{a}_j)\boldsymbol{\tau}_j)_{\text{vec}}}{\partial \mathbf{J}_l(\mathbf{a}_j)\boldsymbol{\tau}_j}(\mathbf{0}) = \begin{bmatrix}(\mathbf{G}_1)_{\text{vec}} & \dots & (\mathbf{G}_6)_{\text{vec}}\end{bmatrix} = \mathbf{G}, \tag{34}$$

where $(\mathbf{G}_i)_{\text{vec}}$ indicates (with slight abuse of notation) the same vectorization as in Eq. 24, i.e., $\mathbf{G}$ is a 12×6 matrix.

It only remains to derive $\partial \mathbf{a}_j / \partial \boldsymbol{\xi}_j$ and $\partial \mathbf{a}_{j+1} / \partial \boldsymbol{\xi}_j$. A useful observation is that, for $j \in \{1, 2, 3\}$:

$$(\text{Exp}(\boldsymbol{\xi}_{j-1})\mathbf{T}_{j-1})^{-1} = \mathbf{T}_{j-1}^{-1}\text{Exp}(-\boldsymbol{\xi}_{j-1}), \tag{35}$$

so we only need to derive $(\partial \mathbf{a}_j / \partial \boldsymbol{\xi}_j)|_{\boldsymbol{\xi}_j=\mathbf{0}}$, since:

$$\frac{\partial \boldsymbol{\Omega}_j}{\partial \boldsymbol{\xi}_{j-1}} = -\frac{\partial \boldsymbol{\Omega}_j}{\partial \boldsymbol{\xi}_j}, \qquad \frac{\partial \mathbf{a}_j}{\partial \boldsymbol{\xi}_{j-1}} = -\frac{\partial \mathbf{a}_j}{\partial \boldsymbol{\xi}_j}, \tag{36}$$

which can be obtained as follows:

$$\frac{\partial \mathbf{a}_j}{\partial \boldsymbol{\xi}_j}(\mathbf{0}) = \frac{\partial \tilde{B}_j(t)\text{Log}(\mathbf{T}_{j-1}^{-1}\text{Exp}(\boldsymbol{\xi}_j)\mathbf{T}_j)}{\partial \boldsymbol{\xi}_j}(\mathbf{0}), \tag{37}$$

$$\overset{5}{=} \frac{\partial \tilde{B}_j(t)\text{Log}(\text{Exp}(\text{Ad}_{\mathbf{T}_{j-1}^{-1}}\boldsymbol{\xi}_j)\mathbf{T}_{j-1}^{-1}\mathbf{T}_j)}{\partial \boldsymbol{\xi}_j}(\mathbf{0}), \tag{38}$$

$$\overset{9}{=} \frac{\partial \tilde{B}_j(t)\mathbf{J}_l^{-1}(\text{Log}(\mathbf{T}_{j-1}^{-1}\mathbf{T}_j))\text{Ad}_{\mathbf{T}_{j-1}^{-1}}\boldsymbol{\xi}_j}{\partial \boldsymbol{\xi}_j}(\mathbf{0}), \tag{39}$$

$$= \tilde{B}_j(t)\ \mathbf{J}_l^{-1}(\text{Log}(\mathbf{T}_{j-1}^{-1}\mathbf{T}_j))\text{Ad}_{\mathbf{T}_{j-1}^{-1}}. \tag{40}$$

Lastly, $(\partial \mathbf{a}_0 / \partial \boldsymbol{\xi}_0)(\mathbf{0}) = \mathbf{I}_{6\times 6}$ (from its definition at Eq. 21). Note that both Eq. 26 and 39 are exact since they are evaluated at $\mathbf{0}$. Hence, only the first-order term has influence.

Now, we focus on the Jacobian of the minimal $se(3)$ representation, $\text{Log}(\mathbf{T})$, w.r.t. the perturbations. Starting off Eq. 23, $\frac{\partial \mathbf{T}_{\text{vec}}}{\partial \mathbf{a}_k}$ ($k \in \{0\,..\,3\}$) is the only term that we need to change in favor of $\frac{\partial \text{Log}(\mathbf{T})}{\partial \mathbf{a}_k}$. For $\mathbf{a_0}$ it can be obtained as:

$$\frac{\partial \text{Log}(\mathbf{T})}{\partial \mathbf{a}_0} = \frac{\partial \text{Log}(\text{Exp}(\mathbf{a}_0)\mathbf{T})}{\partial \mathbf{a}_0} \overset{9}{=} \mathbf{J}_l^{-1}(\text{Log}(\mathbf{T})). \tag{41}$$

Since $\mathbf{a}_0 = \boldsymbol{\xi}_0$ evaluated at $\mathbf{0}$. Lastly, for $k = \{1, 2, 3\}$:

$$\frac{\partial \text{Log}(\mathbf{T})}{\partial \mathbf{a}_k} = \frac{\partial \text{Log}(\mathbf{P}_k\text{Exp}(\mathbf{a}_k + \boldsymbol{\tau}_k)\mathbf{N}'_{k+1})}{\partial \boldsymbol{\tau}_k}(\mathbf{0}), \tag{42}$$

$$\overset{5,8}{=} \frac{\partial \text{Log}(\text{Exp}(\text{Ad}_{\mathbf{P}_k}\mathbf{J}_l(\mathbf{a}_k)\boldsymbol{\tau}_k)\mathbf{T})}{\partial \boldsymbol{\tau}_k}(\mathbf{0}), \tag{43}$$

$$\overset{9}{=} \mathbf{J}_l^{-1}(\text{Log}(\mathbf{T}))\text{Ad}_{\mathbf{P}_k}\mathbf{J}_l(\mathbf{a}_k), \tag{44}$$

which is similar to [38, Eq. 57] but without imposing the adjoint of SO(3). To sum up, the Jacobians of both representations are completely defined by (with $k \in \{0\,..\,3\}$):

$$\frac{\partial \mathbf{a}_k}{\partial \boldsymbol{\xi}_k}(\mathbf{0}) = \tilde{B}_k(t)\ \mathbf{J}_l^{-1}(\text{Log}(\mathbf{T}_{k-1}^{-1}\mathbf{T}_k))\text{Ad}_{\mathbf{T}_{k-1}^{-1}}, \tag{45}$$

$$\frac{\partial \mathbf{T}_{\text{vec}}}{\partial \mathbf{a}_k} = ((\mathbf{N}'_k)^\top \otimes \mathbf{R}_{\mathbf{P}_k})\,\mathbf{G}\,\mathbf{J}_l(\mathbf{a}_k), \tag{46}$$

$$\frac{\partial \text{Log}(\mathbf{T})}{\partial \mathbf{a}_k} = \mathbf{J}_l^{-1}(\text{Log}(\mathbf{T}))\text{Ad}_{\mathbf{P}_k}\mathbf{J}_l(\mathbf{a}_k). \tag{47}$$

For completeness (although not used in our tracking method), we show how to extend them to higher degrees and the analytic Jacobians of the velocity in $SE(3)$ as an appendix[3].

C. Implementation details related to the control points

Our method adds a new control point at each image timestamp t_i. This has cost benefits, since $t = t_i$ at Eq. 12, the control point $\mathbf{T}_i$ has no influence during the optimization (see how in Fig. 2b, $\tilde{B}_i(t_i) = 0$). However, it slightly reduces the interpolation capability. We argue that this loss is not significant since this control point has the smallest weight $\tilde{B}_i(t)$ during the interpolation, as inferred with Eq. 11.

Since the time spacing between control knots (timestamps of images) is quasi-constant, at t_i, the control point with greater influence is $\mathbf{T}_{i-2}$. This can intuitively be seen in Fig. 2b. From this observation, at t_i we initialize the origin of $\mathbf{T}_{i-2}$ to the centroid of the observed object point cloud. Its orientation is initialized with the one of the previous control point. Because of this protocol, to compute the interpolation $\mathbf{T}_{wo}(t_i)$, we need to have estimations of $\{\mathbf{T}_j\}_{j=i-3}^{i}$, so we wait until we have 4 observations to optimize its trajectory.

A simple factor graph for Spline BA with this protocol is shown in Fig. 4. For each set of observations in one image, we only add one control point to the optimization. This is done to deal with the gauge freedoms [47] that arise from optimizing a unique pose, $\mathbf{T}_{wo}$, with another 4 poses (the control points). Specifically, we fix the first (and second, in Local BA) and last two control points in the sliding window, thereby fixing the most optimized variables and the ones supported by the fewest number of observations respectively.

D. Front-end

To bootstrap the trajectory of an object, we first extract N (100 in our experiments) Shi-Tomasi features [48] in the

[3]See Appendices I, II of: https://arxiv.org/abs/2201.10602

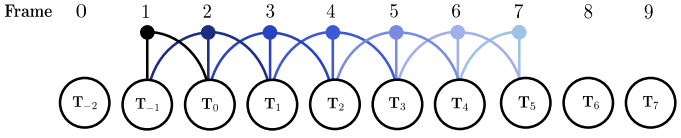


Fig. 4: Simple factor graph of Spline BA (optimization of only the control points). For simplicity a unique node factor is considered per frame (color coded, ● - ●). Each observation can influence three node variables. Fixing state variables (the ones without edges) deals with the gauge freedoms [47].

image region covered by one free mask, creating a point cloud used to initialize the first control point with its centroid and principal axes and subsequently initialize the set of 3D object points $\mathbf{p}_o$. Feature tracking is done with KLT [49]. A mask is considered *free* if it has not already been associated with a tracked object. An association is done when the major part of the tracked object features lie on a mask.

When tracking of features stop being successful, new ones are extracted, aiming at keeping its number at N. Since at the current timestamp t_i we have not initialized yet the control points $\mathbf{T}_{i-1}$ and $\mathbf{T}_i$, we cannot compute the interpolation $\mathbf{T}_{wo}(t_i)$. To tackle this problem, the extracted features at time t_i are tracked backwards to the image at t_{i-2}, at which an estimation of $\mathbf{T}_{wo}(t_{i-2})$ is available and hence we can estimate its 3D coordinates $\mathbf{p}_o$.

Applying naively the previous tracking method would lead to bad performance, as shown in Fig. 5a. On the one hand, once features are tracked to the border of an object, they tend to accumulate there since a region that contains the background has more photometric similarity. On the other hand, since masks are not perfect, features might be extracted at regions which do not belong to the object. For these reasons we impose backward consistency in the optical flow with the previous frame (up to ~ 0.1 pix.) and a mask refinement that discards pixels whose depth value strongly deviates from the robust median absolute value (MAD) [44] of the mask depths. A qualitive comparison between the naive and this improved tracking is shown in Fig. 5.

(a) (b)

Fig. 5: Feature tracks, shown as circles with lines indicating previous locations, obtained during a sequence from [50]. (a) With naive tracking, features tend to accumulate at the object borders and spurious tracks (see dashed yellow circle) appear. (b) Improvements in feature tracking after imposing backward consistency check and mask refinement.

V. EXPERIMENTAL RESULTS

A. Jacobian computations

Table I shows a comparison between running times for computing the Jacobian of an interpolated pose, $\mathbf{T}$, with respect to its 4 control points. They are obtained with an Intel i5-7400 CPU (throughout all experiments) in Python. The analytical (Ana.) method corresponds to the Jacobians derived in Sec. IV-B. Numerical differentiation (Num.) is computed via central differences. Autograd [51] is used for automatic differentiation (Auto.). If "(Lie)" is specified then $\mathrm{Log}(\mathbf{T})$ is differentiated ($\mathbf{T}_{\text{vec}}$ otherwise). Our derivations reduce the time execution by at least one order of magnitude.

Method	Time
Ana.	**2.51**
Ana. (Lie)	**2.84**
Num.	43.24
Num. (Lie)	50.42
Auto.	126.43
Auto. (Lie)	60.38

TABLE I: time [ms] for computing the same Jacobian.

Since our particular end goal is computing the Jacobians of the observation errors with respect to the control points ($\mathbf{J}_{\mathbf{e}_{\mathbf{p}_c}} = \partial \mathbf{e}_{\mathbf{p}_c}/\partial \mathbf{x}$), correspondent timings for a varying number of observations are also shown in Figure 6, where:

$$\frac{\partial \mathbf{e}_{\mathbf{p}_c}}{\partial (\mathbf{T}_{wo})_{\text{vec}}} = -\tilde{\mathbf{p}}_o^{\top} \otimes \mathbf{R}_{cw}, \tag{48}$$

$$\frac{\partial \mathbf{e}_{\mathbf{p}_c}}{\partial \mathrm{Log}(\mathbf{T}_{wo})} = -\begin{bmatrix}\mathbf{I}_{3\times 3} & -\mathbf{p}_o^{\wedge}\end{bmatrix} \mathbf{J}_l(\mathrm{Ad}_{\mathbf{T}_{cw}} \mathrm{Log}(\mathbf{T}_{wo}))\mathrm{Ad}_{\mathbf{T}_{cw}},$$

are the additional Jacobians composed with the chain rule. Both analytical fashions have better performance than the alternatives. In our implementation, since $N \sim 100$, the improvement is at least of one magnitude order.

B. Velocity estimations

We evaluate linear and angular velocity estimation errors with both continuous-time (CT) and discrete-time (DT) formulations in a synthetic setup. The evaluated trajectories (see Fig. 7d) consist of an object following a global z-axis turn parameterized with an angle θ_{transl} and a rotation over its own x-axis parameterized with the angle θ_{rot}. Both angles measure the relative increment between consecutive timestamps t_k and t_{k+1}. A wide range of $\{\theta_{\text{transl}}, \theta_{\text{rot}}\}$ is used to evaluate both small and big increments.

For DT, as it is common in the literature [8], [10], we assume constant kinematics between two timestamps, with $t_{k+1} = t_k + \Delta t$. The linear ($\mathbf{v}_o$) and angular ($\boldsymbol{\omega}_o$) velocity at $t \in [t_k, t_{k+1})$, for *coupled* and *decoupled* translation and rotation, are given by Eqs. 49 and 50 respectively.

$$\boldsymbol{\tau}_{o,k} = \begin{bmatrix}\mathbf{v}_{o,k}^{\top} & \boldsymbol{\omega}_{o,k}^{\top}\end{bmatrix}^{\top} = \frac{1}{\Delta t}\mathrm{Log}(\mathbf{T}_{wo,k}^{-1}\mathbf{T}_{wo,k+1}), \tag{49}$$

$$\mathbf{v}_{o,k} = \frac{1}{\Delta t}\mathbf{R}_{wo,k}^{\top}\mathbf{t}_{w_{k,k+1}},\ \boldsymbol{\omega}_{o,k} = \frac{1}{\Delta t}\mathrm{Log}(\mathbf{R}_{o_{k,k+1}}), \tag{50}$$

with $\mathbf{t}_{w_{k,k+1}} = \mathbf{t}_{wo,k+1} - \mathbf{t}_{wo,k}$, $\mathbf{R}_{o_{k,k+1}} = \mathbf{R}_{wo,k}^{\top}\mathbf{R}_{wo,k+1}$.

Since we are only interested in evaluating the velocity estimations, we assume a known object location (best case). Thereby, for the DT evaluation, the object reference frames match the ground-truth. For CT, since there is no closed form solution for the control points given a trajectory, we also match

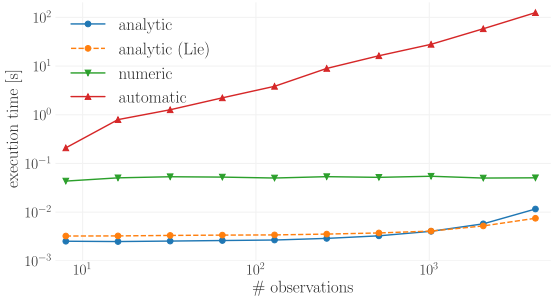
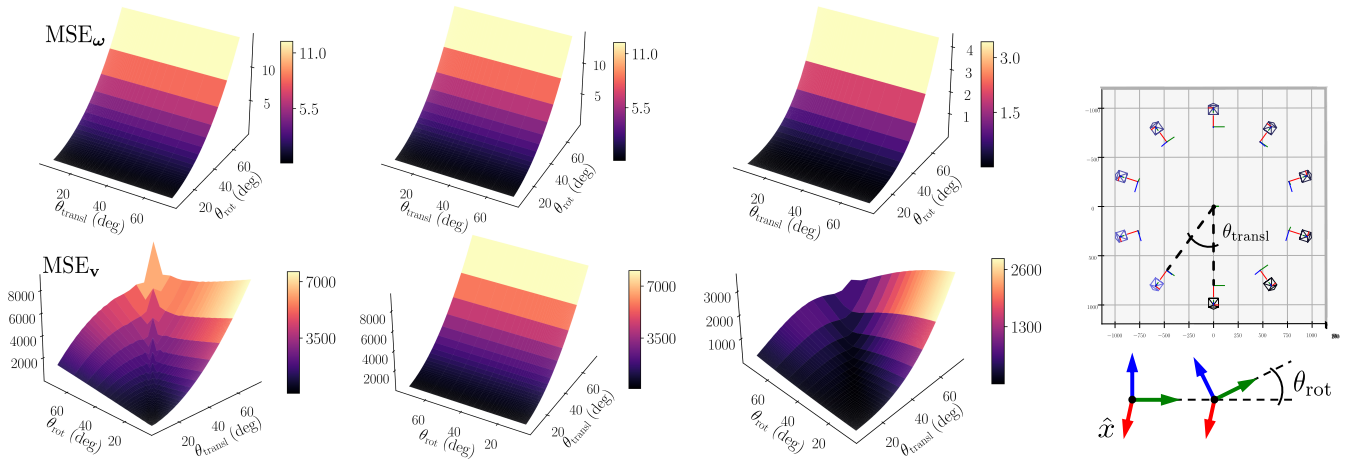


Fig. 6: Jacobian computation timings whith Eqs. 48 included.



(a) Discrete-time (coupled). (b) Discrete-time (decoupled). (c) Continuous-time. (d) Schematic of the experiment.

Fig. 7: (a), (b), (c): MSE surfaces of angular [(rad/s)^2] (top) and linear [(mm/s)^2] (bottom) velocity estimations for a circular trajectory parameterized with θ_{transl} and θ_{rot} (spatial and orientation increment between timestamps) defined as in (d). (a) and (b) correspond with Eqs. 49 and 50. Observe how the continuous-time estimation yields significantly lower errors.

them to the object poses. Although this disadvantages CT estimations, it can be seen in Fig. 7 that they still yield the lowest mean squared errors MSE$_\mathbf{v}$ and MSE$_\boldsymbol{\omega}$.

C. Comparison against baselines

Finally, we evaluate the whole method in the sequence swinging_4_unconstrained of OMD [50], which contains Vicon ground truth trajectories of 4 textured boxes experimenting multiple and independent $SE(3)$ motions. Since our system estimates the control points of a cumulative B-Spline curve, for this evaluation we compute the interpolation $\mathbf{T}_{wo,k}(t_k)$ at each timestamp t_k of the sequence with Eq. 12. Following previous works, two $SE(3)$ transformations are applied to align both the global and object coordinate systems by only using the first 50 images.

In Table II we compare our system against the state of the art using the metrics reported in [9], [11], which are the maximum component of the translation error and the norm (instead of each component) of the maximum angular errors. We compare with the pose-only results of [11] since it resembles the most to our optimization (we do not include kinematics in the error term). Additionally in Table III, we compare the Absolute Trajectory Error (ATE), which measures the global consistency, against [10].

In terms of the trajectories global consistency, our system consistently gives lower errors than [10]. We believe this is due to its constant velocity assumption between frames. Our system has also the flexibility of estimating a constant velocity but is not constrained to only that, it can exploit more complex kinematics due to its $\mathbb{C}^2$ continuity.

System	Box 1		Box 2		Box 3		Box 4	
	xyz	$\|$rpy$\|$	xyz	$\|$rpy$\|$	xyz	$\|$rpy$\|$	xyz	$\|$rpy$\|$
[11] MVO (Pose)	**0.09**	11.21	0.31	68.20	**0.13**	**5.40**	0.55	93.14
[9] ClusterVO	0.24	**6.09**	0.45	**66.70**	0.24	15.03	4.69	193.54
[8] VDO	1.06	56.77	0.40	169.58	1.30	19.12	0.76	155.55
Ours (w/ Local BA)	0.39	42.27	**0.30**	126.11	0.77	33.33	0.47	169.92
Ours (w/o Local BA)	0.29	36.55	0.38	107.28	0.27	26.44	**0.32**	**61.28**

TABLE II: Maximum component of translation error [m] (xyz), and norm of the maximum angular error [deg] ($\|$rpy$\|$), in swinging_4_unconstrained sequence [50].

System	Box 1	Box 2	Box 3	Box 4
[10] DynaSLAM II	0.41	0.37	1.09	0.28
Ours (w/ Local BA)	0.16	**0.18**	0.37	0.38
Ours (w/o Local BA)	**0.12**	0.19	**0.12**	**0.21**

TABLE III: ATE [m] in swinging_4_unconstrained [50].

In terms of the maximum translational error, our proposal gives lower values in at least half of the motions. However, it only performs better in one of the trajectories w.r.t. the angular errors. We believe this is due to reaching local minima during the optimization. This situation is specially harmful, since this propagates to several timestamps due to the interpolation nature of our formulation. We think that this can be addressed with a more sophisticated optimization.

VI. CONCLUSIONS AND FUTURE WORK

This work presents a continuous-time 6-DoF tracking approach for dynamic objects observed by a mobile RGB-D sensor. This is done by fitting their trajectories to cubic cumulative B-Spline curves. Special care has been taken in reducing the computational costs by deriving the analytical Jacobians of the interpolated pose with respect to the control points, thus promoting real-time capabilities for future works using this kind of curve. The evaluation has shown the potential of the proposal. Our results are on par with the state of the art, showing significant improvements in certain aspects like global consistency and velocity estimation.

As future work, we find interesting to explore higher order continuity curves. This could increase the flexibility of the trajectories as the recent work [52] suggests. Additionally, integration with a real-time SLAM system can increase its applicability, something that could be achieved thanks to our sequential formulation and analytical Jacobians. Finally, to discover new objects in the scene, we find motion clustering [7] very promising instead of relying on 2D masks.

REFERENCES

[1] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[2] J. L. Schonberger and J.-M. Frahm, "Structure-from-motion revisited," in *IEEE conference on computer vision and pattern recognition*, 2016.

[3] M. R. U. Saputra, A. Markham, and N. Trigoni, "Visual slam and structure from motion in dynamic environments: A survey," *ACM Computing Surveys (CSUR)*, vol. 51, no. 2, pp. 1–36, 2018.

[4] B. Bescos, J. M. Fácil, J. Civera, and J. Neira, "Dynaslam: Tracking, mapping, and inpainting in dynamic scenes," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4076–4083, 2018.

[5] D.-H. Kim and J.-H. Kim, "Effective background model-based rgb-d dense visual odometry in a dynamic environment," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1565–1573, 2016.

[6] I. Ballester, A. Fontan, J. Civera, K. H. Strobl, and R. Triebel, "Dot: dynamic object tracking for visual slam," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 11 705–11 711.

[7] K. M. Judd, J. D. Gammell, and P. Newman, "Multimotion visual odometry (mvo): Simultaneous estimation of camera and third-party motions," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3949–3956.

[8] J. Zhang, M. Henein, R. Mahony, and V. Ila, "Vdo-slam: a visual dynamic object-aware slam system," *arXiv:2005.11052*, 2020.

[9] J. Huang, S. Yang, T.-J. Mu, and S.-M. Hu, "Clustervo: Clustering moving instances and estimating visual odometry for self and surroundings," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 2168–2177.

[10] B. Bescos, C. Campos, J. D. Tardós, and J. Neira, "Dynaslam ii: Tightly-coupled multi-object tracking and slam," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5191–5198, 2021.

[11] K. M. Judd and J. D. Gammell, "Multimotion visual odometry (mvo)," *arXiv preprint arXiv:2110.15169*, 2021.

[12] M.-J. Kim, M.-S. Kim, and S. Y. Shin, "A general construction scheme for unit quaternion curves with simple high order derivatives," in *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, 1995, pp. 369–376.

[13] 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.

[14] C. Tomasi and T. Kanade, "Shape and motion from image streams under orthography: a factorization method," *International journal of computer vision*, vol. 9, no. 2, pp. 137–154, 1992.

[15] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, 2003.

[16] P. Sturm and B. Triggs, "A factorization based algorithm for multi-image projective structure and motion," in *European conference on computer vision*. Springer, 1996, pp. 709–720.

[17] M. Han and T. Kanade, "Reconstruction of a scene with multiple linearly moving objects," *International Journal of Computer Vision*, vol. 59, no. 3, pp. 285–300, 2004.

[18] L. Zappella, A. Del Bue, X. Lladó, and J. Salvi, "Joint estimation of segmentation and structure from motion," *Computer Vision and Image Understanding*, vol. 117, no. 2, pp. 113–129, 2013.

[19] R. Sabzevari and D. Scaramuzza, "Monocular simultaneous multi-body motion segmentation and reconstruction from perspective views," in *IEEE International Conference on Robotics and Automation*, 2014.

[20] C.-C. Wang, C. Thorpe, and S. Thrun, "Online simultaneous localization and mapping with detection and tracking of moving objects: Theory and results from a ground vehicle in crowded urban areas," in *2003 IEEE International Conference on Robotics and Automation*, vol. 1, 2003, pp. 842–849.

[21] C.-C. Wang, C. Thorpe, S. Thrun, M. Hebert, and H. Durrant-Whyte, "Simultaneous localization, mapping and moving object tracking," *The International Journal of Robotics Research*, vol. 26, no. 9, pp. 889–916, 2007.

[22] C. Bibby and I. Reid, "Simultaneous localisation and mapping in dynamic environments (slamide) with reversible data association," in *Proceedings of Robotics: Science and Systems*, vol. 66, 2007, p. 81.

[23] ——, "A hybrid slam representation for dynamic marine environments," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 257–264.

[24] A. Kundu, K. M. Krishna, and C. Jawahar, "Realtime multibody visual slam with a smoothly moving monocular camera," in *2011 International Conference on Computer Vision*, 2011, pp. 2080–2087.

[25] P. Li, T. Qin, *et al.*, "Stereo vision-based semantic 3d object and ego-motion tracking for autonomous driving," in *European Conference on Computer Vision*, 2018.

[26] S. Yang and S. Scherer, "Cubeslam: Monocular 3-d object slam," *IEEE Transactions on Robotics*, vol. 35, no. 4, pp. 925–938, 2019.

[27] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *IEEE international conference on computer vision*, 2017.

[28] Q. Wang, L. Zhang, L. Bertinetto, W. Hu, and P. H. Torr, "Fast online object tracking and segmentation: A unifying approach," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[29] 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 *2018 International Conference on 3D Vision (3DV)*. IEEE, 2018, pp. 381–389.

[30] A. Patron-Perez, S. Lovegrove, and G. Sibley, "A spline-based trajectory representation for sensor fusion and rolling shutter cameras," *International Journal of Computer Vision*, vol. 113, no. 3, pp. 208–219, 2015.

[31] C. Kerl, J. Stückler, and D. Cremers, "Dense continuous-time tracking and mapping with rolling shutter rgb-d cameras supplementary material," *TC*, vol. 1, no. 1, p. A1.

[32] 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, 2018.

[33] D. Droeschel and S. Behnke, "Efficient continuous-time slam for 3d lidar-based online mapping," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 5000–5007.

[34] A. J. Yang, C. Cui, I. A. Bârsan, R. Urtasun, and S. Wang, "Asynchronous multi-view slam," *arXiv:2101.06562*, 2021.

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

[36] K. M. Lynch and F. C. Park, *Modern robotics*. Cambridge University Press, 2017.

[37] S. Agarwal, K. Mierle, and Others, "Ceres solver," http://ceres-solver.org.

[38] C. Sommer, V. Usenko, D. Schubert, N. Demmel, and D. Cremers, "Efficient derivative computation for cumulative b-splines on lie groups," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 148–11 156.

[39] J. Sola, J. Deray, and D. Atchuthan, "A micro lie theory for state estimation in robotics," *arXiv preprint arXiv:1812.01537*, 2018.

[40] T. D. Barfoot, *State estimation for robotics*. Cambridge University Press, 2017.

[41] M. G. Cox, "The numerical evaluation of b-splines," *IMA Journal of Applied mathematics*, vol. 10, no. 2, pp. 134–149, 1972.

[42] K. Qin, "General matrix representations for b-splines," *The Visual Computer*, vol. 16, no. 3-4, pp. 177–186, 2000.

[43] F. Dellaert, M. Kaess, *et al.*, "Factor graphs for robot perception," *Foundations and Trends in Robotics*, vol. 6, no. 1-2, pp. 1–139, 2017.

[44] P. J. Huber, *Robust statistics*. John Wiley & Sons, 2004, vol. 523.

[45] J.-L. Blanco, "A tutorial on se (3) transformation parameterizations and on-manifold optimization."

[46] H. Strasdat, "Local accuracy and global consistency for efficient visual slam," Ph.D. dissertation, Department of Computing, Imperial College London, 2012.

[47] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment—a modern synthesis," in *International workshop on vision algorithms*. Springer, 1999, pp. 298–372.

[48] J. Shi and C. Tomasi, "Good features to track," in *IEEE conference on computer vision and pattern recognition*, 1994.

[49] J.-Y. Bouguet *et al.*, "Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm," *Intel corporation*, vol. 5, no. 1-10, p. 4, 2001.

[50] K. M. Judd and J. D. Gammell, "The oxford multimotion dataset: Multiple se (3) motions with ground truth," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 800–807, 2019.

[51] D. Maclaurin, D. Duvenaud, and R. P. Adams, "Autograd: Effortless gradients in numpy," in *ICML 2015 AutoML workshop*, 2015.

[52] G. Cioffi, T. Cieslewski, and D. Scaramuzza, "Continuous-time vs. discrete-time vision-based slam: A comparative study," *IEEE Robotics and Automation Letters*, pp. 1–1, 2022.