Analytical Jacobian Approximation for Direct Optimization of a Trajectory of Interpolated Poses on SE(3)

Kazii Botashev[1] and Gonzalo Ferrer[1]

Abstract— **This paper relates to time-continuous trajectory representation using direct linear interpolation on SE(3). Our approach focuses on a novel analytical Jacobian approximation of a sequence of linearly interpolated poses on SE(3). This paper shows a derivation of the proposed analytical Jacobian using retraction mapping and an approximation to the commutativity property of infinitesimal group elements. We provide plenty of evaluations for 3 different optimization problems. For the synthetic point cloud alignment problem, our proposed Jacobian is compared with a numerical one. For the synthetic pose graph optimization problem, the proposed Jacobian approximation allows us to reduce by x7 factor the state dimensions while keeping a similar magnitude of resulting error compared to the full discrete-time trajectory. Finally, we show the validity of our approach in a time-continuous approach for real-world LIDAR odometry problem.**

I. INTRODUCTION

Pose estimation problems are often formulated using a discrete-time representation of a sequence of 3D poses over time – a trajectory. It is the default representation due to a consolidated theory on state estimation and clear interpretability of the results [1], [2]. In addition, many trajectory estimation approaches for robot-sensor systems successfully use a discrete-time formulation since it allows keeping the state size tractable [3], [4]. However, this is unattainable for setups that include different high-rate or asynchronous sensors [5], [6], making discrete-time formulation non-suitable for those scenarios without additional assumptions.

The straightforward solution is to use a time-continuous trajectory representation that enables the addition of information from a high-rate sensor output, preventing to estimate a discrete state for each observation which would produce a high number of state variables to be estimated. Time-continuous trajectories are also capable of fusing multiple asynchronous sensors of different high rates in a natural way. In addition, a time-continuous trajectory representation should allow for a direct calculation of the Jacobian since most modern state estimation techniques are built around optimization.

Time-continuity of the trajectory can be ensured using different methods, the most convenient being the Gaussian process used in [7], [8], B-splines interpolation used for split rotation and translation interpolation as in [5], [6] or for direct interpolation on Special Euclidean Group $SE(3)$ as in [9], [10], and finally, linear interpolation of split pose representation as in [11], [12] or for direct interpolation on $SE(3)$ as in [13], [14], [15].

A comprehensive comparison of different motion interpolation methods is provided in [16]. Despite the $\mathcal{C}^d$ continuity of the B-spline interpolated trajectory, it has a higher computational cost than the linear form and may over-smooth the trajectory by vanishing its high-frequency components. Alternatively, the linear interpolation keeps the raw high-frequency trajectory components and has a much lower computational cost. It is continuous but, at the same time, non smooth on the boundary of the control poses, which is its main disadvantage.

Concerning the optimization, it is a valid question to wonder about the efficiency and correctness of the Jacobian calculation for the corresponding time-continuous representations. Unfortunately, many of the above-mentioned methods keep this issue out of scope by using Jacobians obtained with auto-differentiation functionality of popular non-linear least square problem solvers. Nevertheless, having the analytical form is essential for real-time applications with limited computational resources.

The Jacobians for linear interpolation of split pose representation are introduced in [12]. It presents a Jacobian form related to spherical linear interpolation for rotation and linear interpolation for translation and solves the continuous-time LiDAR-Inertial Odometry problem with Sweep Reconstruction. The analytical Jacobian with respect to control poses of direct B-splines interpolation on $SE(3)$ is derived in [9]. Authors approximate it by applying Baker–Campbell–Hausdorff (BCH) formula to 3D poses, which is previously derived in Barfoot's book [17] for the direct linear pose interpolation on $SE(3)$.

The contributions of this paper are as follows. We derive an analytical form of the Jacobian for linearly interpolated poses on $SE(3)$ using an approximation by the commutativity property of infinitesimal group elements instead of the BCH formula approximation. We evaluate empirically our approach on several synthetic and real-world datasets and problems, provide a comparison with the formulation proposed in [17] and show its consistency in robustly achieving accurate results in optimization-based state estimation of 3D trajectories.

The paper is organized as follows. Section II introduces background on Special Euclidean Group $SE(3)$ and the retraction operation. Section III describes interpolation in the manifold and the derivation of our proposed Jacobian approximation. Section IV presents the evaluation results for two synthetic problems, the multi-registration of point clouds

[1]The authors with the Skolkovo Institute of Science and Technology (Skoltech), Center for AI Technology. {kazii.botashev, g.ferrer}@skoltech.ru

and pose SLAM, and one real problem from [18] LiDAR data. Finally, Section V concludes the paper.

II. BACKGROUND

We operate with poses or Rigid Body Transformations (RBT) that are elements of Special Euclidean group $SE(3)$:

$$SE(3) = \left\{ T = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix} \mid \mathbf{R} \in SO(3)\,,\, \mathbf{t} \in \mathbb{R}^3 \right\}. \quad (1)$$

Rigid body transformations are elements of the matrix group $SE(3) \subset \mathbb{R}^{4\times 4}$, representing a manifold, a lower-dimensional structure. As such, a connection exists between elements of the $SE(3)$ group with a vector space $\mathbb{R}^6$. We will make use of the definition of a *retraction* $R_T(\xi)$, as defined in [19], which is a smooth mapping from the tangent space around the $T \in SE(3)$ element to the manifold:

$$R_T(\xi) : \mathbb{R}^6 \rightarrow SE(3) \quad (2)$$

Two conditions must be satisfied: i) $R_T(0_T) = T$ and ii) $\mathcal{TM}$ local rigidity.

Accordingly, the derivative around the element T becomes a well-defined operation now that the retraction allows us to operate in a vector space, where a directional derivative can be expressed as:

$$\left.\frac{\partial R_T(\xi)}{\partial \xi}\right|_{\xi=0}. \quad (3)$$

Note that this is not a derivative, but a mapping, defined at the zero element.

There are multiple options for choosing a retraction. In our case, the exponential map is the most natural retraction choice: its definition is highly related to the concept of perturbations over $SE(3)$, which is the main topic of this paper since we are interested in calculating the derivatives of linear interpolated 3D poses.

The exponential map defines the retractions R_T in the following way:

$$R_T(\xi) = \operatorname{Exp}(\xi)T, \quad (4)$$

where we have chosen a left-hand side emplacement of the exponent map as our default convention. A right-hand side convention would require re-doing all further derivations, but the procedure would be identical to what is expanded in the following sections.

III. INTERPOLATION IN THE MANIFOLD

Smooth manifolds allow us to import all the tools from calculus and real analysis into manifolds by following simple rules.

We follow [17] and define a linear interpolation of poses as $T(T_1, T_2, \tau) : SE(3) \times SE(3) \times [0,1] \rightarrow SE(3)$

$$T(T_1, T_2, \tau) = (T_2 T_1^{-1})^{\tau} T_1. \quad (5)$$

This form provides a direct interpolation in $SE(3)$. However, in order for us to differentiate for each pose, we require to define a new retraction as the Cartesian product of two manifold elements. In particular, when substituting the retraction (4), we obtain a map of the local coordinates of each of the poses:

$$\begin{aligned} R_{T_1T_2}(\xi_1, \xi_2) &= T\big(R_{T_1}(\xi_1), R_{T_2}(\xi_2), \tau\big) \\ &= \big(\operatorname{Exp}(\xi_2) T_2 T_1^{-1} \operatorname{Exp}(-\xi_1)\big)^{\tau} \operatorname{Exp}(\xi_1) T_1, \end{aligned} \quad (6)$$

where we have used the property $\operatorname{Exp}(\xi)^{-1} = \operatorname{Exp}(-\xi)$.

We want to compare the tangent space around both transformations, such that they are equal find out the relation of the tangent spaces ξ_1, ξ_2 and the new joint variable $\xi = [\xi_1, \xi_2]$ composed of the two poses as expressed in the retraction (6).

We will expand each of the terms, evaluated at the null element of the tangent vector $\xi = 0$:

$$\begin{aligned} \left. R_{T_1T_2}(0, \xi_2)\right|_{\xi_2=0} &= \left.\big(\operatorname{Exp}(\xi_2) T_2 T_1^{-1}\big)^{\tau} T_1\right|_{\xi_2=0} \\ &= \left.\operatorname{Exp}(\tau\xi_2)\big(T_2 T_1^{-1}\big)^{\tau} T_1\right|_{\xi_2=0} \end{aligned} \quad (7)$$

We have made use of the property $\operatorname{Exp}(\xi)^{\alpha} = \operatorname{Exp}(\alpha\xi)$ and the fact that this expression is *evaluated* at the zero element $\xi_2 = 0$. Thereby, the exponent $\operatorname{Exp}()$ equals the identity in the limit, and the matrix multiplication becomes commutative under this condition. Accordingly, $\operatorname{Exp}(\xi_2)$ can be moved away from the parenthesis if we consider that this expression is multiplied τ times.

We will take this relaxation to expand the infinitesimal exponent and obtain an analytical solution that is very close to the real analytical gradient, as discussed in the evaluation section. In general, the multiplication of matrix elements infinitesimally close to the identity is a non-commutative operation.

Similarly, the same derivation can be done for ξ_1

$$\begin{aligned} R_{T_1T_2}(\xi_1, 0) &= \big(T_2 T_1^{-1} \operatorname{Exp}(-\xi_1)\big)^{\tau} \operatorname{Exp}(\xi_1) T_1 \\ &= \big(T_2 T_1^{-1})\big)^{\tau} \operatorname{Exp}(-\xi_1)^{\tau} \operatorname{Exp}(\xi_1) T_1 \\ &= \Delta T \operatorname{Exp}((1-\tau)\xi_1) T_1 \\ &= \left.\operatorname{Exp}\big((1-\tau) Adj_{\Delta T}\xi_1\big) \cdot T\right|_{\xi_1=0}, \end{aligned} \quad (8)$$

where $\Delta T = (T_2 T_1^{-1})^{\tau}$.

Suppose we arrange (7) and (8) under the condition that both expressions will be evaluated at zero. In that case, we obtain the following compact linear relation that is only valid for calculating derivatives:

$$\left.\xi\right|_{\xi=0} = (1-\tau) Adj_{\Delta T}\xi_1 + \left.\tau\xi_2\right|_{\xi_1=0,\,\xi_2=0}. \quad (9)$$

Now, suppose we have a cost function that depends on a linearly interpolated pose on $SE(3)$. In that case, we can find its derivative with respect to reference poses T_1 and T_2 by using the retraction mapping in (6) and the chain rule [19] as follows:

$$\left.\frac{\partial h(T)}{\partial T}\right|_{T} = \left.\frac{\partial h_T(\xi)}{\partial \xi}\right|_{\xi=0} \quad (10)$$

$$\frac{\partial h_T(\xi(\xi_1,\xi_2))}{\partial \xi_{1,2}}\bigg|_{\xi=0} = \frac{\partial h_T(\xi)}{\partial \xi}\frac{\partial \xi}{\partial \xi_{1,2}}\bigg|_{\xi=0}$$
$$= \frac{\partial h_T(\xi)}{\partial \xi}\bigg|_{\xi=0}\begin{bmatrix}(1-\tau)Adj_{\Delta T} & \tau I\end{bmatrix} \quad (11)$$

Alternatively, we can state it as:

$$\frac{\partial T(T_1,T_2,\tau)}{\partial T_1} = (1-\tau)Adj_{\Delta T} \quad (12)$$

$$\frac{\partial T(T_1,T_2,\tau)}{\partial T_2} = \tau I \quad (13)$$

The results (11), (12) and (13) are our proposed analytical Jacobian approximation that we imply to use for direct optimization of trajectories of interpolated poses on $SE(3)$. Analyzing it more closely, we can see that the resulting Jacobian form is in direct ratio with the interpolation time τ in the way that gradually moving from T_1 to the T_2 is coupled with the increase or decrease of the corresponding parts of the resulting Jacobian. Besides that, one can see that computation of this Jacobian is relatively lightweight since it is just a single matrix-scalar multiplication.

In the following sections, we evaluate it in three problems showing its consistency. For that, we first evaluate it alongside another form that is introduced in [17] by approximation with Baker–Campbell–Hausdorff (BCH) formula resulting in:

$$\begin{gathered}\delta\xi = (\mathbf{1} - \mathcal{A}(\tau,\xi_{21}))\,\delta\xi_1 + \mathcal{A}(\tau,\xi_{21})\,\delta\xi_2 \\ A(\tau,\xi) = \tau\mathcal{J}(\tau\xi)\mathcal{J}(\xi)^{-1}\end{gathered} \quad (14)$$

where ξ_{21} defined as $R_T(\xi_{21}) = \text{Exp}(\xi_{21})T_2T_1^{-1}$, and $\mathcal{J}$ is a left Jacobian of $SE(3)$ (see [17]).

IV. EVALUATION

The main purpose of the experiments is to prove the applicability of the proposed approximated gradient form to be used in various problems that can apply time-continuous trajectory representation and optimization.

For that, we first conduct an experiment on time-continuous point-cloud alignment problem (multi-view registration) with synthetic randomly generated data. In this task we compare results achieved using the proposed gradient form (11), Barfoot's [17] formulation (14) and the numerically estimated gradient obtained using the finite difference method. We imply numerical gradient as the most accurate one to compare and use its results as a baseline.

In the second experiment we investigate how the proposed gradient formulation is suitable for solving time-continuous Pose Simultaneous Localization and Mapping (SLAM) problem on a synthetic benchmark - Sphere dataset [20].

And finally, we qualitatively describe the applicability of the proposed gradient form for real data implementing it inside the Continuous-Time Iterative Closest Point (CT-ICP) [11] algorithm and achieving trajectory consistency for short subsamples of raw uncorrected KITTI dataset [18] scenes. We obtain results for all three experiments using Intel i7-9750H CPU.

A. Synthetic Time-Continuous Point Cloud Alignment

We start with testing the applicability of the proposed gradient formulation (11) for a multi-view point cloud alignment problem using randomly generated synthetic data.

1) Problem description: Point cloud alignment task implies estimating a relative transform $T_{1\rightarrow 2}$ between two poses $\{T_1,T_2\} \in SE(3)$ from which we observe a shared cloud of points.

In our case, we randomly generate point cloud $\mathcal{P}\{p_1,p_2,...,p_N | p_i \in \mathbb{R}^3\}$ of size N. After that, we generate two random poses $\{T_1,T_2\} \in SE(3)$. We treat these poses as the interpolation interval's initial and final ends. We uniformly split this interval with K intermediate timestamps $\tau_k \in [0,1]$. Using these timestamps, we interpolate initial and final poses, getting corresponding intermediate poses $\mathcal{T}\{T_{\tau_1},T_{\tau_2},...,T_{\tau_K} | T_{\tau_k} = T(T_1,T_2,\tau_k)\}$.

For each interpolated pose we generate corresponding point cloud observations $p_i^{\tau_k}$ by adding noise $\alpha_i^{\tau_k}$ randomly sampled from normal distribution $\mathcal{N}(0,\Sigma_i^{\tau_k})$ to each point obtaining K different sets of points $\mathcal{P}_k$. Here $\Sigma_i^{\tau_k} = I_3\alpha_i^{\tau_k}$ where $\alpha_i^{\tau_k} \in \mathbb{R}^3$ sampled from one more distribution $\mathcal{N}(0,\sigma I_3)$. By that, we can specify the noise magnitude of the points observations by varying parameter σ and thus investigate the performance of the gradient forms during optimization with different challenging conditions.

Furthermore, the last important parameter related to our synthetic data generation approach is a scale parameter s. Changing this parameter, we can scale by s factor translation components $t \in \mathbb{R}^3$ of all poses and points, keeping the rotation parts $R \in SO(3)$ unchanged. One can think about this as stretching or shrinking the scene without changing its internal structure.

Having all these data generated, we seek to find optimal initial and final poses T_1 and T_2 to achieve a points observation consistency for all poses in $\mathcal{T}$. For that we minimize the following cost function $h(T_1,T_2)$ with respect to initial or final pose:

$$\begin{gathered}h(T_1,T_2) = \frac{1}{2}\sum_{i=0}^{N-1}\sum_{\substack{\tau_j \\ \tau_j\neq\tau_k}}\sum_{\tau_k}\bigg\| T_\tau(T_1,T_2,\tau_j)p_i^{\tau_j} - \\ - T_\tau(T_1,T_2,\tau_k)p_i^{\tau_k}\bigg\|^2_{\Sigma_i^{\tau_j}+\Sigma_i^{\tau_k}}\end{gathered} \quad (15)$$

This cost function implies pair-wise all-vs-all comparison of i-th point observation for all poses in $\mathcal{T}$.

Denoting the expression under the norm as a residual $r_i(T_1,T_2,\tau_j,\tau_k)$ that describes an observation point relation between some two interpolated poses and introducing $\Sigma_i = \Sigma_i^{\tau_j} + \Sigma_i^{\tau_k}$ we rewrite (15) in a shorter form as:

$$h(T_1,T_2) = \frac{1}{2}\sum_{i=0}^{N-1}\sum_{\substack{\tau_j \\ \tau_j\neq\tau_k}}\sum_{\tau_k}\|r_i(T_1,T_2,\tau_j,\tau_k)\|^2_{\Sigma_i} \quad (16)$$

In order to separately testify parts of the gradient related to the initial or final pose, we run two experiments. In the first case, we optimize only initial pose T_1, fixing final pose T_2 as an identity transform. In the second case, we do the vice-verse and optimize T_2 keeping T_1 fixed. As a result, our goal is to solve the following optimization problem:

$$\min_{\substack{T_1, T_2 = \boldsymbol{I} \\ \text{or} \\ T_2, T_1 = \boldsymbol{I}}} h(T_1, T_2) \tag{17}$$

Using the following gradient expression:

$$\nabla_{T_{1,2}} h = \sum_{i=0}^{N-1} \sum_{\tau_j} \sum_{\substack{\tau_k \\ \tau_j \neq \tau_k}} r_i(T_{1,2})^\top \Sigma_i^{-1} \frac{\partial r_i}{\partial T_{1,2}} \tag{18}$$

where $T_{1,2} \in \{T_1, T_2\}$ and:

$$\frac{\partial r_i}{\partial T_{1,2}} = \left[\frac{\partial r_i}{\partial T_{\tau_j}} \frac{\partial T_{\tau_j}}{\partial T_{1,2}} - \frac{\partial r_i}{\partial T_{\tau_k}} \frac{\partial T_{\tau_k}}{\partial T_{1,2}} \right] \tag{19}$$

is a gradient of the residual r_i with respect to optimization pose and $\frac{\partial T_\tau}{\partial T_{1,2}}$ - is a gradient of the interpolation pose computed using our proposed method or Barfoot's form.

2) Experiment results: In our experiments for that task, we test different setups of the problem by checking different combinations of the parameters defined in the synthetic data generation description above. To be more precise, we solve this task using the Gauss-Newton method for all combinations of the following parameters: point cloud size $N \in [20, 40, 80, 100]$, number of interpolation timestamps $K \in [5, 20, 40, 80, 100]$, scale factors $s \in [1, 10, 100]$ and noise magnitude $\sigma = 0.01$.

We optimize these tasks using the proposed gradient, Barfoot's BCH approximation, and numerical form. We initialize optimized $T_{1,2}$ as identity transform and compare results with a ground truth T_{gt} using a distance metric which we denote as a norm of the Lie algebra coordinates vector:

$$d = \left\| Ln(T_{1,2} T_{gt}^{-1}) \right\| \tag{20}$$

These experiments allow us to comprehensively test each Jacobian formulation performance and limitations. We provide main insights on results in Fig. 1. The left column graphs (**a**)-(**e**) - results for the initial pose T_1 optimization (first case), and the right column (**f**)-(**j**) - results for the final pose T_2 (second case). In (**a**) and (**f**), we show that with an increase in the number of the interpolated states K used, the accuracy of the final result also increases for both cases. The same improvement also occurs with the increase of the number of points in a point cloud N as shown in (**b**) and (**g**). Direct comparison of Jacobian magnitudes presented in (**c**) and (**h**) shows the close similarity of all three methods.

Summing the final results for all experiments, we see that all three approaches show similar results and achieve a feasible solution for both cases. Barfoot's method exactly reproduces the numerical form with the same results. Our proposed method closely follows the others providing similar or better final optimization results. Although our method does not exactly follow the numerical one, it still results in a similar number of optimization steps to converge as shown in (**d**) and (**i**) and, more importantly, being computationally lighter, it requires noticeably less time to obtain the result. As shown in (**e**) and (**j**), our method reaches the solution ~ 2 times faster than the BCH form and ~ 3 times faster than the numerical. More precisely, our method requires 0.02 ms to compute $\frac{\partial T_\tau}{\partial T_{1,2}}$ in (19), while Barfoot's form - 0.16 ms.

This evaluation experiment shows that our gradient approximation for the linearly interpolated rigid body transform (RBT) has no usage limitations and allows us to expect it to be sufficient for other problems.

B. Synthetic Pose-SLAM

We proceed by testing the applicability of the proposed Jacobian approximation for the Pose SLAM problem.

1) Problem description: The Pose SLAM is a nonlinear pose graph optimization problem that estimates N robot poses $T_i \in SE(3)$ using M relative pose observations $T_{ij}^{obs} \in SE(3)$. This problem can be seen as a directed graph: robot poses T_i are graph nodes that we want to estimate, while relative pose observations are graph edges $\mathcal{E}_{ij}$ that encodes a relative pose transformation measurement between two poses T_i and T_j.

The classic Pose SLAM graph optimization estimates robot poses by solving the following optimization problem:

$$\min_{\{T_i\} \in SE(3)} \sum_{\{\mathcal{E}_{ij}\}} g(T_i, T_j, T_{ij}^{obs}) \tag{21}$$

$$g(T_i, T_j, T_{ij}^{obs}) = T_i T_{ij}^{obs} T_j^{-1} \tag{22}$$

Function $g(T_i, T_j, T_{ij}^{obs})$ measures how well our observation matches pair (i, j) of input nodes.

We use this approach as a baseline for comparison. In order to check the performance of our Jacobian of interpolated RBT, we modify the classic Pose SLAM problem by decimating the trajectory in the factor graph using linear interpolation among them. The decimation factor is expressed by the δ parameter. Regarding the graphical model, the number of nodes is reduced δ times and the number of factors (observations) remains unaltered.

With that, it is possible to achieve almost similar final results with less computations per optimization step by solving a smaller optimization task. We further refer to those nodes we keep as *base*-nodes and those we replace with interpolation as *inter*-nodes.

Accordingly, we can modify our Pose SLAM optimization problem (21) and cost function (22) as:

$$\min_{\{\hat{T}_i\} \in SE(3)} \sum_{\{\mathcal{E}_{ij}\}} \hat{g}(T_{\tau_i}, T_{\tau_j}, T_{ij}^{obs}) \tag{23}$$

$$\hat{g}(T_{\tau_i}, T_{\tau_j}, T_{ij}^{obs}) = T_{\tau_i} T_{ij}^{obs} T_{\tau_j}^{-1} \tag{24}$$

$$T_{\tau_i}(\hat{T}_a, \hat{T}_b, \tau_i) = (\hat{T}_b \hat{T}_a^{-1})^{\tau_i} \hat{T}_a \tag{25}$$

where $\hat{g}$ is our new interpolated cost function, $\{\hat{T}_i\}$ - our novel sparse set of *base*-nodes to estimate, T_{τ_i} - pose of the i-th *inter*-node, its corresponding previous $\hat{T}_a$ and next $\hat{T}_b$ *base*-nodes used for interpolation with timestamp τ_i.

TABLE I

RPE before and after optimization of pose graphs with different sparsity factors

Sparsity factor	Num factors	Num nodes	Time per opt step, sec	Base nodes only			Full trajectory			
							Interpolated posegraph		Classic	
				RPE Before	RPE Interp. posegraph	RPE Classic	RPE Before	RPE After	RPE Before	RPE After
x1 (classic)	9788	2500	0.51	73.13	—	4.13	—	—	73.13	4.13
x4		625	0.19	73.09	10.71	4.13	84.94	11.46		
x8		313	0.16	73.02	12.35	4.16	84.95	14.22		
x16		157	0.11	72.96	63.71	4.20	84.96	73.64		
x32		79	0.07	73.00	85.35	4.18	85.73	85.16		

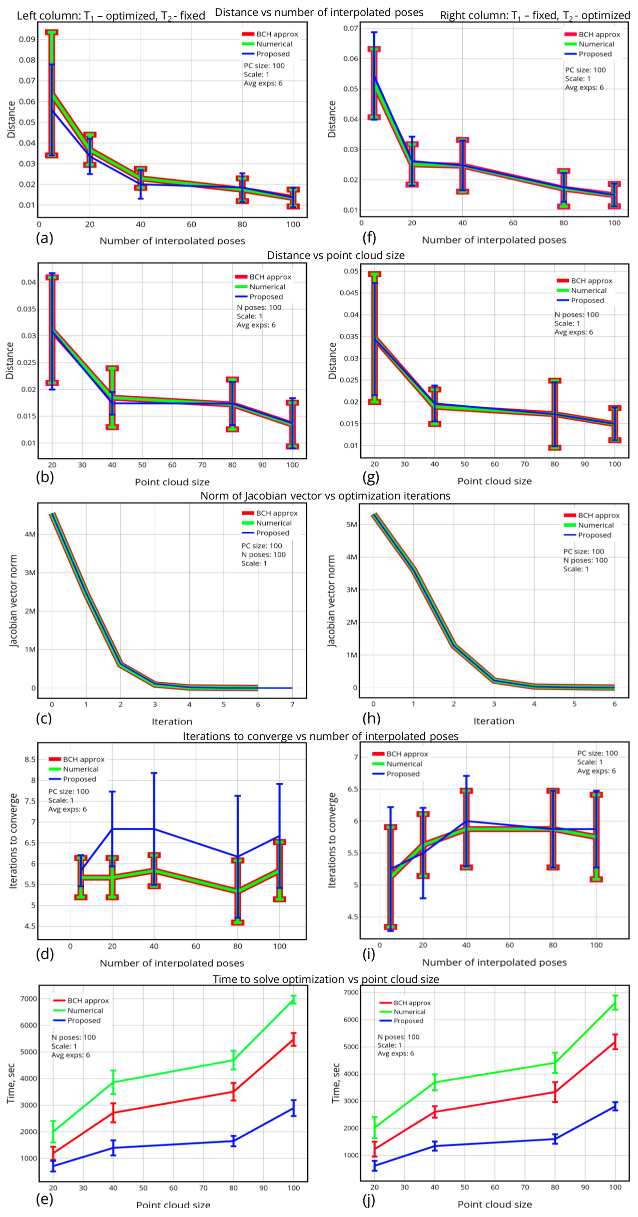

Fig. 1. Aggregated results for the synthetic time-continuous point cloud registration problem. The left column shows the results of optimizing T_1, right column - T_2. Plots (a) and (f) represent optimization results vs different numbers of interpolated poses K. Graphs (b) and (g) show optimization results vs different numbers N of points in point cloud. Magnitudes of Jacobians are represented at (c) and (h). The number of iterations to converge vs number of interpolated poses K is represented at (d) and (i), time to converge vs number N of points in point cloud is at (e) and (j).

2) Experiment results: We use the sphere benchmark [20] as a data source. It consists of 2500 poses and 9788 observations. We first solve the classic Pose SLAM problem (21) and use it as a baseline for comparison. After that, we solve Interpolated Pose SLAM (23) using our analytical Jacobian form, comparing its results for different sparsity factors with a baseline. We use the mrob C++ library as a framework for optimization [21].

We provide visualization for the Interpolated Pose SLAM for sparsity factor $\delta = 4$ at Fig. 2 where (a) depicts an initial state of $base$-nodes and (b) represents the same nodes after optimization. One can see the visible improvement obtained by keeping only 625 of the original 2500 poses of the original dataset.

Tab I provides more comprehensive results and comparison with a classic approach for different sparsity factors. We use Relative Pose Error (RPE) as a metric for comparison. We define it as an average pair-wise all-vs-all distance (20) between relative transforms of poses inside the estimated trajectory and the ground truth one. Base nodes only comparison performed using only nodes that we use as $base$-nodes in our interpolated pose graph. Full trajectory comparison implies that we compare whole trajectories of 2500 original nodes. We obtain missing $inter$-nodes for interpolated pose graph using linear interpolation with corresponding $base$-nodes.

As we see from Tab I, our Jacobian form ensures a feasible solution with sparsity factors of 4 and 8, decreasing the number of optimization state variables to 625 and 313 nodes out of the initial 2500. Further decimation with sparsity factors of 16 and 32 leads to unusable results due to the far distance between interpolation base nodes and the inability to restore the trajectory's circularity using linear interpolation in such conditions.

C. Time-Continuous LIDAR Odometry

In the last experiment, we perform a qualitative performance estimation of our Jacobian formulation for Lidar-only odometry problem solving it using a modified version of the Continuous-Time Iterative Closest Point (CT-ICP) algorithm presented in [11] for short sequences of KITTI-raw dataset [18].

1) Problem description: The main feature of method [11] is a combination of continuity in the scan matching and discontinuity between scans.

They use split rotation and translation representation and achieve time continuity for the scan matching by using spher-

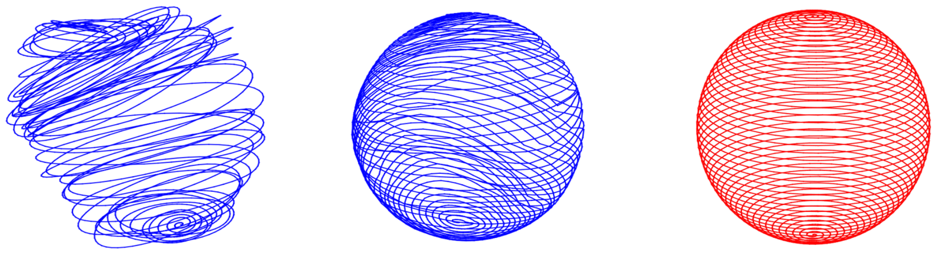
(a) Before optimization (b) Interpolated Pose SLAM results (c) Classic Pose SLAM results

Fig. 2. Visual representation of the full trajectory Interpolated Pose SLAM results with sparsity factor $\delta = 4$ on Sphere data. Here (a) - depicts an initial state of the graph, (b) - Interpolated Pose SLAM results, (c) - Classic Pose SLAM results

ical linear interpolation (slerp) for rotations and standard linear interpolation for translations.

We can very shortly describe their time-continuous scan matching procedure (for more details, please, refer to original paper [11]) as minimizing the following point-to-plane residual function with respect to starting and ending scan poses $\mathbf{T} = (T_b, T_e) \in SE(3)^2$ for each i-th combination of sample point p_i^L, map point q_i^W and related neighborhood normal n_i:

$$r_i[\mathbf{T}] = \left(p_i^W[\mathbf{T}] - q_i^W\right) \cdot n_i \tag{26}$$

$$p_i^W[\mathbf{T}] = T^{\tau_i}[\mathbf{T}]p_i^L \tag{27}$$

where T^{τ_i} - interpolated pose for τ_i timestamp of scanning.

While the original work performs interpolation using split spherical linear interpolation (slerp) for rotations and linear interpolation for translations, we replace it by direct interpolation in manifold and evaluate the performance of our proposed analytical Jacobian on short sequences of ~ 100 frames from KITTI-raw [18] dataset.

2) Experiment results: We use a mrob C++ library [21] as our optimization backbone providing it with analytical Jacobian and replacing the parts of the original pipeline that uses Ceres-solver [22] framework alongside Jacobians obtained using its auto differentiation module.

We provide the qualitative visual results for short ~ 100 frames long subsequences from KITTI-raw [18] dataset scenes at Fig. 3. The green trajectory depicts the result obtained with our Jacobian formulation, while the white one describes the original CT-ICP result. The proposed analytical Jacobian approximation results in a trajectory that is generally very close to the original result (top image) but does not replicate it exactly (bottom image).

Our analytical Jacobian approximation results in stable trajectory and consistent maps for short sequences. Testing it with longer trajectory subsamples or reaching a closer match with the original CT-ICP results implies implementing additional constraints described in the original paper that ensure discontinuity consistency between sequential frames, which is out of the scope of the current work.

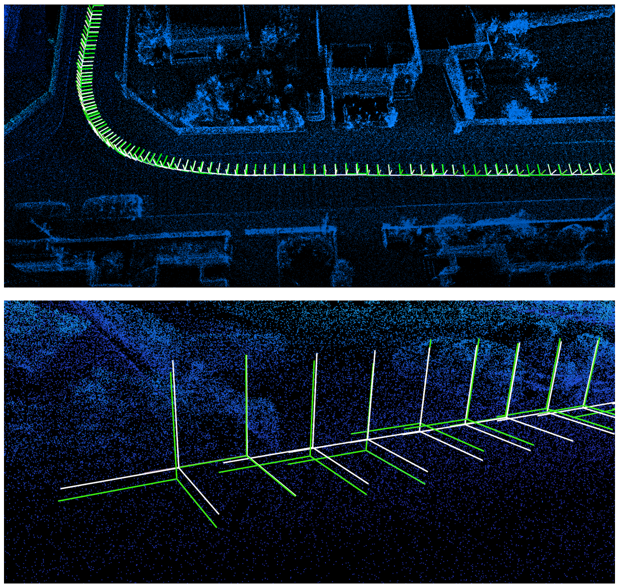
Fig. 3. Lidar odometry results obtained using our proposed Jacobian approximation (green) and original CT-ICP method (white) on short sequences of KITTI-raw dataset

V. CONCLUSIONS

We have presented an analytical Jacobian approximation of interpolated poses on $SE(3)$ that can be used for tasks involving the direct optimization of trajectories in 3D. We provide its derivation and report its performance on several synthetic and real-world datasets and problems, showing its consistency in achieving correct results.

REFERENCES

[1] F. Dellaert and M. Kaess, "Square root sam: Simultaneous localization and mapping via square root information smoothing," *Int. J. Rob. Res.*, vol. 25, no. 12, p. 1181–1203, dec 2006. [Online]. Available: https://doi.org/10.1177/0278364906072768

[2] A. G. Cunningham, V. Indelman, and F. Dellaert, "Ddf-sam 2.0: Consistent distributed smoothing and mapping," *2013 IEEE International Conference on Robotics and Automation*, pp. 5220–5227, 2013.

[3] M. Kaess, A. Ranganathan, and F. Dellaert, "isam: Incremental smoothing and mapping," *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1365–1378, 2008.

[4] X. Wang, R. J. Marcotte, G. Ferrer, and E. Olson, "Apriisam: Real-time smoothing and mapping," *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2486–2493, 2018.

[5] J. Huai, Y. Zhuang, Q. Yuan, and Y. Lin, “Continuous-time spatiotemporal calibration of a rolling shutter camera-imu system,” *IEEE Sensors Journal*, vol. 22, pp. 7920–7930, 2021.

[6] P.-A. Gohard, B. Vandeportaele, and M. Devy, “Spatiotemporal optimization for rolling shutter camera pose interpolation,” in *Computer Vision, Imaging and Computer Graphics – Theory and Applications*, A. P. Cláudio, D. Bechmann, P. Richard, T. Yamaguchi, L. Linsen, A. Telea, F. Imai, and A. Tremeau, Eds. Cham: Springer International Publishing, 2019, pp. 154–175.

[7] J. Dong, B. Boots, and F. Dellaert, “Sparse gaussian processes for continuous-time trajectory estimation on matrix lie groups,” 2017. [Online]. Available: https://arxiv.org/abs/1705.06020

[8] Y. Kapushev, A. Kishkun, G. Ferrer, and E. Burnaev, “Random fourier features based slam,” in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 6597–6602.

[9] J. Tirado and J. Civera, “Jacobian computation for cumulative b-splines on se(3) and application to continuous-time object tracking,” *IEEE Robotics and Automation Letters*, vol. 7, pp. 1–8, 07 2022.

[10] E. Mueggler, G. Gallego, H. Rebecq, and D. Scaramuzza, “Continuous-time visual-inertial odometry for event cameras,” *IEEE Transactions on Robotics*, vol. PP, pp. 1–16, 08 2018.

[11] P. Dellenbach, J.-E. Deschaud, B. Jacquet, and F. Goulette, “Ct-icp: Real-time elastic lidar odometry with loop closure,” 2021.

[12] Z. Yuan, F. Lang, and X. Yang, “Sr-lio: Lidar-inertial odometry with sweep reconstruction,” 2022. [Online]. Available: https://arxiv.org/abs/2210.10424

[13] C. Park, P. Moghadam, S. Kim, A. Elfes, C. Fookes, and S. Sridharan, “Elastic lidar fusion: Dense map-centric continuous-time slam,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 1206–1213.

[14] S. Ceriani, C. Sánchez, P. Taddei, E. Wolfart, and V. Sequeira, “Pose interpolation slam for large maps using moving 3d sensors,” in *IEEE/RSJ international conference on intelligent robots and systems (IROS)*, 2015, pp. 750–757.

[15] C. Park, P. Moghadam, J. L. Williams, S. Kim, S. Sridharan, and C. Fookes, “Elasticity meets continuous-time: Map-centric dense 3d lidar slam,” *IEEE Transactions on Robotics*, vol. 38, no. 2, pp. 978–997, 2022.

[16] 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)*, 2018, pp. 381–389.

[17] T. D. Barfoot, *State Estimation for Robotics*. Cambridge University Press, 2017.

[18] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? the kitti vision benchmark suite,” in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3354–3361.

[19] P.-A. Absil, R. Mahony, and R. Sepulchre, *Optimization algorithms on matrix manifolds*. Princeton University Press, 2009.

[20] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, “G2o: A general framework for graph optimization,” in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 3607–3613.

[21] “Mrob: Mobile robotics library,” https://github.com/g-ferrer/mrob, accessed: 2023-02-01.

[22] S. Agarwal, K. Mierle, and T. C. S. Team, “Ceres Solver,” 3 2022. [Online]. Available: https://github.com/ceres-solver/ceres-solver