X-SLAM: Scalable Dense SLAM for Task-aware Optimization using CSFD

ZHEXI PENG, State Key Lab of CAD&CG, Zhejiang University, China
YIN YANG, University of Utah, United States of America
TIANJIA SHAO*, State Key Lab of CAD&CG, Zhejiang University, China
CHENFANFU JIANG, University of California, Los Angeles, United States of America
KUN ZHOU*, State Key Lab of CAD&CG, Zhejiang University, China

Fig. 1. Real-time robot active scanning and reconstruction with semantic segmentation based on our X-SLAM system. With X-SLAM, robots can carry out automatic navigation and scanning within an unknown environment (left), and obtain a reconstruction with semantic segmentation (middle). The scanning process is presented on the right. We propose the first real-time differentiable dense SLAM system utilizing CSFD. By integrating it with a neural network, we facilitate robot active scanning and scene comprehension with semantic awareness.

We present X-SLAM, a real-time dense differentiable SLAM system that leverages the complex-step finite difference (CSFD) method for efficient calculation of numerical derivatives, bypassing the need for a large-scale computational graph. The key to our approach is treating the SLAM process as a differentiable function, enabling the calculation of the derivatives of important SLAM parameters through Taylor series expansion within the complex domain. Our system allows for the real-time calculation of not just the gradient, but also higher-order differentiation. This facilitates the use of high-order optimizers to achieve better accuracy and faster convergence. Building on X-SLAM, we implemented end-to-end optimization frameworks for two important tasks: camera relocalization in wide outdoor scenes and active robotic scanning in complex indoor environments. Comprehensive evaluations on public benchmarks and intricate real scenes underscore the improvements in the accuracy of camera relocalization and the efficiency of robotic navigation achieved through our task-aware optimization. The code and data are available at https://gapszju.github.io/X-SLAM.

*Corresponding authors: Tianjia Shao (tjshao@zju.edu.cn) and Kun Zhou (kunzhou@acm.org).

Authors' addresses: Zhexi Peng, zhexipeng@zju.edu.cn, State Key Lab of CAD&CG, Zhejiang University, Hangzhou, China; Yin Yang, yin.yang@utah.edu, University of Utah, Salt Lake City, United States of America; Tianjia Shao, tjshao@zju.edu.cn, State Key Lab of CAD&CG, Zhejiang University, Hangzhou, China; Chenfanfu Jiang, chenfanfu.jiang@gmail.com, University of California, Los Angeles, Los Angeles, United States of America; Kun Zhou, kunzhou@acm.org, State Key Lab of CAD&CG, Zhejiang University, Hangzhou, China.

1 INTRODUCTION

Simultaneous Localization and Mapping (SLAM) is a key technology in a variety of fields including augmented reality, robot navigation, autonomous driving, etc. While the primary objective of most SLAM algorithms is to reconstruct 3D scenes through RGB/RGBD cameras [Dai et al. 2017; Newcombe et al. 2011; Whelan et al. 2015], they have been deeply integrated with an array of downstream tasks, where the 3D geometry of the scene is more or less an "intermediate product". For instance, Liu et al. [2018] and Gottipati et al. [2019] combine SLAM with the robot navigation task based on scene understanding. Wu et al. [2021] present a SLAM-based framework for complex robotic manipulation and autonomous perception tasks. McCormac et al. [2017] and Zhang et al. [2018] also map environments containing semantics for robot intelligence and

user interaction. Stenborg et al. [2018] propose a cross-seasonal localization task after the SLAM 3D reconstruction.

Typically, these methods treat the SLAM system and the downstream task as separate modules. That said, the SLAM system first produces the results of 3D reconstruction and camera poses, and the downstream application makes the subsequent decisions based on the SLAM outputs. The quality, reliability, robustness, and performance of those tasks rely on the accuracy of the SLAM result. Once errors accumulate during the scanning process, the downstream task has to be processed based on the erroneous SLAM results. More recently, researchers propose to express the SLAM system as a differentiable function [Jatavallabhula et al. 2020], which can be combined with downstream tasks in an end-to-end manner allowing the task-based error signals to be back-propagated all the way to the raw sensor observations. In this way, the 3D reconstruction can be learned to optimize the task performance. For instance, SLAM-net [Karkus et al. 2021] replaces the SLAM modules with neural networks to enable differentiability. Its generalization ability is naturally limited to seen scenes. As a seminal work, ∇SLAM [Jatavallabhula et al. 2020] first presents a fully differentiable 3D dense SLAM system that thoroughly addresses the non-differentiable steps in traditional SLAM frameworks by offering mathematically differentiable approximations while preserving accuracy. This paradigm relies on the computation graph [Paszke et al. 2019]. The computational graph of all tracking/mapping functions needs to be maintained at every frame, leading to a continuous increase in memory consumption. Based on the public code, the SLAM procedure will crash after 60 iterations on PointFusion with image resolution of 640×480. The efficiency is not sufficient for the practical application due to the backward propagation cost in large computation graphs. These limitations restrict the capability of differentiable SLAM on real applications, where long-duration and real-time scanning is common.

In this paper, we present X-SLAM, a real-time and differentiable dense SLAM system that allows task-oriented optimization of SLAM parameters based on the loss backpropagation. Inspired by [Jatavallabhula et al. 2020], we address several fundamental challenges to make differentiable SLAM practical. First, we need to find a memory-efficient way to compute the differentials, so as to avoid the memory problem caused by the increasing computation graph. Second, the differential computation should be in real time to match the frame rate of SLAM. Third, in many real-world tasks, the first-order optimization may get stuck in the local minimum, prohibiting high accuracy of the end-to-end optimization. Our method should support the computation of high-order differentials.

We employ the complex-step finite difference (CSFD) method [Shen et al. 2021] to make a fully differentiable SLAM system for task-aware optimization, avoiding the expensive cost of computation graphs. CSFD calculates the differential by Taylor series expansion in the complex domain. We treat the SLAM procedure as a function f and represent the SLAM input x as a complex number x^* with a small non-zero imaginary perturbation, and the differential can be directly obtained by taking the imaginary part of $f(x^*)$ without the need of maintaining the computation graph. In order to achieve real-time computation of a differential, we utilize CSFD to automatically track the differentials of all variables w.r.t. perturbed inputs during the SLAM process, which naturally eliminates the backpropagation step in the computation graph. CSFD can also be generalized using high-order complex numerics to compute high-order differentials such as Hessian matrix. This allows us to exploit more powerful optimizers to achieve high accuracy.

We demonstrate X-SLAM in two important tasks: robot active scanning and reconstruction of indoor scenes, and camera relocalization in outdoor scenes. For both tasks, we develop task-aware objective functions and optimization pipelines to optimize SLAM parameters based on the loss backpropagation. We conduct careful comparisons on diverse public benchmark datasets and evaluate the performance of our approach in multiple challenging real scenes. The comparisons show that the X-SLAM based framework outperforms the state-of-the-art methods qualitatively and quantitatively.

2 RELATED WORK

Task-aware SLAM. SLAM within a commodity RGBD camera has been extensively studied. Many excellent RGBD SLAM algorithms [Mur-Artal et al. 2015; Newcombe et al. 2011; Whelan et al. 2015] have been proposed and the latest SLAM methods [Dai et al. 2017; Xu et al. 2022] can achieve impressive reconstruction and localization accuracy. In recent years, higher-level information such as semantics in the SLAM process has received increasing attention. Huang et al. [2021a] propose a novel convolution operation on supervoxels which can fuse the multi-view 2D features and 3D features during the online reconstruction. Liu et al. [2022] achieve online 3D joint semantic and instance segmentation based on an incremental sparse convolutional network. On the other hand, Salas-Moreno et al. [2013] exploit an object database to improve the accuracy of localization and reconstruction quality. Gümeli et al. [2023] present a semantic and object-centric camera pose estimator for RGB-D SLAM pipelines to complete robust registration. Ma et al. [2022] present a novel concept called virtual correspondences to establish geometric relationships between little overlap images. However, they do not apply those segmentations toward explicit downstream tasks. Jatavallabhula et al. [2020] propose a fully differentiable SLAM framework, called ∇SLAM, based on the computational graph and points out that the combination of differentiable SLAM and gradient-based optimization can achieve task-aware SLAM systems. X-SLAM is inspired by ∇SLAM. We avoid the fundamental limitations of ∇SLAM, which is based on automatic differentiation (AD), and propose CSFD-based differentiation to achieve more efficient and memory-economic differentiation.

Derivatives evaluation. Forward difference (FD) is commonly used numerical differentiation method. A drawback of FD is that the numerical errors introduced during the subtraction process can significantly impact the accuracy. CSFD addresses numerical stability issues by extending the calculations to the complex domain and adding perturbations to the imaginary part. Extending the function to the *dual domain* can also achieve similar effects. In the dual domain, a *dual number* $d = x + y\epsilon$ has the real part x and a dual part $y\epsilon$. Dual numerics define $\epsilon^2 = 0$ so $f(x_0 + h\epsilon) = f(x_0) + f'(x_0) \cdot h\epsilon$ because the higher-order terms of ϵ are vanished. High-order differentials can be computed by *hyper-dual number* [Aguirre-Mesa

et al. 2020; Cohen and Shoham 2016; Fike and Alonso 2011]. Different from CSFD, the derivative is extracted by setting $h = 1$ as: $f'(x_0) = \text{Du}(f(x_0 + \epsilon))$, so the dual number method is step-size-independent and exact to machine precision. However, Aguirre-Mesa et al. [2020] show that CSFD can achieve around 10^{-8} relative error for step size $h = 10^{-8}$, and it has better compatibility with other computational libraries such as Eigen and cuBLAS. Therefore, we use CSFD to compute differentials.

Camera relocalization. Camera relocalization in known scenes is a 3D geometry task closely related to SLAM. Most existing methods accomplish relocalization through the process of feature detection [Sattler et al. 2011], description [DeTone et al. 2018], and matching [Sarlin et al. 2020]. For example, Sarlin et al. [2019, 2021] accomplishes relocalization through feature matching on 2D images from coarse to fine. Brachmann et al. [2017] predicts dense correspondences between the input image and the 3D scene space, using differentiable RANSAC to optimize the camera pose. On the other hand, some research indicates that neural networks can directly predict camera poses. PoseNet [Kendall et al. 2015] is the first method to directly regress the camera pose from a single image and recent works use attention and transformer to improve accuracy [Li and Ling 2022; Wang et al. 2020]. However, these methods rely on color features, making it difficult to complete relocalization in areas with missing or repetitive textures. Although some methods [Brachmann and Rother 2021; Tang et al. 2021] use depth images to increase geometric constraints, their accuracy is still limited. Thanks to X-SLAM, we reconstruct from query images and reference images respectively, calculate errors, build a differentiable objective function to optimize the pose parameters and achieve high-accuracy camera relocalization.

Robot active scanning. Active scanning and online reconstruction for robots in unknown environments is one of the important tasks of SLAM [Callieri et al. 2004; Zeng et al. 2020]. González-Baños and Latombe [2002] propose an exploration approach for selecting a new goal according to the maximum map expansion. Building upon this idea, many efforts have been made based on reconstructed maps to develop exploration strategies. For example, RH-NBV [Bircher et al. 2016] compute a random tree where branch quality is determined by the amount of unmapped space that can be explored and the first eddge is executed at every planning step. Xu et al. [2017] harness a time-varying tensor field conforming to the progressively reconstructed scene to guide robot movement. On the other hand, semantic information is also applied to provide high-level guidance. Liu et al. [2018] introduce an approach that interleaves between object analysis to identify the next best object for global exploration, and object-aware information gain analysis to plan the next best view for local scanning. Zheng et al. [2019] propose an online estimated discrete viewing score field (VSF) parameterized over the 3D space of 2D location and azimuth rotation to guide the robot. These methods depend on the mapping quality and tracking accuracy of SLAM. When errors accumulate, obvious mistakes will happen. We develop a novel active scanning approach which can accomplish both activate exploration and semantic segmentation, based on our differentiable SLAM system and object recognition with deep learning.

3 COMPLEX-STEP FINITE DIFFERENCE

To make the paper more self-contained, we start with a brief review of CSFD. We refer the reader to related literature e.g., see [Martins et al. 2003] for a more thorough discussion and analysis.

Given a differentiable function $f : \mathbb{R} \to \mathbb{R}$, we apply a small perturbation h at $x = x_0$. The perturbed function can be Taylor expanded as:

$$f(x_0 + h) = f(x_0) + f'(x_0) \cdot h + \mathbf{O}(h^2), \tag{1}$$

which leads to the forward difference of:

$$f'(x_0) = \frac{f(x_0 + h) - f(x_0)}{h} + \mathbf{O}(h) \approx \frac{f(x_0 + h) - f(x_0)}{h}. \tag{2}$$

The numerator of Eq. (2) evaluates the difference between two quantities of similar magnitude as we often want h to be as small as possible for a better approximation. Subtraction is numerically unstable [Higham 2002], and the so-called *subtraction cancellation* could occur when $f(x_0 + h)$ and $f(x_0)$ become nearly equal to each other. Subtraction between them would eliminate many leading significant digits, and the result after the rounding could largely deviate from the actual value of $f(x_0 + h) - f(x_0)$. Because of this limitation, the finite difference is poorly suited for estimating derivatives of functions when high accuracy is needed.

CSFD offers a different approach to numerical derivatives. Instead of perturbing the function in the real domain, CSFD makes the perturbation an imaginary quantity, i.e., hi, and the corresponding Taylor expansion becomes:

$$f^*(x_0 + hi) = f^*(x_0) + f^{*'}(x_0) \cdot hi + \mathbf{O}(h^2). \tag{3}$$

Here f^* suggests the function is "promoted" to the complex domain, and its first-order derivative can be approximated by extracting the imaginary parts of Eq. (3):

$$f'(x_0) = \frac{\text{Im}(f^*(x_0 + hi))}{h} + \mathbf{O}(h^2) \approx \frac{\text{Im}(f^*(x_0 + hi))}{h}. \tag{4}$$

Compared with Eq. (2), it is noted that CSFD has a second-order convergence ($\mathbf{O}(h^2)$) as h approaches zero, and it does not suffer from the subtraction cancellation. In other words, we can make h as small as needed to fully suppress the approximate error: when h is at the order of $\sqrt{\epsilon}$ (ϵ is the machine epsilon), CSFD becomes as accurate as the analytic derivative for a given floating point system.

CSFD can be generalized to tackle high-order differentiation by further lifting the perturbation to be a *multicomplex* quantity. A multicomplex number is defined recursively: the base cases are the real number $\mathbb{R}$ and the regular complex number $\mathbb{C}$, which are considered as the zero- and first-order multicomplex sets $\mathbb{C}^0$ and $\mathbb{C}^1$. The complex number set $\mathbb{C}^1$ extends the real set by adding an imaginary unit i as: $\mathbb{C}^1 = \{x + yi | x, y \in \mathbb{C}^0\}$, and the multicomplex number up to an order of n is defined as:

$$\mathbb{C}^n = \left\{ z_1 + z_2 i_n | z_1, z_2 \in \mathbb{C}^{n-1} \right\}. \tag{5}$$

Following the derivation in [Lantoine et al. 2012], the Taylor series expansion of $f^\star$ under a multicomplex perturbation is:

$$f^\star(x_0 + hi_1 + \cdots + hi_n) = f^\star(x_0) + f^{\star(1)}(x_0) \cdot h \sum_{j=1}^{n} i_j$$

$$+ \frac{f^{\star(2)}(x_0)}{2} \cdot h^2 \left(\sum_{j=1}^{n} i_j\right)^2 + \cdots + \frac{f^{\star(n)}}{n!} \cdot h^n \left(\sum_{j=1}^{n} i_j\right)^n + \cdots. \tag{6}$$

Here, $f^{\star(n)}$ is the n-th-order derivative of $f^\star$. $\left(\sum i_j\right)^k$ can be expanded following the multinomial theorem, which contains products of mixed k imaginary directions for the k-th-order term. Because $\left(\sum i_j\right)^k \neq \left(\sum i_j\right)^l$ for $k \neq l$, Eq. (6) allows us to approximate an arbitrary-order derivative by extracting the corresponding imaginary combination, just as we did in Eq. (4). For instance, elements of the Hessian matrix (of a function $f(x, y) : \mathbb{R}^2 \to \mathbb{R}$) can be easily obtained as:

$$\begin{cases} \frac{\partial^2 f(x, y)}{\partial x^2} \approx \frac{\text{Im}^{(2)}\left(f(x + hi_1 + hi_2, y)\right)}{h^2}, \\ \frac{\partial^2 f(x, y)}{\partial y^2} \approx \frac{\text{Im}^{(2)}\left(f(x, y + hi_1 + hi_2)\right)}{h^2}, \\ \frac{\partial^2 f(x, y)}{\partial x \partial y} = \frac{\partial^2 f(x, y)}{\partial y \partial x} \approx \frac{\text{Im}^{(2)}\left(f(x + hi_1, y + hi_2)\right)}{h^2}. \end{cases} \tag{7}$$

CSFD lays out the foundation of our X-SLAM framework serving as the primary differentiation modality of the pipeline. Since the complex numerics are only for gradient/differentiation calculation, the imaginary part of the computation is always small (i.e., corresponding to a small variation induced by the input perturbation hi). In this situation, it is unnecessary to perform general-purpose complex numerics and operations. For instance, the multiplication and division between two complex numbers $z_1 = a_1 + b_1 i$ and $z_2 = a_2 + b_2 i$ can be simplified as:

$$z_1 z_2 = (a_1 a_2 - b_1 b_2) + (a_1 b_2 + a_2 b_1)i \approx a_1 a_2 + (a_1 b_2 + a_2 b_1)i, \tag{8}$$

and

$$\frac{z_1}{z_2} = \frac{a_1 a_2 + b_1 b_2}{a_2^2 + b_2^2} + \frac{b_1 a_2 - a_1 b_2}{a_2^2 + b_2^2} i \approx \frac{a_1}{a_2} + \frac{b_1 a_2 - a_1 b_2}{a_2^2} i. \tag{9}$$

We use such a simplification strategy for most elementary functions such as exponential functions, logarithmic functions, power functions, trigonometric functions, etc., which are used in the SLAM pipeline by discarding high-order products between two imaginary quantities. We encapsulate these implementations as a complex library using C++ on CPU and CUDA on GPU by overloading most floating-point operators to enable accelerated complex number computations.

To better understand computational overhead induced by complex promotion, we conduct a simple experiment to compare CSFD with AD and FD. We randomly sample x within the range of $[0, 1]$ and calculate the first-order numerical derivative of $f(x) = (e^x + x^3 + x)/(x+1)$ at x. The process is repeated one million times on the CPU. AD is implemented in Python with PyTorch, FD is implemented in C++ with STD library and CSFD is implemented in C++ with our complex library. We report the total time, memory usage, and

Table 1. **Comparison between inverse AD, FD and CSFD.** The timing information of the forward computation is also given in the parentheses. The results show that CSFD achieves better accuracy compared with FD, and outperforms inverse AD in computational efficiency

Method	Time	Memory	Relative error
AD	120.2s (39.8s)	1.53MB	-
FD	163.8ms (81.9ms)	14KB	6.49e-02
CSFD	227.2ms (227.2ms)	32KB	8.93e-08

relative error in Table 1. The relative error is defined as:

$$e_{relative} = |\frac{v_0 - v}{v}|, \tag{10}$$

where v is the analytic value, and v_0 is the numerical derivation. We regard the result of AD as the analytic value. The results indicate that both FD and CSFD outperform AD method in terms of time and memory performance because they do not require maintaining the computational graph. Although FD has a faster forward speed, the time difference or the performance difference between CSFD and FD is not significant because FD needs to compute the original function twice, and the accuracy of CSFD far exceeds that of FD.

It should also be noted that the time performance also depends on the input and output dimensions of the function. The derivative of a parameter needs one forward pass of the function, and the total computational cost linearly depends on the number of parameters. To this end, we slightly change the previous example by making the input of f a 10-dimension vector such that $f(y) = (e^y + y^3 + y)/(y+1)$, for $y = \|x\|$, and $x \in \mathbb{R}^{10}$. The total calculation time for AD, FD, CSFD increases to 137.72s, 1.05s, and 2.64s respectively. In this case, inverse AD can obtain derivatives for all parameters with one backpropagation process, making it a better choice for applications that differentiate with respect to a large number of parameters, such as deep learning.

4 X-SLAM

In this section, we walk through our X-SLAM pipeline with details. Similar to existing dense SLAM systems, Our X-SLAM consists of surface measurement, prediction, and camera pose estimation. With the help of CSFD, we aim to calculate derivatives at each step with improved scalability and efficiency.

As long as the imaginary perturbation is applied to the parameter of interest, we can easily obtain the corresponding derivative by extracting the imaginary part of the result (i.e., Eq. (4)). There are a few non-smooth computations that could potentially lead to ill-defined gradients, and some special treatments will be needed. Unless the gradient is needed, we could discard and ignore the real-part computation as in [Luo et al. 2019].

4.1 X-KinectFusion

Kinect Fusion (KF) [Newcombe et al. 2011] is one of the most used dense SLAM algorithms. When an RGBD frame $\{I_k, D_k\}$ is received from the sensor at time k, KF estimates the current camera pose $T_{g,k}$ and updates to the global fusion $\mathcal{M}$.

Surface measurement. Given the camera calibration matrix K, the depth map D_k is obtained by applying a bilateral filter [Tomasi and Manduchi 1998] to the raw depth map for noise reduction. Each image pixel $u = [u_x, u_y]^\top$ corresponds to a vertex in the current camera frame:

$$V_k(u) = D_k(u)K^{-1}\dot{u}. \tag{11}$$

Here V_k is the vertex map in the camera's local coordinate. $\dot{u}$ is the homogeneous vector $\dot{u} := (u^\top|1)^\top$. The normal vector of the vertex can be computed as:

$$N_k(u) = \frac{[V_k(u^{x+}) - V_k(u)] \times [V_k(u^{y+}) - V_k(u)]}{\|[V_k(u^{x+}) - V_k(u)] \times [V_k(u^{y+}) - V_k(u)]\|}. \tag{12}$$

Here $u^{x+} = [u_x+1, u_y]^\top$ and $u^{y+} = [u_x, u_y+1]^\top$. This computation is trivially parallelizable on the GPU as we launch one CUDA thread to process each pixel.

In general, the procedure of surface measure is a map of:

$$(V_k, N_k) \leftarrow \texttt{surface\_measure}(u, K, D_k). \tag{13}$$

We typically regard K accurate and constant and want to understand how the depth errors are propagated to $V_k(u)$ and $N_k(u)$, as well as in the follow-up computations. This information allows us to control the depth noise according to vertex and normal maps. Therefore, we apply the CSFD promotion to $D_k(u)$ as $D_k^*(u)$, making both $V_k(u)$ and $D_k(u)$ complex, i.e., $V_k^*(u)$ and $N_k^*(u)$ in Eqs. (11) and (12):

$$D_k^*(u) = D_k(u) + hi, \tag{14}$$

where we set $h = 1e - 8$ to match the CSFD approximation error to ϵ. The partial derivative of V_k and N_k w.r.t. depth variation can be obtained as:

$$\frac{\partial V_k(u)}{\partial D_k(u)} = \frac{\texttt{Im}(V_k^*(u))}{h}, \text{ and } \frac{\partial N_k(u)}{\partial D_k(u)} = \frac{\texttt{Im}(N_k^*(u))}{h}. \tag{15}$$

KF computes depth maps of different resolutions for multi-level ICP alignment. A coarser level depth is computed by applying Gaussian smoothing over $D_k(u)$. The perturbed hi is also Gaussian smoothed.

Ray casting. Ray casting is needed for surface prediction. Given the camera pose $T_{g,k}$, a ray is cast through the center of each pixel u in the image:

$$r(\alpha, u) = \alpha R_k K^{-1}\dot{u} + t_k, \text{ where } T_{g,k} = \begin{bmatrix} R_k & t_k \\ 0 & 1 \end{bmatrix} \in \mathbb{SE}(3). \tag{16}$$

We trace along all the rays by increasing the parameter α each time by $\Delta\alpha$: $\alpha_{m+1} \leftarrow \alpha_m + \Delta\alpha$. This corresponds to a point in the global frame $v_m = r(\alpha_m, u)$ whose TSDF value $F(v_m)$ is obtained via trilinear interpolation. A zero crossing is then confirmed once we have $F(v_{m+1}) < 0$ and $F(v_m) > 0$ as:

$$\alpha^0 = \alpha_m - \frac{\Delta\alpha F(v_m)}{F(v_{m+1}) - F(v_m)}, \tag{17}$$

which corresponds to a vertex $V_{g,k}(u)$ on the surface. Then the normal vector at this surface point can be computed as: $N_{g,k}(u) = \nabla F(r(\alpha^0, u))/\|\nabla F(r(\alpha^0, u))\|$.

This procedure suggests that the prediction of a surface point depends on the camera pose $T_{g,k}$:

$$(V_{g,k}, N_{g,k}) \leftarrow \texttt{surface\_prediction}(u, T_{g,k}). \tag{18}$$

To evaluate its differentiation w.r.t. the camera pose, we apply the complex promotion at the exponential map of the camera rotation:

$$T_{g,k}^* = \exp(\xi_k^*) = \exp\left(\begin{bmatrix} \phi_k^* \\ \rho_k^* \end{bmatrix}\right) = \begin{bmatrix} \exp(\phi_k^*) & \widetilde{R}_k^*\rho_k^* \\ 0 & 1 \end{bmatrix}. \tag{19}$$

We then use CSFD to compute the derivative w.r.t. ξ_k^* after promoting it to $\mathbb{C}^6$. $\phi_k^* \in \mathbb{C}^3$ is the promoted rotation vector such that:

$$R_k^* = \exp(\phi_k^*) = \cos\theta^* \texttt{Id} + (1 - \cos\theta^*)a^*a^{*H} + \sin\theta^*[a^*]. \tag{20}$$

Id is the 3 by 3 identity matrix. θ^* is the complex-perturbed rotation angle which can be obtained as:

$$\theta^* = \arccos\left(\frac{\texttt{tr}(R_k^*) - 1}{2}\right). \tag{21}$$

$a^* = \phi^*/\theta^*$ is the rotation axis, and a^{*H} is its conjugate transpose. $[a^*]$ denotes the skew-symmetric matrix of a^*. $\widetilde{R}^*$ is the rotation Jacobi defined as:

$$\widetilde{R}^* = \frac{\sin\theta^*}{\theta^*}\texttt{Id} + \left(1 - \frac{\sin\theta^*}{\theta^*}\right)a^*a^{*H} + \left(\frac{1 - \cos\theta^*}{\theta^*}\right)[a^*]. \tag{22}$$

The gradient of the predicted surface geometry can be obtained as:

$$\frac{\partial V_{g,k}}{\partial\xi_{k,i}} = \frac{\partial V_{g,k}}{\partial T_{g,k}} : \frac{\partial T_{g,k}}{\partial\xi_{k,i}} = \frac{\texttt{Im}\left(V_{g,k}^*\right)}{h},$$

$$\frac{\partial N_{g,k}}{\partial\xi_{k,i}} = \frac{\partial N_{g,k}}{\partial T_{g,k}} : \frac{\partial T_{g,k}}{\partial\xi_{k,i}} = \frac{\texttt{Im}\left(N_{g,k}^*\right)}{h}. \tag{23}$$

Here $\xi_{k,i}$ is the i-th perturbed component of ξ_k. CSFD tracks the initial perturbation hi along the computation. As a result, the chain rule in Eq. (23) is never explicitly evaluated (as opposed to AD), and we can conveniently calculate the gradient in an end-to-end manner.

The derivative w.r.t. pixel coordinate u is less intuitive as $u \in \mathbb{Z}^2$ is discrete. To this end, we generalize the domain of integer-value pixel index to $\mathbb{R}^2$ by bilinear interpolation. If the vertex depth significantly differs from its neighbors:

$$\sum_{u' \in \mathcal{N}(u)} |D_k(u) - D_k(u')| \geq \delta, \tag{24}$$

we do not perturb this coordinate to avoid noisy and misinformed derivatives.

Differentiable ICP. As soon as the vertex and normal maps are ready, we estimate the camera pose by applying multi-level ICP between $\{V_k, N_k\}$ and $\{V_{g,k-1}, N_{g,k-1}\}$. The ICP objective function is the summation of all the point-to-plane distances for all the credible surface measures:

$$E(\xi_k) = \sum \left\|\left(T_{g,k}V_k(u) - V_{g,k-1}(\hat{u})\right)^\top \cdot N_{g,k-1}(\hat{u})\right\|. \tag{25}$$

Here $\hat{u} = \left(KT_{g,k-1}^{-1}(\widetilde{T}_{g,k})_j V_k(u)\right)$. The function $\pi(\cdot)$ performs perspective projection. $(\widetilde{T}_{g,k})_j$ is the camera pose at the j-th iteration during ICP and $(\widetilde{T}_{g,k})_{j=0}$ is initialized with the previous frame pose. In original KF, the optimization of ICP requires linearization of the transformation matrix. Let α, β, γ be the three Euler angles, and assume that the pose change between adjacent frames is small. The

transformation matrix is approximated by setting $cos(\theta) \approx 1$ and $sin(\theta) \approx \theta$ as:

$$T_{g,k} = \begin{bmatrix} R_k & t_k \\ 0 & 1 \end{bmatrix} \approx \begin{bmatrix} 1 & \alpha & -\gamma & t_x \\ -\alpha & 1 & \beta & t_y \\ \gamma & -\beta & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (26)$$

This poses Eq. (25) as a linear least squares problem in the form of $Ax = b$, which can be solved by various methods such as the Normal Equation and Singular Value Decomposition.

The ICP process in the dense SLAM system is slightly different from generic ICP. In KF, the ICP process is executed on vertex maps and normal maps, and the corresponding vertex $V_{g,k-1}(\hat{u})$ is computed by perspective projection instead of searching for the nearest point. Therefore, ICP becomes differentiable as long as we extend the discrete pixel coordinate to $\mathbb{R}^2$, as in the case of ray casting. In general, differentiating an iterative optimization procedure is difficult. The computational graph is often unknown at the time of compiling as we do not know how many iterations will suffice. Most optimization techniques are sequential, and the results from previous iterations are needed in order to proceed. Therefore, it is a common practice in existing differentiable frameworks to assume a fixed iteration count for iterative procedures [Hu et al. 2019]. CSFD does not depend on the computational graph, and the derivative evaluation is agnostic on the global computation. Therefore, we can use any optimization algorithms to solve Eq. (25) without worrying about the iteration count. To this end, we implemented a second-order ICP procedure using Newton's method at each ICP step. The Hessian can be conveniently computed via Eq. (7), and we do not need to approximate trigonometric functions as in Eq. (26).

Surface Update. KF updates the global fusion $\mathcal{M}$ according to the camera pose $T_{g,k}$ and the depth map D_k. $\mathcal{M}$ represents a 3D TSDF volume. Let $p \in \mathbb{R}^3$ be a voxel in $\mathcal{M}$ to be reconstructed. Its truncated signed distance $F_k(p)$ and weight $W_k(p)$ are updated via:

$$F_k(p) = \frac{W_{k-1}(p)F_{k-1}(p) + W_{D_k}(p)F_{D_k}(p)}{W_{k-1}(p) + W_{D_k}(p)},$$
$$W_k(p) = W_{k-1}(p) + W_{D_k}(p). \quad (27)$$

Here, $F_{D_k}(p)$ and $W_{D_k}(p)$ are:

$$F_{D_k}(p) = \Psi\left(\|K^{-1}\dot{x}\|^{-1}\|t_{g,k} - p\| - L(x)\right), \quad (28)$$
$$W_{D_k}(p) = 1, \quad (29)$$

and we have x and Ψ defined as:

$$x = \pi(KT_{g,k}^{-1}p), \quad \Psi(\eta) = \begin{cases} \min\left(1, \dfrac{\eta}{\mu}\right)\text{sgn}(\eta) & \text{if } \eta \geqslant -\mu \\ \texttt{null} & \text{else.} \end{cases} \quad (30)$$

We use the bilinear interpolation function $L(x)$ in Eq. (28) to estimate the depth value instead of using the nearest-neighbor lookup in original KF to restore the smoothness of the function.

As in Eq. (19), we use ξ_k to encode the pose matrix $T_{g,k}$ and promote it to the complex form ξ_k^*, which transforms (F_k, W_k) into the complex quantities (F_k^*, W_k^*). The gradient of the function $(F_k, W_k) \leftarrow \texttt{surface\_update}(\mathcal{M}_{k-1}, T_{g,k}, D_k)$ can be computed
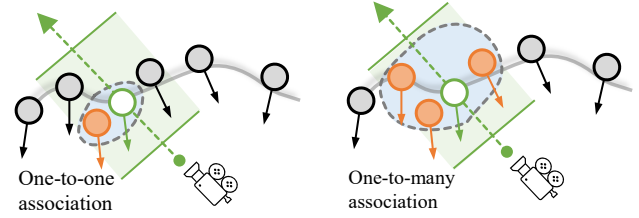

Fig. 2. **An illustration of the one-to-one association (left) and our one-to-multiple association (right).** The associated global model points (gray dots) of camera ray (green dashed line) is colored in orange.

as:

$$\frac{\partial F_k}{\partial \xi_{k,i}} = \frac{\text{Im}(F_k^*)}{h}, \quad \text{and} \quad \frac{\partial W_k}{\partial \xi_{k,i}} = \frac{\text{Im}(W_k^*)}{h}. \quad (31)$$

4.2 X-ElasticFusion and X-PointFusion

ElasticFusion (EF) and PointFusion (PF) are two classic dense SLAM algorithms. Unlike KF, they use *surfels* as the representation of the global map. The front-end odometry of EF is similar to that of PF. However, EF includes a back-end optimization part, introducing *random ferns* for loop detection and map optimization through the deformation graph. This section focuses on the CSFD-based differentiation of the front-end odometry to jointly discuss those two methods. Other steps such as the camera pose estimation are dealt with in a similar way as in KF.

In EF/PF, $\mathcal{M}$ is an unstructured set of global model points. At time stamp k, each point $p_{g,k}$ is associated with the position $v_{g,k}$, normal $n_{g,k}$, radius $r_{g,k}$, confidence counter $c_{g,k}$ and time stamp $t_{g,k}$. Given a new measurement $\{I_k, D_k\}$, vertex map V_k and normal map N_k are computed similar as in KF, i.e., Eqs (11) and (12). After that, the pixel is either, as a new point, inserted into $\mathcal{M}$ or merged with existing global model points. This includes two steps.

Data association. Given the estimated camera pose $T_{g,k}$ and its intrinsic matrix K, all global model points are rendered into an index map at the resolution 4 by 4 times higher than D_k, where the respective index i is stored. Each pixel in D_k thus corresponds up to 16 model points in this index map. We remove unqualified model point candidates for pixel u if: 1) its depth distance along the viewing ray passing through $V_{g,k}(u)$ is beyond $\pm\delta_{depth}$; 2) its normal deviates from $N_{g,k}(u)$ by more than δ_{norm}; 3) the distance to the latest measure $V_{g,k}(u)$ is farther than $\delta_{distance}$. Because the number of points in $V_{g,k}(u)$ is smaller than the total number of global model points, one-to-one point association as adopted in the original EF pipeline will generate a lot of vanished gradients at global model points. To reduce the sparsity of the gradient/Hessian, we use a one-to-many association strategy, which updates all the points with weight $w_{g,k}$ based on the Gaussian kernel function according to the distance from $V_{g,k}(u)$, i.e., see Fig. 2.

Surface update. If pixel u does not associate with existing model points, a new point is added to the global model. Otherwise, for all global model points corresponding to pixel u, the updated model

point $p_{g,k}$ is computed as:

$$v_{g,k} \leftarrow \frac{c_{g,k} v_{g,k} + w_{g,k} C_k(u) V_{g,k}(u)}{c_{g,k} + w_{g,k} C_k(u)},$$

$$n_{g,k} \leftarrow \frac{c_{g,k} n_{g,k} + w_{g,k} C_k(u) N_{g,k}(u)}{c_{g,k} + w_{g,k} C_k(u)},$$

$$c_{g,k} \leftarrow c_{g,k} + w_{g,k} C_k(u), \quad t_{g,k} \leftarrow k. \tag{32}$$

The confidence $C_k(u)$ is computed as in the vanilla PF:

$$C_k(u) = e^{-\frac{\gamma^2}{0.72}}. \tag{33}$$

Here γ is the normalized radial distance of $V_k(u)$ from the camera center. Clearly, $p_{g,k}$ is a function of the depth map and camera pose. We promote ξ_k to the complex form ξ_k^*, and $p_{g,k}$ is transformed into the complex form $p_{g,k}^*$ and the gradient can be computed with CSFD as:

$$\frac{\partial p_{g,k}}{\partial \xi_{k,i}} = \frac{\text{Im}(p_{g,k}^*)}{h}. \tag{34}$$

5 EVALUATION

We implemented X-KF on a desktop computer with an `intel i9 13900KF` CPU and an `nvidia RTX 4090` GPU. We implemented a (high-order) complex numerics library dedicated to CSFD differentiation, i.e., as discussed in § 3. Some linear algebra computations are based on the `Eigen` library on the CPU using the C++ version of our library as a template class, and most large-scale computations are implemented with `CUDA` on the GPU.

We also built an active scanning system on a customized robot platform with a 6-DOF articulated arm holding an `intel RealSense D435i` RGBD sensor. The robot has a built-in computer running a ROS system, which provides a package to enable robot manipulation, such as navigation and arm actions. A laptop computer with an `intel i7 10750H` CPU and `nvidia 2070` GPU to run X-EF and deep networks controls the robot through a wireless network.

5.1 X-KF Evaluation

Datasets. We conducted experiments on two datasets: *ICL-NUIM dataset* [Handa et al. 2014] and *TUM RGB-D dataset* [Sturm et al. 2012]. *ICL-NUIM dataset* is a synthetic dataset with rendered RGBD images and ground truth camera poses. We selected two sequences including synthetic noise (*lr kt1_n, lr kt2_n*) commonly used in the previous work to evaluate the performance of our pipeline. *TUM RGB-D dataset* is a dataset captured by a *Microsoft Kinect v1* with motion-captured camera poses as ground truth, which is widely used to evaluate the tracking accuracy of SLAM systems. We selected two sequences (*fr1/desk, fr1/xyz*) for evaluation.

Comparison with AD method. First, we demonstrate the advantages of CSFD compared to AD. We conduct tests on X-KF and ∇KF as in ∇SLAM [Jatavallabhula et al. 2020], which uses the backpropagation shipped with `PyTorch` as the major modality for computing differentiation. These two SLAM systems share the same parameters and the gradient $\frac{\partial F_k}{\partial \xi_{k,0}}$ is computed at each frame. We show the time performance and tracking accuracy in Table 2. Due to the huge GPU memory consumption of ∇KF, we can only run the first 100 frames of each sequence. On the other hand, CSFD is like a

Table 2. **Time performance and tracking accuracy of X-KF and ∇KF.** The tracking accuracy is measured with absolute trajectory error (ATE RMSE). Because the computational graph of AD method continuously grows in SLAM process, this experiment is performed on the first 100 frames for each sequence. The timing information (in millisecond) of the forward computation is also given in the parentheses. Even so, X-KF achieves around 10× speed of ∇KF and achieve similar tracking accuracy.

Sequence	Res.	X-KF Time (ms)	∇KF Time (ms)	X-KF ATE (m)	∇KF ATE (m)
fr1_desk	128^3	8.4 (8.4)	118.9 (54.1)	0.026	0.025
	256^3	9.5 (9.5)	154.6 (72.9)	0.022	0.024
	512^3	11.2 (11.2)	475.2 (137.9)	0.024	0.024
fr1_xyz	128^3	7.9 (7.9)	113.3 (63.5)	0.018	0.020
	256^3	10.1 (10.1)	155.6 (81.2)	0.018	0.019
	512^3	11.5 (11.5)	500.4 (160.3)	0.019	0.019

Table 3. **Comparison of tracking accuracy.** Our tracking accuracy (ATE RMSE) is lower because we have not yet incorporated global optimization, leading to the drift along longer trajectories. The measurement is in meters.

Sequence	X-KF	BundleFusion [Dai et al. 2017]	ElasticFusion [Whelan et al. 2015]
fr1_desk	0.053	0.020	0.016
fr1_xyz	0.020	0.010	0.011

forward AD which does not need backpropagation. CSFD shows clear advantages in terms of both efficiency and accuracy.

Comparison with traditional SLAM. To provide a more comprehensive and quantitative comparison, we compare the tracking accuracy between X-KF (at resolution of 512^3) and two traditional SLAM systems, BundleFusion [Dai et al. 2017] and ElasticFusion [Whelan et al. 2015] in Table 3. X-SLAM does not achieve the state-of-the-art performance because those methods incorporate a back-end global optimization module, which is currently not integrated into X-SLAM. We also compare the time and memory performance of X-KF with original KF in Table 4. The speed of X-KF is slightly slower than original KF due to the additional computational costs for complex numbers. The spatial complexity of CSFD reformulation is $\mathbf{O}(N)$. In the worst case, the memory consumption is doubled due to the added imaginary part. For example, the main GPU memory consumption in KF is the TSDF volume which stores color (4 bytes), weight (4 bytes), and TSDF value (4 bytes) in each voxel. To use CSFD, we additionally use a float number (4 bytes) as the imaginary part of TSDF value, which results in approximately 33% memory overhead. However, we believe that this result is comparable and X-KF provides additional differential information.

Simplified CSFD computation. We have briefly introduced our complex library based on simplified complex functions for CSFD in Section 3 and now we compare the time performance with other publicly available libraries. First, we compare the C++ version of our complex library with the C++ standard library on the CPU.

Table 4. **Comparison of time/memory performance.** Both the time and memory performance are slightly worse than that of original KF due to the extra complex number computing process.

Sequence	X-KF FPS	KF FPS	X-KF Memory (MB)	KF Memory (MB)
fr1_desk	82.9	101.5	2824	1937
fr1_xyz	92.6	113.5	2234	1582

Table 5. **Computation time (unit: ms) of our complex numerics library and C++ STD library.** We test a variety of common mathematical operations, and the results show that our complex number numerics library is faster. And as the complexity of the operations increases, our improvement becomes more apparent.

Methods	$z_1 \cdot z_2$	$z1 / z2$	$exp(z_1 + z_2)$	$sin(z_1 + z_2)$	$(z_1 + z_2)^3$
STD	26.92	39.94	58.61	81.58	155.08
Ours	24.32	30.45	47.48	67.06	102.44

We set $z_1 = 0.5 + hi$, $z_2 = -1.5 + hi$ and evaluate some common functions. Each function is executed one million times, and the total time is presented in Table 5. Also, we compare the CUDA version of our complex library with the nvidia's library, libcu++ on GPU across the *fr1/xyz* sequence of *TUM RGB-D dataset* for a 512^3 voxel reconstruction. X-KF implemented by our complex library can achieve 92.6 FPS, while X-KF implemented by libcu++ only achieves 59.4 FPS, resulting in 1.55 times slower.

High order ICP. Finally, we test the performance of ICP based on different optimization algorithms for pose estimation. We project a depth image from *lr kt1_n* of *ICL-NUIM dataset* to obtain the source point cloud, and the target point cloud is generated using a random transformation. We run point-to-plane ICP using three methods: Gradient Descent, Nonlinear-CG, and Newton's method. The optimization process is shown in Figure 3 and the results show that higher-order optimization methods have a significantly faster convergence speed.
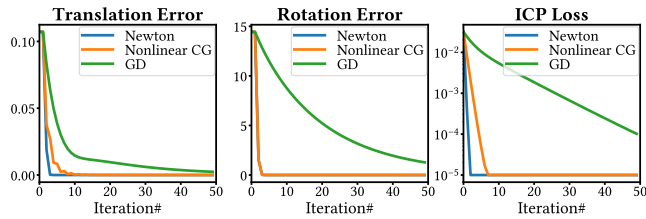

Fig. 3. **The optimization process of ICP based on different optimization methods.** Translation error (unit: m), rotation error (unit: °) and point-to-plane ICP loss are reported.

Also, we compare the tracking robustness of our X-KF and original KF on *lr kt2_n* of the *ICL-NUIM dataset* with different frame step. It can be found that although the tracking robustness is similar when frame step is 1, as the frame step increases, the performance of X-KF
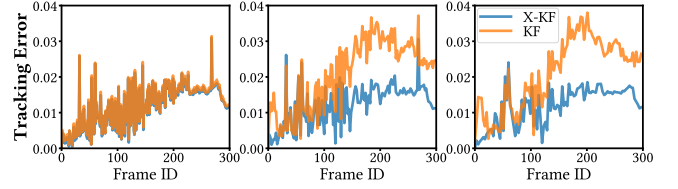
Fig. 4. **Absolute distance error (unit: m) during tracking.** Left: frame step = 1, Mid: frame step = 2, Right: frame step = 3. High-order ICP can achieve better tracking accuracy when the camera moves faster.

surpasses that of original KF. This is attributed to the following reasons. First, the convergence speed of Newton's method is higher than that of first-order optimization methods. Additionally, as the frame step increases, the pose differences between adjacent frames become larger, and the linear approximation of ICP in original KF works only when the pose differences between adjacent frames are small.

5.2 Camera Relocalization Based on X-KF

A precise relocalization of the camera is a key component in vision tasks. Given multiple reference images with the corresponding camera poses $\{I_r, T_{g,r}\}$, camera relocalization refers to estimating the camera pose $T_{g,q}$ of any query image I_q. CSFD is very suitable for camera pose optimization because it only requires perturbations on a small number (i.e., 6) of parameters. We show that an accurate camera relocalization can be conveniently obtained using depth maps based on X-KF. Let $\mathcal{M}_r$ be the reference TSDF volume. We compute a TSDF volume $\mathcal{M}_q$ such that $(F_q, W_q) \leftarrow$ surface_update$(\mathcal{M}_r, T_{g,q}, D_q)$ for a query RGBD image (I_q, D_q) and an estimated camera pose $T_{g,q}$. Then an objective function can be defined as:

$$E(T_{g,q}) = \sum_p \left(F_r(p) - F_q(p) \right)^2 . \tag{35}$$

The initial estimation for optimization can be obtained using any existing method. With X-KF, we can easily obtain the Hessian matrix and the Jacobian matrix of the objective function $E(T_{g,q})$, and use an optimizer, i.e. Newton's method, to minimize $E(T_{g,q})$. Since the TSDF values of the same voxel may vary at different viewpoints, we only use the images around the initial estimation to reconstruct $\mathcal{M}_r$.

Comparison. We compare our method with multiple state-of-the-art visual relocalization approaches. HLoc [Sarlin et al. 2019] and PixLoc [Sarlin et al. 2021] are feature-based methods for estimating the poses of the query images, which only require RGB images as input. VS-Net [Huang et al. 2021b] proposes scene-specific 3D landmarks and generates landmark segmentation maps to help with camera relocalization. DSM [Tang et al. 2021] is a scene-agnostic camera relocalization method that regresses the scene coordinates of the query images and the camera poses are solved by PnP algorithms. However, DSM requires the scene coordinates corresponding to the reference images. DSAC* [Brachmann and Rother 2021] is a combination of scene coordinate regression and differentiable RANSAC

for end-to-end training and designs multiple workflows for different input data structures. SC-wLS [Xin et al. 2019] proposes a network to directly regress the camera pose based on the pipeline of DSAC*.

Experiments on 7-Scenes dataset. We start our evaluation on *7-Scenes dataset* [Shotton et al. 2013] including small indoor scenes, which is commonly used to compare camera relocalization approaches. We use the aforementioned visual relocalization approaches to estimate the camera pose of the query image as the initial solution. For the following optimization, we implement both Gradient Descent and Newton's method. As Newton's method is sensitive to the initial solution, we set a threshold η when applying Newton's method for optimization. If the loss exceeds η, we use the gradient direction $-g$; instead, we use the Newton's direction $-H^{-1}g$. We report the statistical results in Table 7 and the initial/optimized HLoc trajectories in *7-Scenes dataset*. The results indicate that our method is applicable not only to HLoc, PixLoc and SC-wLS, which only use RGB images for relocalization, but also improves the accuracy of the results for DSM and DSAC* that already utilize depth information. This improvement is particularly significant in *Pumpkin*, *redkitchen*, and *stairs* where traditional methods do not perform well. In addition, we must point out the important role of higher-order differentials in the optimization process. It is evident that the Newton's method outperforms Gradient Descent in terms of convergence accuracy. On the other hand, we list the results of Gradient Descent based on the gradients computed by FD, and the results are worse than those of CSFD due to the low accuracy. We also report the optimization curves in Figure 6 and the results indicate that Newton's method outperforms Gradient Descent in terms of convergence speed as well.

Experiments on our dataset. We also build a large-scale RGBD dataset to validate our method in larger scale environments. Our dataset consists of two outdoor buildings and two office rooms. For each scene, we conduct two scans using a *Microsoft Kinect v2*. One scan reconstructed by BundleFusion is the reference set and the other is the query set. Because the depth images are not registered to the color images, we only evaluate HLoc, PixLoc and DSAC* (RGB mode) and each method is re-trained on the reference set. It should be noted that due to the issues such as drift during the scanning of large-scale scenes, we do not have the ground truth camera poses for the query set. Given the reference pointcloud $P_{g,r}$ and the query pointcloud P_q corresponding to the depth map D_q, i.e., Eq (11), we compute e_q to measure the accuracy of the camera pose $\mathsf{T}_{g,q}$:

$$e_q(\mathsf{T}_{g,q}) = \frac{1}{|P_q|} \sum_{p_q \in P_q} \min_{p_{g,r} \in P_{g,r}} ||\mathsf{T}_{g,q}p_q - p_{g,r}||. \qquad (36)$$

We consider the frames with $e_q > 0.1m$ as outliers and exclude them from statistical analysis. We report the result in Table 6. Similar to the results in *7-Scenes dataset*, our optimization method can still improve the accuracy of camera relocalization even in complex and large-scale environments. We also show the relocalization trajectories initialized from PixLoc and the reference model in Figure 7.

Table 6. **Camera relocalization results (unit: *cm*) in our large-scale datasets.** We use the nearest neighbor distance between the projected query depth map and the reference model. The initial/GD optimized/Newton optimized results are reported.

Scene	HLoc	PixLoc	DSAC*
Outside A	2.34/1.46/**1.33**	2.75/1.75/**1.58**	3.36/1.95/**1.78**
Outside B	1.90/1.53/**1.47**	2.20/1.75/**1.67**	3.38/2.33/**2.17**
Office A	3.03/1.84/**1.72**	3.19/2.12/**1.99**	3.30/2.43/**2.20**
Office B	3.02/2.32/**2.21**	3.69/3.15/**3.02**	Not converge

5.3 Robot Active Scanning Based on X-EF

Active scanning and online reconstruction by robots is another important downstream task of SLAM. We demonstrate that, by leveraging X-EF, we can combine this task with neural networks to achieve better performance. We follow the pipeline in [Liu et al. 2018]:

(1) *Object Segmentation.* Robots perform real-time scene reconstruction using X-EF and then segment the point cloud to obtain a series of objects.
(2) *Target Object Decision.* Robots evaluate these objects based on a set of criteria such as distance, size, orientation, etc., and choose the optimal scanning target, called the Next Best Object (NBO)
(3) *Target View Decision.* Once the NBO is determined, the object point cloud P_O from X-EF is sent to a pre-trained PointNet [Qi et al. 2017] network to predict a label score S, and the Next Best View (NBV) is defined as the viewport that maximizes the recognition score.
(4) *Repeating Scans.* Once the NBV is determined, the robot is instructed to move to the new viewpoint. PointNet is then re-evaluated, assigning scores to the objects based on the new perspective. This process continues until the recognition score exceeds the threshold, and the system returns to select another NBO.

In our system, the NBV selection process aims to maximize the recognition score S. An objective function can be defined as:

$$E(\mathsf{T}_{g,k}) = -S = -\mathtt{point\_net}(P_O). \qquad (37)$$

Using X-EF, we can calculate the differentiation of S w.r.t. the camera pose:

$$\frac{\partial S}{\partial \xi_{k,i}} = \sum_{p_O \in P_O} \frac{\partial S}{\partial p_O} : \frac{\partial p_O}{\partial \xi_{k,i}}. \qquad (38)$$

Here the network gradient $\frac{\partial S}{\partial p_O}$ is computed using AD method, while the SLAM gradient $\frac{\partial p_O}{\partial \xi_{k,i}}$ is calculated by X-EF. Similar to camera relocalization in Section 5.2, CSFD can quickly calculate the SLAM gradient for the 6DOF camera pose. Since computing higher-order differentials is challenging with AD method, and the dimension of ξ is only 6, we employ finite differences based on gradients to compute the Hessian matrix.

Table 7. **Camera relocalization results in *7-Scenes dataset*.** We report the median translation (unit: *cm*), rotation (unit: °) errors and the average recall at (5*cm*, 5°). Whether these methods consider depth information or not, our method can further enhance accuracy, especially Newton's method based on higher-order derivatives.

Method	Optimization	Chess	Fire	Heads	Office	Pumpkin	Redkitchen	Stairs
HLoc [Sarlin et al. 2019]	Init	2.41/0.85/92.3	2.28/0.91/89.3	1.08/0.74/96.3	3.12/0.90/76.7	4.83/1.27/51.8	4.21/1.39/58.9	5.23/1.47/47.3
	GD (FD)	2.32/0.82/93.3	2.22/0.89/90.9	1.02/0.75/97.0	3.09/0.91/77.8	4.52/1.26/54.4	4.02/1.31/61.2	5.25/1.43/45.4
	GD	1.68/0.56/98.8	1.94/0.78/93.3	0.90/0.61/98.6	2.55/0.80/86.1	3.89/1.03/61.0	3.26/1.23/74.8	4.96/1.27/50.7
	Newton	**1.15/0.37/99.3**	**1.19/0.48/99.0**	**0.79/0.47/99.0**	**1.46/0.47/99.0**	**2.10/0.69/74.8**	**2.37/0.86/83.3**	**2.62/0.70/81.2**
PixLoc [Sarlin et al. 2021]	Init	2.37/0.81/91.0	1.90/0.78/86.5	1.26/0.86/87.4	2.57/0.79/81.7	4.11/1.19/59.0	3.41/1.23/67.5	4.49/1.22/53.4
	GD (FD)	2.31/0.79/91.4	1.86/0.76/86.7	1.13/0.83/87.5	2.53/0.79/82.8	3.37/1.19/68.1	3.33/1.22/68.7	4.69/1.19/53.2
	GD	1.65/0.55/96.0	1.61/0.74/87.5	1.01/0.65/90.0	2.15/0.71/85.3	3.43/0.99/66.7	2.93/1.11/75.8	4.39/1.06/55.5
	Newton	**1.11/0.37/96.5**	**1.15/0.46/89.2**	**0.82/0.50/96.4**	**1.47/0.46/95.0**	**2.10/0.69/73.7**	**2.50/0.91/79.1**	**3.18/0.75/65.4**
VS-Net [Huang et al. 2021b]	Init	1.57/0.52/98.9	1.91/0.82/96.2	1.20/0.72/98.7	2.14/0.60/91.5	3.84/1.05/64.1	3.59/1.08/72.7	2.79/0.77/93.4
	GD (FD)	1.54/0.53/99.0	1.86/0.78/96.5	0.96/0.74/99.4	2.07/0.62/92.5	3.57/1.05/65.7	3.44/1.03/74.6	2.78/0.77/92.9
	GD	1.21/0.41/**99.6**	1.46/0.67/98.5	0.77/0.61/**100**	1.78/0.59/95.1	3.44/1.01/67.3	2.82/1.02/85.5	2.58/0.74/94.5
	Newton	**0.96/0.35/**99.5	**0.92/0.41/99.4**	**0.48/0.40/100**	**1.27/0.41/98.5**	**2.62/0.75/76.5**	**2.32/0.87/87.0**	**1.93/0.60/98.2**
DSAC* [Brachmann et al. 2017]	Init	1.01/0.44/98.9	1.12/0.53/99.1	0.98/0.87/**100**	1.19/0.48/**99.9**	1.97/0.67/80.9	2.09/0.83/92.1	2.62/0.78/91.2
	GD (FD)	0.99/0.43/99.0	1.13/0.54/98.9	0.98/0.81/**100**	1.26/0.48/99.1	2.01/0.66/80.4	2.15/0.77/91.5	2.67/0.78/91.6
	GD	0.85/**0.34**/99.4	0.95/0.45/99.3	0.76/0.63/**100**	**1.14**/0.44/99.4	1.67/0.51/86.4	1.64/0.54/93.8	2.58/0.78/**94.2**
	Newton	0.93/**0.34**/99.5	**0.85/0.37/99.7**	**0.50/0.42/100**	1.24/**0.41**/99.7	**1.20/0.38/89.6**	**1.04/0.36/98.4**	**1.89/0.60**/93.6
DSM [Tang et al. 2021]	Init	**2.01**/0.70/94.7	2.62/0.82/0.88	1.47/0.84/95.4	3.53/0.82/70.7	5.21/1.11/47.5	4.80/1.12/52.2	4.89/1.36/50.9
	GD (FD)	2.07/0.76/94.3	2.60/0.87/88.2	1.43/0.92/95.5	3.46/0.79/72.3	4.87/1.20/51.0	4.71/1.15/53.6	4.86/1.16/67.2
	GD	2.13/0.66/96.6	2.04/0.75/95.2	1.26/0.78/97.2	3.01/0.82/79.2	4.08/1.09/61.3	3.96/1.02/64.8	3.27/0.86/76.0
	Newton	2.02/**0.60/96.8**	**1.71/0.67/96.1**	**1.09/0.71/97.5**	**2.72/0.76/83.4**	**3.06/0.86/74.6**	**2.96/0.83/75.9**	**2.66/0.62/82.5**
SC-wLS [Xin et al. 2019]	Init	3.03/0.76/78.9	4.19/1.08/55.0	2.86/1.92/60.7	5.18/0.86/0.48	7.29/1.27/28.7	8.27/1.44/26.4	12.1/2.43/15.7
	GD (FD)	2.75/0.82/80.2	3.91/1.15/56.3	2.80/1.84/60.1	4.52/0.96/54.7	6.72/1.41/36.4	7.45/1.55/32.3	11.9/2.43/19.0
	GD	1.45/0.50/89.7	2.28/0.90/65.4	1.41/1.12/**61.3**	2.66/0.79/76.9	4.65/1.24/52.7	5.01/1.44/49.7	11.2/2.22/26.8
	Newton	**1.04/0.38/90.9**	**1.22/0.52/70.9**	**0.95/0.75/61.3**	**1.56/0.51/83.4**	**3.27/0.85/59.8**	**3.24/1.06/62.7**	**9.55/1.72/39.1**

Performance of object recognition. We first evaluate the performance of NBV-based single object recognition. To provide a quantitative evaluation, we compare our method with [Liu et al. 2018] in the robotic simulation environment of *Gazebo* running on top of ROS. We pre-train a PointNet network on *ModelNet40 dataset* [Wu et al. 2015] and choose several object models from the test split to evaluate. For each model, we scale it to an appropriate size and add random perturbations to the horizontal orientation and position. We limit the scanning trajectory length to 5*m*, and record the recognition score corresponding to the ground truth label every 0.2*m*. 20 models are selected for each category, and the average scores are reported in Figure 8. Please note that we compare the moving distance for achieving the same level of recognition accuracy, which is more appropriate because we find there may be a significant distance between two viewpoints. Our results are better than those of [Liu et al. 2018] in most categories. On the other hand, although both first-order and second-order optimization end up with similar scores in most cases, second-order optimization is more stable and converges faster.

Performance of virtual scene scanning. Figure 9 shows robot activate scanning process in some virtual scenes from *3D-FRONT dataset* [Fu et al. 2021]. It should be noted that since [Liu et al. 2018] did not provide the 3D model database, we build our own model database using *ModelNet40 dataset* for comparison. All methods are compared for achieving the same level of recognition completeness. The scanning process is stopped when all objects are recognized (with score > 0.8). We draw the scanning trajectory, and the trajectory length is also reported. It can be observed that our method achieves simpler and shorter paths. Although Gradient Descent can get a good moving direction, it is limited by overshoot, which sometimes makes the robot move back and forth. In contrast, Newton's method can solve this problem, improving the scanning path.

Performance of real scene scanning. We test our system by scanning five unknown scenes, including a resting room, a living room and three outdoor terraces. Figure 10 shows the reconstruction results in these scenes. The results show that the robot can actively complete scene reconstruction and recognize most objects.
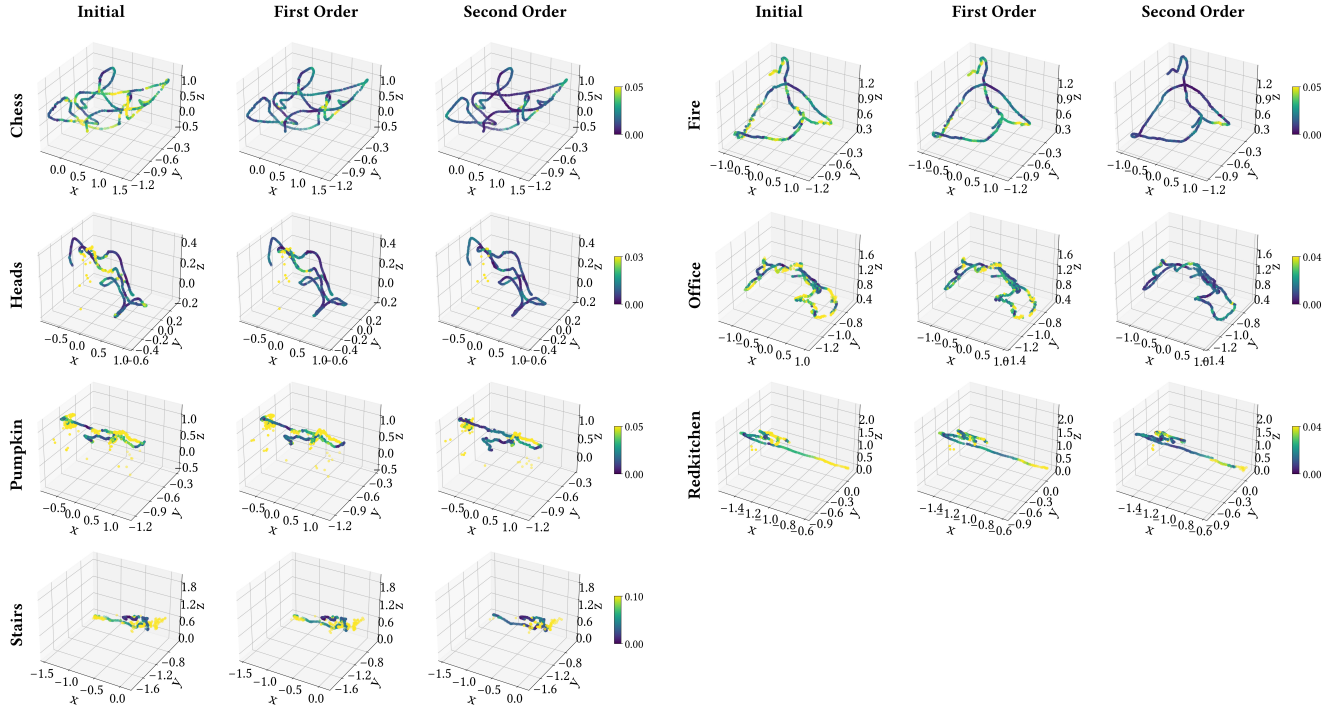
Fig. 5. **Camera relocalization trajectory in *7-Scenes dataset* for HLoc.** For all scenes, our optimization method based on X-KF can significantly enhance the relocalization accuracy. The results also demonstrate the importance of higher-order optimization, which is difficult for AD method.
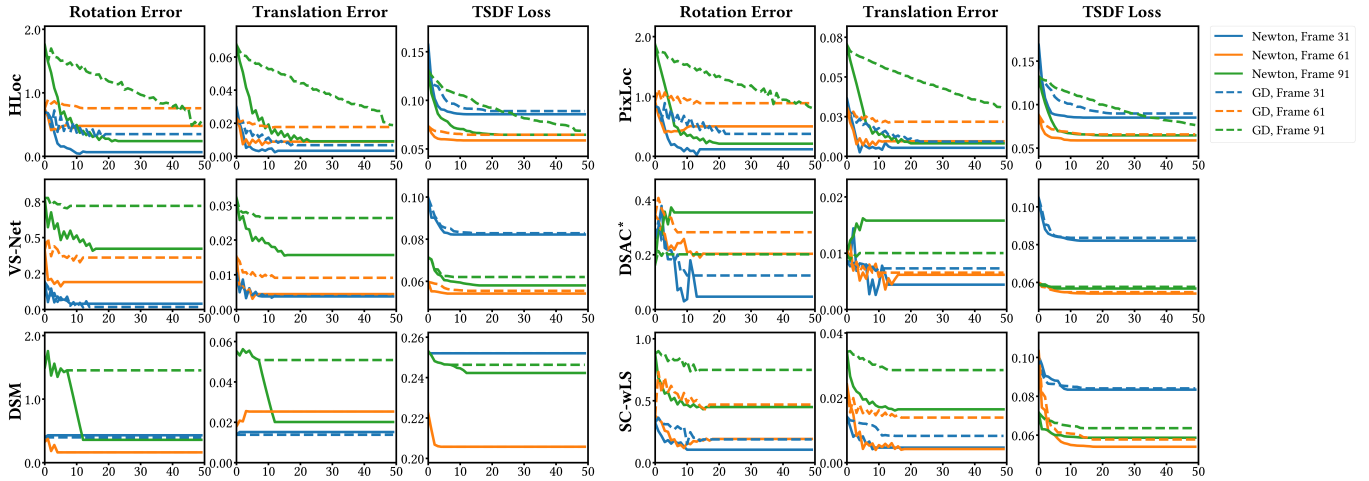
.

Fig. 6. **Optimization curves on the chess scene in *7-Scenes dataset*.** Three frames (No. 31, No. 61, No. 91) are optimized 50 iterations in Gradient Descent and Newton's method. Not only does Newton's method achieve a faster convergence speed, but it also prevents falling into local minima.

6 CONCLUSION

We present a real-time and differentiable dense SLAM system based on CSFD. By adding perturbations to variables of interest, the numerical differentiations can be computed in efficiently during the SLAM process. We propose X-KF and X-EF, which are the real-time differentiable versions of the two classic SLAM systems, and demonstrate

that CSFD is faster and uses less memory compared to auto differentiation. Based on X-SLAM, we introduce task-aware optimization frameworks for two downstream tasks, camera relocalization and robot active scanning. Thanks to the ability of X-SLAM to compute high-order derivatives, our optimization framework shows superior performance in a variety of public datasets and difficult real-world
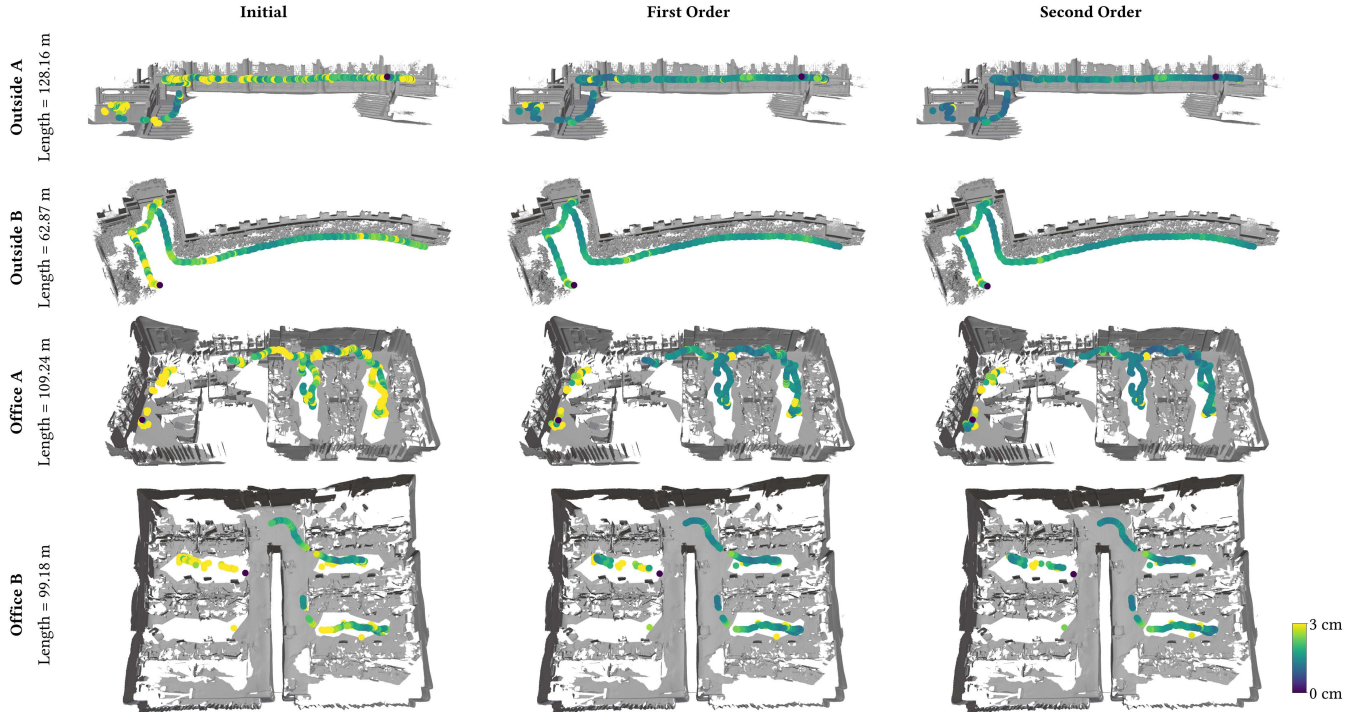
Fig. 7. Camera relocalization trajectories in our large-scale dataset for PixLoc. The trajectories are colored by relocalization error. Please note that we have discarded outliers with $e_q > 0.1m$, which occur more frequently in indoor scenes because of local repetition.
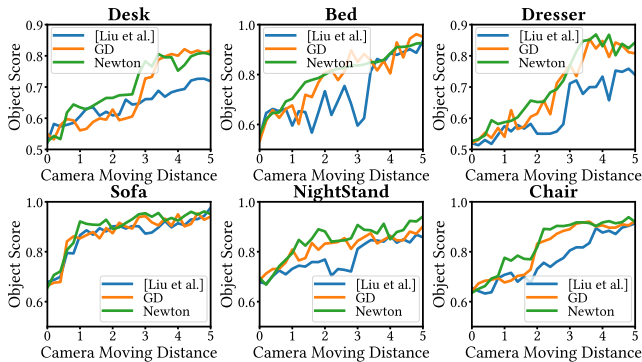

Fig. 8. Recognition score during robot activate scanning. We record the object recognition score given by PointNet every $0.2m$. On most objects, our method can achieve better results than the database-based method.

environments. As CSFD requires perturbing variables one by one to calculate derivatives, our method mainly focuses on pose optimization over a smaller number of degrees of freedom. It remains challenging to carry out high-order differential calculations and high-dimension degrees of freedom in the SLAM system.

ACKNOWLEDGMENTS

REFERENCES

Andres M. Aguirre-Mesa, Manuel J. Garcia, and Harry Millwater. 2020. MultiZ: A Library for Computation of High-order Derivatives Using Multicomplex or Multidual Numbers. *ACM Trans. Math. Softw.* 46, 3, Article 23 (jul 2020), 30 pages. https://doi.org/10.1145/3378538

Andreas Bircher, Mina Kamel, Kostas Alexis, Helen Oleynikova, and Roland Siegwart. 2016. Receding Horizon "Next-Best-View" Planner for 3D Exploration. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 1462–1468. https://doi.org/10.1109/ICRA.2016.7487281

Eric Brachmann, Alexander Krull, Sebastian Nowozin, Jamie Shotton, Frank Michel, Stefan Gumhold, and Carsten Rother. 2017. Dsac-differentiable ransac for camera localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 6684–6692.

Eric Brachmann and Carsten Rother. 2021. Visual camera re-localization from RGB and RGB-D images using DSAC. *IEEE transactions on pattern analysis and machine intelligence* 44, 9 (2021), 5847–5865.

Marco Callieri, Andrea Fasano, Gaetano Impoco, Paolo Cignoni, Roberto Scopigno, G Parrini, and Giuseppe Biagini. 2004. RoboScan: an automatic system for accurate and unattended 3D scanning. In *Proceedings. 2nd International Symposium on 3D Data Processing, Visualization and Transmission, 2004. 3DPVT 2004*. IEEE, 805–812.

Avraham Cohen and Moshe Shoham. 2016. Application of hyper-dual numbers to multibody kinematics. *Journal of Mechanisms and Robotics* 8, 1 (2016), 011015.

Angela Dai, Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Christian Theobalt. 2017. BundleFusion: Real-Time Globally Consistent 3D Reconstruction Using On-the-Fly Surface Reintegration. *ACM Trans. Graph.* 36, 4, Article 76a (jul 2017), 18 pages. https://doi.org/10.1145/3072959.3054739

Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. 2018. Superpoint: Self-supervised interest point detection and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 224–236.

Jeffrey Fike and Juan Alonso. 2011. The development of hyper-dual numbers for exact second-derivative calculations. In *49th AIAA aerospace sciences meeting including*

Trajectory Length = 36.34m   Trajectory Length = 33.60m   Trajectory Length = 26.97m

Trajectory Length = 31.30m   Trajectory Length = 31.17m   Trajectory Length = 28.87m

Trajectory Length = 65.74m   Trajectory Length = 55.84m   Trajectory Length = 52.03m

Trajectory Length = 50.39 m   Trajectory Length = 40.50 m   Trajectory Length = 32.35 m

[Liu et al.]   Gradient Descent   Newton's Method

Fig. 9. **Robot active scanning in virtual scenes.** The robot starts from the same position for active scanning, and the moving trajectory is drawn on the map. We also report the trajectory length.

the new horizons forum and aerospace exposition. 886.

Huan Fu, Rongfei Jia, Lin Gao, Mingming Gong, Binqiang Zhao, Steve Maybank, and Dacheng Tao. 2021. 3d-future: 3d furniture shape with texture. *International Journal of Computer Vision* (2021), 1–25.

Héctor H. González-Baños and Jean-Claude Latombe. 2002. Navigation Strategies for Exploring Indoor Environments. *The International Journal of Robotics Research* 21,
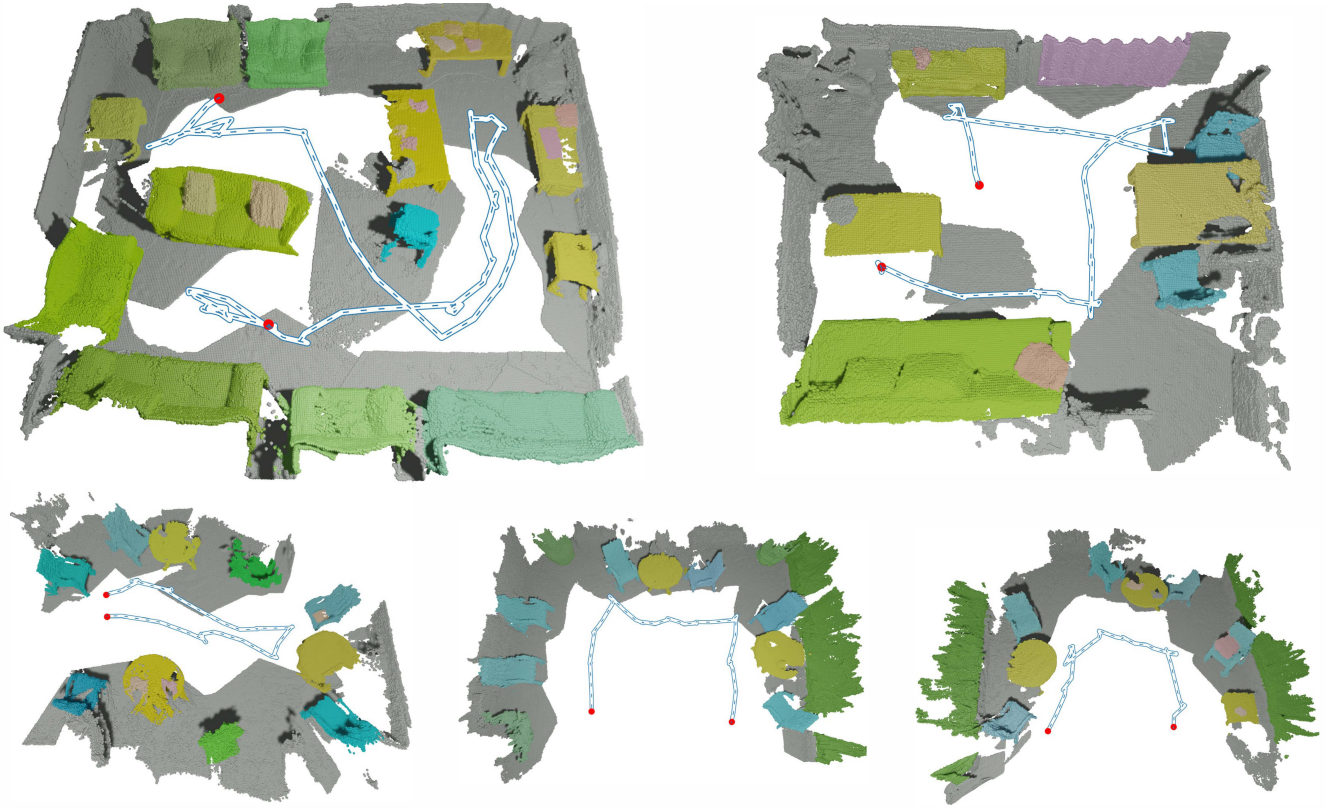
Fig. 10. **Segmentation and reconstruction results in real scene scanning.**

10-11 (2002), 829–848. https://doi.org/10.1177/0278364902021010834

Sai Krishna Gottipati, Keehong Seo, Dhaivat Bhatt, Vincent Mai, Krishna Murthy, and Liam Paull. 2019. Deep active localization. *IEEE Robotics and Automation Letters* 4, 4 (2019), 4394–4401.

Can Gümeli, Angela Dai, and Matthias Nießner. 2023. ObjectMatch: Robust Registration using Canonical Object Correspondences. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 13082–13091. https://doi.org/10.1109/CVPR52729.2023.01257

Ankur Handa, Thomas Whelan, John McDonald, and Andrew J. Davison. 2014. A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*. 1524–1531. https://doi.org/10.1109/ICRA.2014.6907054

Nicholas J Higham. 2002. *Accuracy and stability of numerical algorithms.* SIAM.

Yuanming Hu, Jiancheng Liu, Andrew Spielberg, Joshua B Tenenbaum, William T Freeman, Jiajun Wu, Daniela Rus, and Wojciech Matusik. 2019. Chainqueen: A real-time differentiable physical simulator for soft robotics. In *2019 International conference on robotics and automation (ICRA)*. IEEE, 6265–6271.

Shi-Sheng Huang, Ze-Yu Ma, Tai-Jiang Mu, Hongbo Fu, and Shi-Min Hu. 2021a. Supervoxel Convolution for Online 3D Semantic Segmentation. *ACM Transactions on Graphics* 40, 3 (Aug. 2021), 34:1–34:15. https://doi.org/10.1145/3453485

Zhaoyang Huang, Han Zhou, Yijin Li, Bangbang Yang, Yan Xu, Xiaowei Zhou, Hujun Bao, Guofeng Zhang, and Hongsheng Li. 2021b. Vs-net: Voting with segmentation for visual localization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 6101–6111.

Krishna Murthy Jatavallabhula, Ganesh Iyer, and Liam Paull. 2020. ∇SLAM: Dense SLAM meets Automatic Differentiation. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2130–2137. https://doi.org/10.1109/ICRA40945.2020.9197519

Peter Karkus, Shaojun Cai, and David Hsu. 2021. Differentiable SLAM-net: Learning Particle SLAM for Visual Navigation. arXiv:2105.07593 [cs, stat]

Alex Kendall, Matthew Grimes, and Roberto Cipolla. 2015. Posenet: A convolutional network for real-time 6-dof camera relocalization. , 2938–2946 pages.

Gregory Lantoine, Ryan P Russell, and Thierry Dargent. 2012. Using multicomplex variables for automatic computation of high-order derivatives. *ACM Transactions*

on Mathematical Software (TOMS) 38, 3 (2012), 16.

Xinyi Li and Haibin Ling. 2022. GTCaR: Graph Transformer for Camera Re-Localization. In *Computer Vision – ECCV 2022*, Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner (Eds.). Vol. 13670. Springer Nature Switzerland, Cham, 229–246. https://doi.org/10.1007/978-3-031-20080-9_14

Ligang Liu, Xi Xia, Han Sun, Qi Shen, Juzhan Xu, Bin Chen, Hui Huang, and Kai Xu. 2018. Object-aware guidance for autonomous scene reconstruction. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–12.

Leyao Liu, Tian Zheng, Yun–Jou Lin, Kai Ni, and Lu Fang. 2022. INS-Conv: Incremental Sparse Convolution for Online 3D Segmentation. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 18953–18962. https://doi.org/10.1109/CVPR52688.2022.01840

Ran Luo, Weiwei Xu, Tianjia Shao, Hongyi Xu, and Yin Yang. 2019. Accelerated complex-step finite difference for expedient deformable simulation. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 1–16.

Wei-Chiu Ma, Anqi Joyce Yang, Shenlong Wang, Raquel Urtasun, and Antonio Torralba. 2022. Virtual Correspondence: Humans as a Cue for Extreme-View Geometry. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 15903–15913. https://doi.org/10.1109/CVPR52688.2022.01546

Joaquim RRA Martins, Peter Sturdza, and Juan J Alonso. 2003. The complex-step derivative approximation. *ACM Transactions on Mathematical Software (TOMS)* 29, 3 (2003), 245–262.

John McCormac, Ankur Handa, Andrew Davison, and Stefan Leutenegger. 2017. SemanticFusion: Dense 3D Semantic Mapping with Convolutional Neural Networks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*. 4628–4635. https://doi.org/10.1109/ICRA.2017.7989538

Raul Mur-Artal, J. M. M. Montiel, and Juan D. Tardos. 2015. ORB-SLAM: A Versatile and Accurate Monocular SLAM System. *IEEE Transactions on Robotics* 31, 5 (Oct. 2015), 1147–1163. https://doi.org/10.1109/TRO.2015.2463671 arXiv:1502.00956 [cs]

Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. 2011. Kinectfusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE international symposium on mixed and augmented reality*. Ieee, 127–136.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.). Curran Associates, Inc., 8024–8035. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. 2017. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 652–660.

Renato F. Salas-Moreno, Richard A. Newcombe, Hauke Strasdat, Paul H.J. Kelly, and Andrew J. Davison. 2013. SLAM++: Simultaneous Localisation and Mapping at the Level of Objects. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*. 1352–1359. https://doi.org/10.1109/CVPR.2013.178

Paul-Edouard Sarlin, Cesar Cadena, Roland Siegwart, and Marcin Dymczyk. 2019. From Coarse to Fine: Robust Hierarchical Localization at Large Scale. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 12708–12717. https://doi.org/10.1109/CVPR.2019.01300

Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. 2020. Superglue: Learning feature matching with graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 4938–4947.

Paul-Edouard Sarlin, Ajaykumar Unagar, Mans Larsson, Hugo Germain, Carl Toft, Viktor Larsson, Marc Pollefeys, Vincent Lepetit, Lars Hammarstrand, Fredrik Kahl, et al. 2021. Back to the feature: Learning robust camera localization from pixels to pose. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 3247–3257.

Torsten Sattler, Bastian Leibe, and Leif Kobbelt. 2011. Fast Image-Based Localization Using Direct 2D-to-3D Matching. In *2011 International Conference on Computer Vision*. 667–674. https://doi.org/10.1109/ICCV.2011.6126302

Siyuan Shen, Yin Yang, Tianjia Shao, He Wang, Chenfanfu Jiang, Lei Lan, and Kun Zhou. 2021. High-order differentiable autoencoder for nonlinear model reduction. *ACM Trans. Graph.* 40, 4 (2021), 68:1–68:15. https://doi.org/10.1145/3450626.3459754

Jamie Shotton, Ben Glocker, Christopher Zach, Shahram Izadi, Antonio Criminisi, and Andrew Fitzgibbon. 2013. Scene Coordinate Regression Forests for Camera Relocalization in RGB-D Images. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*. 2930–2937. https://doi.org/10.1109/CVPR.2013.377

Erik Stenborg, Carl Toft, and Lars Hammarstrand. 2018. Long-Term Visual Localization Using Semantically Segmented Images. https://doi.org/10.48550/arXiv.1801.05269 arXiv:1801.05269 [cs]

Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. 2012. A benchmark for the evaluation of RGB-D SLAM systems. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 573–580. https://doi.org/10.1109/IROS.2012.6385773

Shitao Tang, Chengzhou Tang, Rui Huang, Siyu Zhu, and Ping Tan. 2021. Learning camera localization via dense scene matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1831–1841.

Carlo Tomasi and Roberto Manduchi. 1998. Bilateral filtering for gray and color images. In *Sixth international conference on computer vision (IEEE Cat. No. 98CH36271)*. IEEE, 839–846.

Bing Wang, Changhao Chen, Chris Xiaoxuan Lu, Peijun Zhao, Niki Trigoni, and Andrew Markham. 2020. Atloc: Attention guided camera localization. , 10393–10401 pages.

Thomas Whelan, Stefan Leutenegger, Renato F Salas-Moreno, Ben Glocker, and Andrew J Davison. 2015. ElasticFusion: Dense SLAM without a pose graph.. In *Robotics: science and systems*, Vol. 11. Rome, Italy, 3.

Yanmin Wu, Yunzhou Zhang, Delong Zhu, Xin Chen, Sonya Coleman, Wenkai Sun, Xinggang Hu, and Zhiqiang Deng. 2021. Object slam-based active mapping and robotic grasping. In *2021 International Conference on 3D Vision (3DV)*. IEEE, 1372–1381.

Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 2015. 3D ShapeNets: A deep representation for volumetric shapes. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 1912–1920. https://doi.org/10.1109/CVPR.2015.7298801

Zhe Xin, Yinghao Cai, Tao Lu, Xiaoxia Xing, Shaojun Cai, Jixiang Zhang, Yiping Yang, and Yanqing Wang. 2019. Localizing discriminative visual landmarks for place recognition. In *2019 International conference on robotics and automation (ICRA)*. IEEE, 5979–5985.

Kai Xu, Lintao Zheng, Zihao Yan, Guohang Yan, Eugene Zhang, Matthias Niessner, Oliver Deussen, Daniel Cohen-Or, and Hui Huang. 2017. Autonomous reconstruction of unknown indoor scenes guided by time-varying tensor fields. *ACM Trans. Graph.* 36, 6, Article 202 (nov 2017), 15 pages. https://doi.org/10.1145/3130800.3130812

Yabin Xu, Liangliang Nan, Laishui Zhou, Jun Wang, and Charlie C. L. Wang. 2022. HRBF-Fusion: Accurate 3D Reconstruction from RGB-D Data Using On-the-fly Implicits. *ACM Transactions on Graphics* 41, 3 (June 2022), 1–19. https://doi.org/10.1145/3516521

Rui Zeng, Yuhui Wen, Wang Zhao, and Yong-Jin Liu. 2020. View planning in robot active vision: A survey of systems, algorithms, and applications. *Computational Visual Media* 6 (2020), 225–245.

Liang Zhang, Leqi Wei, Peiyi Shen, Wei Wei, Guangming Zhu, and Juan Song. 2018. Semantic SLAM Based on Object Detection and Improved Octomap. *IEEE Access* 6 (2018), 75545–75559. https://doi.org/10.1109/ACCESS.2018.2873617

Lintao Zheng, Chenyang Zhu, Jiazhao Zhang, Hang Zhao, Hui Huang, Matthias Niessner, and Kai Xu. 2019. Active Scene Understanding via Online Semantic Reconstruction. *Computer Graphics Forum* 38, 7 (2019), 103–114. https://doi.org/10.1111/cgf.13820 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.13820