2020 IEEE International Conference on Robotics and Automation (ICRA)
31 May - 31 August, 2020. Paris, France

∇SLAM: Dense SLAM meets Automatic Differentiation

Krishna Murthy Jatavallabhula[1], Ganesh Iyer[3], and Liam Paull [1,2]

[1]Université de Montréal, Mila, Robotics and Embodied AI Lab (REAL), [2]Candian CIFAR AI Chair, [3]Carnegie Mellon University

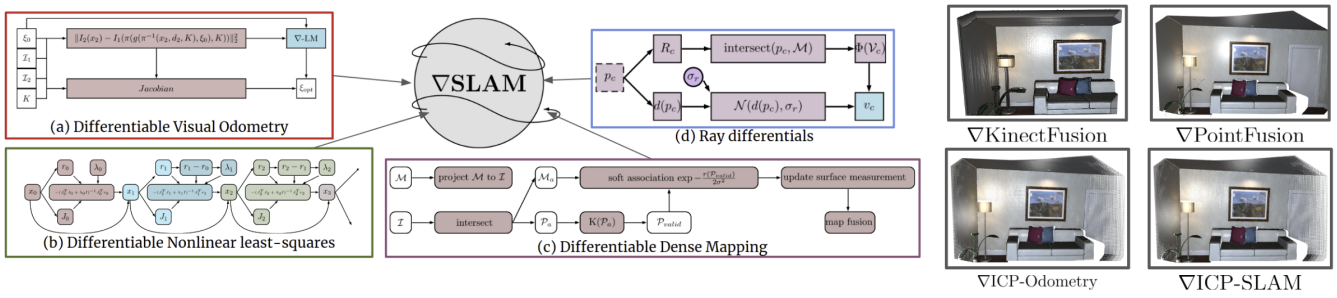

Fig. 1. **∇SLAM (gradSLAM)** is a *fully differentiable* dense simultaneous localization and mapping (SLAM) system. The central idea of ∇SLAM is to construct a computational graph representing every operation in a dense SLAM system. We propose differentiable alternatives to several non-differentiable components of traditional dense SLAM systems, such as optimization, odometry estimation, raycasting, and map fusion. This creates a pathway for gradient-flow from 3D map elements to sensor observations (e.g., *pixels*). We implement differentiable variants of three dense SLAM systems that operate on voxels, surfels, and pointclouds respectively. ∇SLAM thus is a novel paradigm to integrate representation learning approaches with classical SLAM.

***Abstract*— The question of "representation" is central in the context of dense simultaneous localization and mapping (SLAM). Learning-based approaches have the potential to leverage data or task performance to directly inform the representation. However, blending representation learning approaches with "classical" SLAM systems has remained an open question, because of their highly modular and complex nature. A SLAM system *transforms* raw sensor inputs into a distribution over the state(s) of the robot and the environment. If this *transformation* (SLAM) were expressible as a differentiable function, we could leverage task-based error signals over the outputs of this function to learn representations that optimize task performance. However, this is infeasible as several components of a typical dense SLAM system are non-differentiable. In this work, we propose ∇SLAM (*grad*SLAM), a methodology for posing SLAM systems as *differentiable* computational graphs, which unifies gradient-based learning and SLAM. We propose differentiable trust-region optimizers, surface measurement and fusion schemes, and raycasting, without sacrificing accuracy. This amalgamation of dense SLAM with computational graphs enables us to backprop all the way from 3D maps to 2D pixels, opening up new possibilities in gradient-based learning for SLAM[1].**

I. Introduction

Gradient-based learning architectures (deep neural networks) have tremendously improved robot perception [1] and action [2]. For decades, simulataneous localization and mapping (SLAM) has been a central element of robot perception and state estimation. SLAM allows robots to operate in previously unseen environments, a core capability that robots must possess for real-world deployment. A large portion of the visual SLAM literature has focused either directly or indirectly on the question of *map representation*. For example, feature-based SLAM systems build a map of unique features (landmarks), including interest points [3], lines [4], planes [5], objects [6], etc. In dense SLAM systems [7], [8], each pixel in each image contributes to the dense 3D map. Hybrid approaches try to exploit the best elements of both approaches [9]. This fundamental choice of representation dramatically impacts the design of processing blocks in the SLAM pipeline, as well as all other downstream tasks that depend on the output of the SLAM system. In particular, for *dense 3D maps* generated from RGB-D cameras, there has been a lack of consensus on the *right* map representation.

Learning representations, as has been done in several other domains [1], [10], [11] is thus appealing for SLAM. However, it is not straightforward because SLAM systems are composed of several subsystems (tracking, mapping, global optimization, etc.) many of which are not inherently differentiable. It is unclear whether *neural* SLAM systems of the future must subsume all these subsystems into a monolithic architecture [12], or retain the inductive biases from traditional SLAM systems and *employ learning only where necessary*. In this paper, we argue for the latter and propose ∇SLAM (gradSLAM): a fully differentiable dense SLAM system that harnesses the power of computational graphs and automatic differentiation to enable learning of dense geometric representations. To achieve this differentiability we reformulate each operation involved in SLAM as a computational graph.

This also allows us to solve the *inverse mapping* problem (i.e., answer the question: "*How much does a specific pixel-measurement contribute to the resulting 3D map*"?) something that is not possible with other popular dense visual SLAM systems [7], [13], [14]. Through the composition of the computational graphs, we obtain a function ($\mathcal{S}$) that

Correspondence to krrish94 [at] gmail [dot] com.

[1]A short video explaining the paper and showcasing the results can be found at https://youtu.be/2ygtSJTmo08

relates a pixel in an RGB-D image (or in general, any sensor measurement s) to a 3D geometric map $\mathcal{M}$ of the environment: $\mathcal{M} = \mathcal{S}(s)$. As a result, the *gradient* $\nabla_s \mathcal{S}$ tells us that perturbing the sensor measurement s by an infinitesimal δs causes the map $\mathcal{M}$ to change by $\nabla_s \mathcal{S}(s)\delta s$.

Central to our goal of realizing a fully differentiable SLAM system are *computational graphs*, which underlie most gradient-based learning techniques. If an entire SLAM system can be composed from elementary operations, all of which are differentiable, the system allows end-to-end gradient propagation by construction. However, modern *dense* SLAM systems are quite sophisticated, with several non-differentiable subsystems (optimizers, raycasting, surface mapping), that make such a construction challenging. We show how *all* non-differentiable functions in SLAM can be realised as smooth mappings. In particular, we present differentiable versions of nonlinear least squares optimization, raycasting, and rasterization, while maintaining commensurate accuracy with the non-differentiable counterparts.

We demonstrate ∇SLAM through 3 instantiations, where our differentiable SLAM building blocks are used to realize the following dense SLAM systems: implicit-surface mapping (Kinectfusion [7]), surfel-based mapping (PointFusion [14]), and iterative closest point (ICP) mapping (ICP-SLAM). We also demonstrate examples of backpropagating error signals through the entire SLAM system that enable exciting avenues in representation learning for SLAM[2].

II. Related Work

Several recent approaches have applied representation learning to SLAM or have reformulated a subset of *components* of the full SLAM system in a differentiable manner.

A. Learning-based SLAM Approaches

There is a large body of work in deep learning-based SLAM systems. For example, CodeSLAM [15], SceneCode [16], and DeepFactors [17] represent scenes using compact *codes* that that can be decoded into 2.5D depth maps. DeepTAM [18] trains a tracking network and a mapping network, which learn to reconstruct a voxel representation from a pair of images. CNN-SLAM [19] extends LSD-SLAM [20], a popular monocular SLAM system, to use single-image depth predictions from a convnet.

Another recent trend has been to formulate the SLAM problem over higher level features such as objects, which may be detected with learned detectors [6], [21], [22].

B. Differentiable SLAM Subsystems

While a large fraction of the above approaches *replace* SLAM subsystems with representation learning machinery (neural nets), there is another significant line of work that leverages differentiability to *complement* and accelerate learning mechanisms.

Recent work has formulated passive [23] and active localization [24], [25], active SLAM [12], camera resectioning [26], and cross-sensor calibration [27] in a fully differentiable manner. Another strong demonstration of the benefits of differentiable SLAM subsystems is the Lucas-Kanade iterative matching algorithm [28]. Kerl *et al.* [29] apply this technique to real-time dense visual odometry. This differentiable subsystem has been extensively used for self-supervised depth and motion estimation [30]–[32]. These *two-view* techniques have been extensively used as *layers* in neural networks [33], [34]. However, extending differentiability beyond the two-view case (*frame-frame alignment*) is not straightforward. Global consistency necessitates fusing measurements from live frames into a global model (*model-frame alignment*), which is non-differentiable.

C. Differentiable Optimization

Recent work has proposed to learn the optimization of nonlinear objective functions, motivated by the idea that learning methods can aid in faster convergence and improve solution quality.

DeBrandandere *et al.* [35] perform lane detection by back-propagating least-squares residuals into a frontend module. In BA-Net [36], the authors learn to predict the damping coefficient of the Levenberg-Marquardt optimizer, while in LS-Net [37], the authors entirely replace the Levenberg-Marquard optimizer by an LSTM netowrk [38] that predicts update steps. In GN-Net [39], a differentiable version of the Gauss-Newton loss is used to show better robustness to weather conditions. RegNet [40] employs a learned optimization of the photometric error for image-to-image pose registration. However, all the aforementioned approaches require the training of additional neural networks and this requirement imposes severe limitations on the generalizability. OptNet [41] introduces differentiable optimization layers without learnable parameters, specifically for quadratic programs.

Concurrently, Grefenstette *et al.* [42] unroll optimizers as computational graphs, allowing the computation of arbitrarily higher order gradients. Our proposed differentiable Levenberg-Marquardt optimizer is similar in spirit, with the addition of gating functions to result in better gradient flows.

To the best of our knowledge, there is no *single* approach that models the entire SLAM pipeline as a differentiable model, and this is the motivation that underlies ∇SLAM.

III. ∇SLAM

In this section we will overview our proposed method for ∇SLAM and also detail the individual differentiable sub-components.

A. Preliminaries: Computational Graphs

In gradient-based learning architectures, all functions and approximators are conventionally represented as *computational graphs*. Formally, a computational graph is a directed acyclic graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where each node $v \in \mathcal{V}$ holds an operand or an operator, and each (directed) edge $e \in \mathcal{E}$ indicates the control flow in the graph. Each node v also specifies a *backward* computation rule that computes the gradient of the outputs of the node with respect to its inputs.

[2]Project page: http://montrealrobotics.ca/gradSLAM/

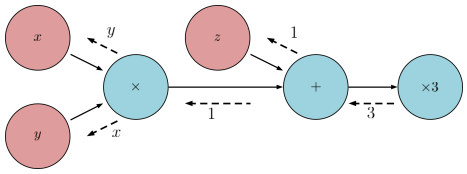

Fig. 2. **A computational graph**. Nodes in red represent variables. Nodes in blue represent operations on variables. Edges represent data flow. This graph computes the function $3(xy + z)$. Dashed lines indicate (local, i.e., per-node) gradients in the backward pass.

Computational graphs can be nested and composed in about any manner, whilst preserving differentiability. Eg. Fig. 2 shows a computational graph for the function $3(xy + z)$.

B. Overview of ∇SLAM

The objective of ∇SLAM is to make every computation in SLAM exactly realised as a composition of differentiable functions. Wherever exact differentiable realizations are not possible, we desire *as-exact-as-possible* differentiable realizations. Broadly, the sequence of operations in dense SLAM systems can be termed as *visual odometry* (V0), (frame-to-frame alignment), *map building* (model-to-frame alignment/local optimization), and *global optimization*. An overview of the approach is shown in Fig. 1. In the remainder of this section, we provide a description of the precise issues that render the non-linear optimization (Sec. III-C) dense mapping (Sec. III-D), and measurement fusion (Sec. III-E), modules non-differentiable, and propose differntiable counterparts for each module. In the following section (Sec. IV), we show that the proposed differentiable variants allow the realization of differentiable versions of several classic dense mapping algorithms (*KinectFusion* [7], *PointFusion* [14], ICP-SLAM) in the ∇SLAM framework.

C. ∇LM: A Differentiable Nonlinear Least Squares Solver

Most state-of-the-art SLAM solutions optimize (minimize) nonlinear least squares objectives to obtain local/globally consistent estimates of the robot state and the map. Such objectives are of the form $\frac{1}{2}\sum \mathbf{r}(\mathbf{x})^2$, where $\mathbf{r}(\mathbf{x})$ is a nonlinear function of residuals. Example application scenarios that induce this nonlinear least squares form include visual odometry, depth measurement registration (e.g., ICP), and pose-graph optimization among others. These objective functions are minimized using a succession of linear approximations ($\mathbf{r}(\mathbf{x} + \delta\mathbf{x})|_{\mathbf{x}=\mathbf{x_0}} = \mathbf{r}(\mathbf{x_0}) + \mathbf{J}(\mathbf{x_0})\delta\mathbf{x}$) inside Gauss-Newton (GN) or Levenberg-Marquardt (LM) solvers. GN solvers are extremely sensitive to intialization, numerical precision, and moreover, provide no guarantees on *non-divergent* behavior. Hence most SLAM systems use LM solvers.

Trust-region methods (such as LM) are not differentiable as at each optimization step, they involve recalibration of optimizer parameters, based on a *lookahead* operation over subsequent iterates [43]. Specifically, after a new iterate is computed, LM solvers need to make a *discrete* decision between damping or undamping the linear system. Furthermore, when undamping, the iterate must be restored to its previous value. This discrete *switching* behavior of LM does not allow for gradient flow in the backward pass.

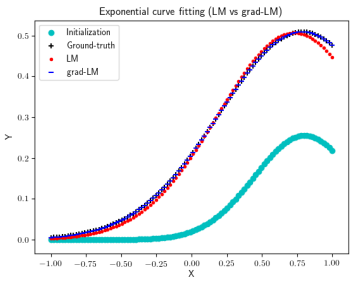

Fig. 3. An example curve fitting problem, showing that ∇LM performs nearly identically to LM, with the added advantage of being fully differentiable.

We propose a computationally efficient *soft reparametrization* of the damping mechanism to enable differentiability in LM solvers. Our key insight is that, if $\mathbf{r_0} = \mathbf{r}(\mathbf{x_0})^\mathbf{T}\mathbf{r}(\mathbf{x_0})$ is the (squared) norm of the error at the current iterate, and $\mathbf{r_1} = \mathbf{r}(\mathbf{x_1})^\mathbf{T}\mathbf{r}(\mathbf{x_1})$ is the norm of the error at the *lookahead* iterate, the value of $\mathbf{r_1} - \mathbf{r_0}$ determines whether to damp or to undamp. And, only when we choose to undamp, we revert to the current iterate. We define two smooth *gating* functions Q_x and Q_λ based on the generalized logistic function [44] to update the iterate and determine the next damping coefficient.

$$\begin{aligned} \lambda_1 = Q_\lambda(r_0, r_1) &= \lambda_{min} + \frac{\lambda_{max} - \lambda_{min}}{1 + De^{-\sigma(r_1 - r_0)}} \\ Q_x(r_0, r_1) &= x_0 + \frac{\delta x_0}{1 + e^{-(r_1 - r_0)}} \end{aligned} \tag{1}$$

where D and σ are tunable hyperparameters that control the slope of the falloff function [44]. $[\lambda_{min}, \lambda_{max}]$ is the range of values the damping function can assume (usually, $\lambda_{min} = \frac{1}{2}$, $\lambda_{max} = 2$, when using multiplicative damping with a damping coefficient of 2). This smooth parameterization of the LM update allows the optimizer to be expressed as a fully differentiable computational graph (Fig. 1(b)). Further, the performance of ∇LM is similar to that of LM, as can be seen in Fig. 3.

∇LM allows differentiable realizations of several SLAM components, such as dense visual odometry estimation [45] (Fig. 1(a)), and Iterative Closest Point (ICP) alignment [46].

D. Differentiable Mapping

Another non-smooth operation in dense SLAM is map construction (*surface measurement*). For example, consider a *global* map $\mathcal{M}$ being built in the reference frame of the first image-sensor measurement I_0. When a new frame I_k arrives at time k, the surface measurement from this frame needs to be aligned with the global map. This *surface alignment* process comprises the following steps.

1) The map $\mathcal{M}$ is intersection-tested with the live frame, to determine *active map elements* $\mathcal{M}_a$, and *active pixels* $\mathcal{P}_a$. Inactive map elements are *clipped*.
2) The active pixels $\mathcal{P}_a$ are checked for measurement validity (missing depths, blur, etc.). This results in a *valid active set* of image pixels $\mathcal{P}_{valid}$.

3) Pixels in $\mathcal{P}_{valid}$ are backprojected to 3D and compared with the map, to discern whether these pixels measure existing elements in $\mathcal{M}_a$, or if they measure new parts of the scene that must be added to the global map.
4) These surface measurements are then *fused* into the global map using a representation-specific mechanism (points, surfels, TSDFs, etc.).

The above process involves a number of differentiable yet non-smooth operations (clipping, indexing, thresholding, new/old decision, active/inactive decision, etc.). Although the above sequence of operations can be represented as a computation graph, only local derivatives can be defined for such operations. The overall function represented by the computation graph will have undefined gradients "almost everywhere". We mitigate this issue by making the functions locally *smooth*. Concretely, we propose the following corrective measures.

1) The surface measurement made at each valid pixel p in the live frame ($p \in \mathcal{P}_{valid}$) is a function of p and also active neighbours of p, $nbd(p)$ via a *kernel* $K(p, nbd(p))$.
2) When a surface measurement is transformed to the global frame, we use *soft* (one-many) rather than *hard* (one-one) associations.
3) Every surface measurement is, by default, assumed to represent a new map element, which is passed to a *differentiable fusion* step (*cf.* Sec III-E).

The kernel $K(p, nbd(p))$ can be a discrete approximation (e.g., constant within a pixel) or can vary at the subpixel level. For faster computation and coarse gradients, we use a bilinear interpolation kernel. While bilinear interpolation is a sensible approximation for image pixels, this is often a poor choice for use in 3D *soft* associations. For forming 3D associations, we leverage characteristics of RGB-D sensors in defining the soft falloff functions. Specifically, we compute, for each point $P \in \mathcal{M}_a$ in the live surface measurement, a set of closest candidate points in a region $exp\left(-\frac{r(P)^2}{2\sigma^2}\right)$, where $r(P)$ is the radial depth of the point from the camera ray, and σ affects the falloff region [14], [47], [48]. The computational graph for this differentiable mapping process is illustrated in Fig. 1(c).

E. Differentiable Measurement Fusion

The aforementioned differentiable mapping strategy results in a smooth observation model, but causes an undesirable effect: the number of map elements increases with *exploration time*. Rather, map elements should increase in proportion to the *explored volume of occupied space*. Conventional dense mapping techniques [7], [14] hence perform *fusion* of redundant *measurements* of each map element. Consequently, the recovered map has a more manageable size and better reconstruction quality. While most fusion strategies are differentiable (eg. [7], [14]), they impose falloff thresholds that cause an abrupt change in gradient flow at the truncation point. We use a logistic falloff function, similar to Eq. 1, to ease gradient flow through these truncation points.

F. Differentiable Ray Backprojection

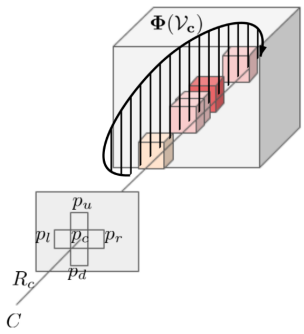

Fig. 4. **Ray differentials**: Inset shows the computation graph of the ray value computation. The dashed rectangle is not differentiable, and its derivatives are approximated as shown in Eq. (2).

Some dense SLAM systems [7], [13] perform global pose estimation by raycasting a map to a live frame. This process involves multiple non-differentiable steps. A ray is backprojected from the camera to every pixel in the image. Its intersection with the first map element is found by marching along the ray until a map element is intersected, or bounds of reconstruction are exitted. To make these operations differentiable, current approaches either involve a) parameterizing the marching process using learnable differentiable functions (such as LSTMs) [49], or b) by pooling over all map elements that intersect the ray [50], [51]. In this work, we make one enhancement to the latter class of ray pooling operations, by performing weighted pooling (averaging) over all voxels along a ray. The weights for each ray are defined by a Gaussian distribution $\mathcal{N}(d, \sigma_d^2)$ (d is the depth measured at the pixel location in the depth map and σ_d^2 is a hyperparameter that controls the spread of the Gaussian kernel). Further, we use finite differences to compute the derivative of the ray potential with respect to the pixel neighbourhood [52], illustrated in Fig. 4. If p_c is the image pixel and $\mathcal{V}_c = \{v_c\}$ is the set of all voxels a ray pierces, the *aggregated value* of the ray v_c is $\Phi(\psi(v_c))\ \forall v_c \in \mathcal{V}_c$. The aggregation function we consider is a weighted convex combination. If v_l, v_r, v_u, and v_b are the *aggregated values* of rays emanating from the pixels to the left, right, above, and below p_c respectively, the partial derivative $\frac{\partial v_c}{\partial p_c}$ can be approximated as

$$\frac{\partial v_c}{\partial p_c} = \begin{pmatrix} (v_r - v_l)/2 \\ (v_u - v_d)/2 \end{pmatrix} \tag{2}$$

The computational graph for the same is shown in Fig. 1(d).

IV. Case Studies: KinectFusion, PointFusion, and *ICP-SLAM*

As concrete demonstrations of ∇SLAM, we leverage the aforementioned differentiable SLAM subsystems and compose them to realise three practical SLAM solutions. We implement differentiable versions of the *KinectFusion* [7] algorithm that constructs TSDF-based volumetric maps, the

Fig. 5. **Qualitative results** on sequences from the ScanNet [53] dataset. Owing to GPU memory constraints, we use each of the differentiable SLAM systems (∇KinectFusion, ∇PointFusion, and ∇ICP-SLAM) to reconstruct parts of the scene. We also show outputs from BundleFusion [54].

Fig. 6. **Left**: Reconstruction obtained upon running ∇ICP-Odometry on a subsequence `rgbd_dataset_freiburg1_xyz` [55]. **Right**: In-house sequence collected from an Intel RealSense D435 camera. Reconstruction is obtained from ∇PointFusion.

PointFusion [14] algorithm that constructs surfel maps, and a pointcloud-based SLAM framework that we call *ICP-SLAM*.

A. KinectFusion

KinectFusion [7] alternates between *tracking* and *mapping* phases. In the tracking phase, the entire up-to-date TSDF volume is raycast onto the live frame, and odometry is estimated via point-to-plane ICP alignment. In the mapping phase, surface measurements from the live frame are *fused* into the volume, using weighted averaging of TSDF volumes [7], [48]. The other components of KinectFusion such as raycasting and ICP are rendered differentiable as explained in Sec III.

B. PointFusion

As a second example, we implement PointFusion [14], which incrementally fuses surface measurements to obtain a global surfel map. Surfel maps compare favourably to volumetric maps due to their reduced memory usage[3]. We closely follow our differentiable mapping formulation (*cf.* Sec III-D) and use surfels as map elements. We adopt the fusion rules from [14] to perform map fusion.

C. ICP-SLAM

As a baseline example, we implement a simple pointcloud based SLAM technique, which uses ICP to incrementally register pointclouds to a global map. We implement two variants: *ICP-Odometry*, which aligns every pair of consecutive incoming frames (frame-to-frame alignment), and *ICP-SLAM*, which aligns each incoming pointcloud to the global map (frame-to-model alignment).

V. Experiments and Results

A. Differentiable Optimization

We design a test suite of nonlinear curve fitting problems (similar to [37]), to measure the performance of ∇LM (Sec III-C) to its non-differentiable counterpart. We uniformly sample the parameters $p = a, b, c$, with initial guess a_0, b_0, c_0 from the exponential family: $p \sim y = aexp(-\frac{(x-b)^2}{2c^2})$. For 1000 sampled problem sets, we optimize using both LM and ∇LM, and measure the following quantities: iterations to converge, quality of the solution (i.e., discrepancy between estimated and true parameters). Notice from Table I and Fig. 3 how ∇LM performs similarly to LM (a slight performance drop is noticeable, due to smoothing).

[3]On the flipside, surfel-based algorithms are harder to parallelize compared to volumetric fusion.

Metric	LM	∇LM
Convergence iters	7.6 ± 3.4	$\mathbf{7.4} \pm 3.8$
$\|a_{opt} - a_{init}\|_1$	$\mathbf{0.011} \pm 0.011$	0.399 ± 0.162
$\|b_{opt} - b_{init}\|_1$	$\mathbf{0.331} \pm 0.174$	0.574 ± 0.137
$\|c_{opt} - c_{init}\|_1$	0.114 ± 0.021	$\mathbf{0.084} \pm 0.032$
Total error	**0.184**	0.207

TABLE I

∇LM performs quite similarly to its non-differentiable counterpart. (Convergence tolerance 10^{-6})

B. Comparitive Analysis of Case Studies

We present an analysis of how each of the differentiable SLAM systems compare to their non-differentiable counterparts. Table II shows the trajectory tracking performance of the non-differentiable and differentiable (∇) versions of ICP-Odometry, ICP-SLAM, PointFusion, and KinectFusion. We observe on-par performance when utilizing the differentiable mapping modules and ∇LM. This is computed over split subsets of the `living_room_traj0` sequence.

We also evaluate the reconstruction quality of ∇-KinectFusion with that of Kintinuous [13]. On a subsection of the `living_room_traj0` sequence of the ICL-NUIM [56] benchmark, the surface reconstruction quality of Kintinuous is 18.625, while that of differentiable KinectFusion is 21.301 (better). However, this quantity is misleading, as Kintinuous only retains a subset of high confidence points in the extracted mesh, while our differentiable KinectFusion

Method	ATE	RPE
ICP-Odometry (non-differentiable)	0.029	0.0318
∇ICP-Odometry	**0.01664**	**0.0237**
ICP-SLAM (non-differentiable)	0.0282	0.0294
∇ICP-SLAM	**0.01660**	**0.0204**
PointFusion (non-differentiable)	**0.0071**	**0.0099**
∇PointFusion	0.0072	0.0101
KinectFusion (non-differentiable)	**0.013**	**0.019**
∇KinectFusion	0.016	0.021

TABLE II

Performance of ∇SLAM compared to non-differentiable counterparts (ATE: Absolute Trajectory Error, RPE: Relative Pose Error).

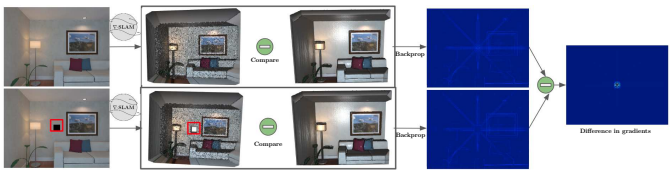

Fig. 7. **Analysis of gradients**: ∇SLAM enables gradients to flow through to the input images. *Top*: An RGB-D image pair (depth not shown) is passed through ∇KinectFusion. The resulting map is compared with a precise (ground-truth) map. The comparision error is backpropagated through the SLAM system, to the depth map (blue colormap). *Bottom*: An occluder is added to the center of the RGB-D pair. This occluder results in a gaping hole. But, using the backpropagated gradients, one can identify the set of image/depthmap pixels that result in the reconstruction error.

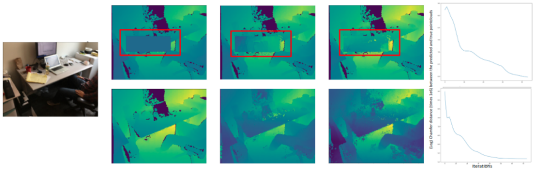

Fig. 8. **End-to-end gradient propagation**: (*Top*): A chunk of a depth map is *chopped*. The resultant sequence is reconstructed using ∇PointFusion and the pointcloud is compared to a *clean* one reconstructed using the unmodified depth map. The Chamfer distance between these two pointclouds is used to define a reconstruction error between the two clouds, which is backpropagated through to the input depth map and updated by gradient descent. (*Bottom*): Similar to the Fig. 7, we show that ∇SLAM can *fill-in* holes in the depthmap by leveraging multi-view gradient information.

outputs (see Fig. 7) contain a few noisy artifacts, due to our smooth truncation functions.

C. Qualitative Results

∇SLAM works out-of-the-box on multiple RGB-D datasets. Fig. 5 shows the result of running ∇SLAM on sequences from the Scannet [53] dataset. Fig. 6 shows results on a sequence from the TUM RGB-D benchmark [55] and an in-house sequence collected using an Intel Realsense D435 Camera. All the differentiable SLAM systems demonstrated execute fully on the GPU, and are capable of computing gradients with respect to *any* intermediate variable (Eg. camera poses, pixel intensities/depths, optimization parameters, camera intrinsics, etc.).

D. Analysis of Gradients

The computational graph approach of ∇SLAM allows us to recover meaningful gradients of 2D (or 2.5D) measurements with respect to a 3D surface reconstruction. We provide an analysis of what these multi-view gradients correlate to in the input image and depth space. In Fig. 7, the top row shows an RGB-D image differentiably transformed—using ∇SLAM—into a (noisy) TSDF surface measurement, and then compared to a more precise global TSDF map. The bottom row is similarly transformed, with the difference being the presence of a small (40×40 px) occluder. Elementwise comparision of aligned volumes gives us a reconstruction error, whose gradients are backpropagated through to the input depthmap using the computational graph maintained by ∇SLAM. Inspecting the gradients with respect to the input indicates the per pixel contribution of the occluding surface to the volumetric error. In Fig. 8, we similarly introduce such occluders (top row) and pixel noise (bottom row) in one of the depth maps of a sequence and reconstruct the scene using ∇PointFusion. We then calculate the chamfer distance between the noisy and true surfel maps and backpropogate the error with respect to each pixel. The minimized loss leads to the targeted recovery of the noisy and occluded regions. We additionally show an RGB-D image completion task (from uniform noise)in Fig. 9.

Thus, ∇SLAM provides a rich interpretation of the computed gradients: they denote the contribution of each pixel towards the eventual 3D reconstruction.

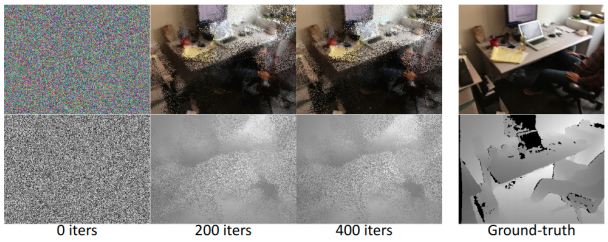

Fig. 9. **RGB-D completion using end-to-end gradient propagation**: Three RGB-D images and a *noise image* are passed through ∇PointFusion, and compared to a clean reconstruction obtained from four RGB-D images. The reconstruction loss is used to optimize the *noise image* by gradient descent. We can recover most of the artifacts from the raw RGB and depth images. Note that finer features are hard to recover from a random initialization, as the overall *SLAM function* is only locally differentiable.

VI. Conclusion

We introduce ∇SLAM, a differentiable computational graph framework that enables gradient-based learning for a large set of localization and mapping based tasks by providing explicit gradients with respect to the input image and depth maps. We demonstrate a diverse set of case studies and showcase how the gradients propogate through the tracking, mapping, and fusion stages. Future efforts will enable ∇SLAM to be directly optimized in conjunction with downstream tasks. ∇SLAM can also enable a variety of self-supervised learning applications, as any gradient-based learning architecture can now be equipped with a sense of *spatial understanding*.

Acknowledgements

Krishna Murthy Jatavallabhula and Liam Paull were partly supported by the NSERC discovery grant program, FRQNT Établissement de nouveaux chercheurs et de nouvelles chercheuses universitaires, CIFAR Canada AI Chair Program, and Samsung Advanced Institute of Technology. The authors also thank Soroush Saryazdi for help with the end-to-end experiments, and Gunshi Gupta, Aaditya Saraiya, Parv Parkhiya, Akshit Gandhi, and Shubham Garg, for their timely assistance.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NEURIPS*, 2012. 1

[2] A. Kendall, J. Hawke, D. Janz, P. Mazur, D. Reda, J.-M. Allen, V.-D. Lam, A. Bewley, and A. Shah, "Learning to drive in a day," in *IEEE ICRA*, 2019. 1

[3] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras," *IEEE TRO*, vol. 33, no. 5, pp. 1255–1262, 2017. 1

[4] P. Smith, I. D. Reid, and A. J. Davison, "Real-time monocular slam with straight lines," 2006. 1

[5] M. Kaess, "Simultaneous localization and mapping with infinite planes," in *IEEE ICRA*, 2015. 1

[6] B. Mu, S. Liu, L. Paull, J. Leonard, and J. P. How, "Slam with objects using a nonparametric pose graph," in *IEEE IROS*, 2016. 1, 2

[7] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. W. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking." in *ISMAR*, 2011. 1, 2, 3, 4, 5

[8] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "Dtam: Dense tracking and mapping in real-time," in *ICCV*, 2011. 1

[9] C. Forster, M. Pizzoli, and D. Scaramuzza, "Svo: Fast semi-direct monocular visual odometry," in *IEEE ICRA*, 2014. 1

[10] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *NEURIPS*, 2017. 1

[11] G. Hinton, L. Deng, D. Yu, G. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, B. Kingsbury, *et al.*, "Deep neural networks for acoustic modeling in speech recognition," *IEEE Signal processing magazine*, 2012. 1

[12] J. Zhang, L. Tai, J. Boedecker, W. Burgard, and M. Liu, "Neural SLAM," *arXiv*, vol. 1706.09520, 2017. 1, 2

[13] T. Whelan, M. Kaess, H. Johannsson, M. Fallon, J. J. Leonard, and J. McDonald, "Real-time large-scale dense rgb-d slam with volumetric fusion," *IJRR*, vol. 34, no. 4-5, pp. 598–626, 2015. 1, 4, 5

[14] M. Keller, D. Lefloch, M. Lambers, S. Izadi, T. Weyrich, and A. Kolb, "Real-time 3d reconstruction in dynamic scenes using point-based fusion," in *International Conference on 3D Vision*, 2013. 1, 2, 3, 4, 5

[15] M. Bloesch, J. Czarnowski, R. Clark, S. Leutenegger, and A. J. Davison, "Codeslam—learning a compact, optimisable representation for dense visual slam," in *CVPR*, 2018. 2

[16] S. Zhi, M. Bloesch, S. Leutenegger, and A. J. Davison, "Scenecode: Monocular dense semantic reconstruction using learned encoded scene representations," in *CVPR*, 2019. 2

[17] J. Czarnowski, T. Laidlow, R. Clark, and A. Davison, "Deepfactors: Real-time probabilistic dense monocular slam," in *IEEE RAL*, 2020. 2

[18] H. Zhou, B. Ummenhofer, and T. Brox, "Deeptam: Deep tracking and mapping," in *ECCV*, 2018. 2

[19] K. Tateno, F. Tombari, I. Laina, and N. Navab, "Cnn-slam: Real-time dense monocular slam with learned depth prediction," in *CVPR*, 2017. 2

[20] J. Engel, T. Schöps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam," in *ECCV*, 2014. 2

[21] P. Parkhiya, R. Khawad, J. K. Murthy, B. Bhowmick, and K. M. Krishna, "Constructing category-specific models for monocular object-slam," in *IEEE ICRA*, 2018. 2

[22] S. Yang and S. Scherer, "Cubeslam: Monocular 3-d object slam," *IEEE TRO*, vol. 35, no. 4, pp. 925–938, 2019. 2

[23] E. Brachmann, A. Krull, S. Nowozin, J. Shotton, F. Michel, S. Gumhold, and C. Rother, "Dsac-differentiable ransac for camera localization," in *CVPR*, 2017. 2

[24] D. S. Chaplot, E. Parisotto, and R. Salakhutdinov, "Active neural localization," in *ICLR*, 2018. 2

[25] S. K. Gottipati, K. Seo, D. Bhatt, V. Mai, K. Murthy, and L. Paull, "Deep active localization," *IEEE RAL*, vol. 4, no. 4, pp. 4394–4401, Oct 2019. 2

[26] B. Chen, T.-J. Chin, and N. Li, "Bpnp: Further empowering end-to-end learning with back-propagatable geometric optimization," *arXiv:1909.06043*, 2019. 2

[27] G. Iyer, K. Ram, J. Krishna Murthy, and K. Madhava Krishna, "Calib-net: Self-supervised extrinsic calibration using 3d spatial transformer networks," 2018. 2

[28] B. D. Lucas, T. Kanade, *et al.*, "An iterative image registration technique with an application to stereo vision," 1981. 2

[29] C. Kerl, J. Sturm, and D. Cremers, "Robust odometry estimation for rgb-d cameras," in *IEEE ICRA*, 2013. 2

[30] R. Garg, V. K. BG, G. Carneiro, and I. Reid, "Unsupervised cnn for single view depth estimation: Geometry to the rescue," in *ECCV*, 2016. 2

[31] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video," in *CVPR*, 2017. 2

[32] R. Li, S. Wang, Z. Long, and D. Gu, "Undeepvo: Monocular visual odometry through unsupervised deep learning," in *IEEE ICRA*, 2018. 2

[33] A. Handa, M. Bloesch, V. Pătrăucean, S. Stent, J. McCormac, and A. Davison, "gvnn: Neural network library for geometric computer vision," in *ECCV Workshop on Geometry Meets Deep Learning*, 2016. 2

[34] E. Riba, D. Mishkin, D. Ponsa, E. Rublee, and G. Bradski, "Kornia: an open source differentiable computer vision library for pytorch," in *Winter Conference on Applications of Computer Vision*, 2019. 2

[35] B. D. Brabandere, W. V. Gansbeke, D. Neven, M. Proesmans, and L. V. Gool, "End-to-end lane detection through differentiable least-squares fitting," *arXiv*, vol. 1902.00293, 2019. 2

[36] C. Tang and P. Tan, "Ba-net: Dense bundle adjustment network," *ICLR*, 2019. 2

[37] R. Clark, M. Bloesch, J. Czarnowski, S. Leutenegger, and A. J. Davison, "Ls-net: Learning to solve nonlinear least squares for monocular stereo," *ECCV*, 2018. 2, 5

[38] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, 12 1997. 2

[39] L. von Stumberg, P. Wenzel, Q. Khan, and D. Cremers, "Gn-net: The gauss-newton loss for deep direct SLAM," *CoRR*, 2020. 2

[40] L. Han, M. Ji, L. Fang, and M. Nießner, "Regnet: Learning the optimization of direct image-to-image pose registration," *arXiv*, vol. 1812.10212, 2018. 2

[41] B. Amos and J. Z. Kolter, "OptNet: Differentiable optimization as a layer in neural networks," in *ICML*, 2017. 2

[42] E. Grefenstette, B. Amos, D. Yarats, P. M. Htut, A. Molchanov, F. Meier, D. Kiela, K. Cho, and S. Chintala, "Generalized inner loop meta-learning," *arXiv:1910.01727*, 2019. 2

[43] M. Lampton, "Damping–undamping strategies for the levenberg–marquardt nonlinear least-squares method," *Computers in Physics*, 1997. 3

[44] F. Richards, "A flexible growth function for empirical use," *Journal of experimental Botany*, 1959. 3

[45] F. Steinbrücker, J. Sturm, and D. Cremers, "Real-time visual odometry from dense rgb-d images," in *ICCV Workshops*, 2011. 3

[46] P. J. Bes, N. D. McKay, *et al.*, "A method for registration of 3-d shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992. 3

[47] C. V. Nguyen, S. Izadi, and D. Lovell, "Modeling kinect sensor noise for improved 3d reconstruction and tracking," in *2012 second international conference on 3D imaging, modeling, processing, visualization & transmission*. IEEE, 2012. 4

[48] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," 1996. 4, 5

[49] V. Sitzmann, M. Zollhöfer, and G. Wetzstein, "Scene representation networks: Continuous 3d-structure-aware neural scene representations," in *NEURIPS*, 2019. 4

[50] S. Tulsiani, T. Zhou, A. A. Efros, and J. Malik, "Multi-view supervision for single-view reconstruction via differentiable ray consistency," in *CVPR*, 2017. 4

[51] J. Gwak, C. B. Choy, M. Chandraker, A. Garg, and S. Savarese, "Weakly supervised 3d reconstruction with adversarial constraint," in *International Conference on 3D Vision*, 2017. 4

[52] H. Igehy, "Tracing ray differentials," in *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., 1999, pp. 179–186. 4

[53] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "Scannet: Richly-annotated 3d reconstructions of indoor scenes," in *CVPR*, 2017. 5, 6

[54] A. Dai, M. Nießner, M. Zollöfer, S. Izadi, and C. Theobalt, "Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface re-integration," *ACM Transactions on Graphics 2017 (TOG)*, 2017. 5

[55] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *IEEE IROS*, 2012. 5, 6

[56] A. Handa, T. Whelan, J. McDonald, and A. Davison, “A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM,” in *IEEE ICRA*, 2014. 5