∇SLAM: Automagically differentiable SLAM

Krishna Murthy Jatavallabhula[1], Ganesh Iyer*[3], Soroush Saryazdi *[4], 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, [4]Concordia University

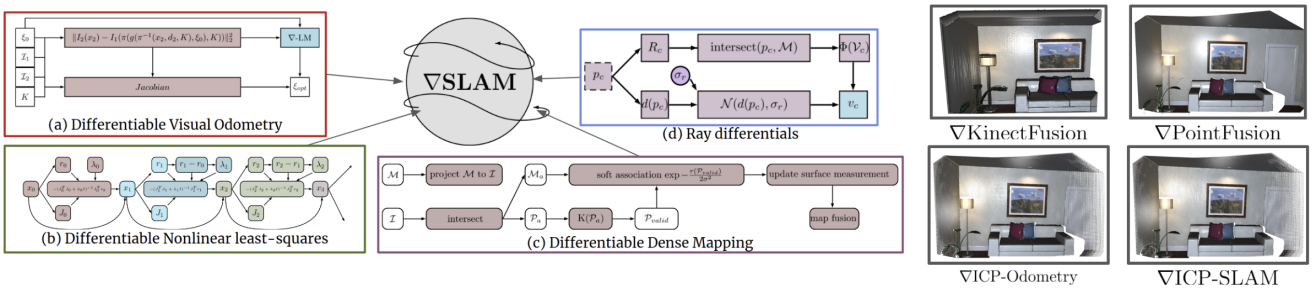

Figure 1. **∇SLAM (gradSLAM)** is a *fully differentiable* dense simultaneous localization and mapping (SLAM) system. The central idea of ∇SLAM is to construct a computational graph representing every operation in a dense SLAM system. We propose differentiable alternatives to several non-differentiable 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. ∇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). While learning-based approaches have the potential to leverage downstream tasks to learn representations, blending such approaches with "classical" SLAM systems has remained an open question. In this work, we propose ***∇SLAM (gradSLAM)****, 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. A short video explaining the paper and showcasing the results can be found here.*

1. Introduction

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* [8, 10, 11]. 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 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. In this paper, we argue for a declarative approach [7] 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.

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 [8, 11, 15]. The computational graph frame-

*Authors contributed equally

†No ∇students were harmed in the production of this manuscript.

work allows us to obtain a function ($\mathcal{S}$) that 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 $\nabla_s \mathcal{S}(s)\delta s$.

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.

While there has been progress in deep learning based SLAM systems [3], and declarative sub-components for SLAM [2, 6], 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.

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 [11]), surfel-based mapping (PointFusion [8]), and iterative closest point (ICP) mapping (ICP-SLAM). We also demonstrate examples of back-propagating error signals through the entire SLAM system that enable exciting avenues in representation learning for SLAM[1].

2. ∇SLAM

2.1. 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. Specifically, we propose differentiable alternatives to trust-region optimization (Sec. 2.2), dense mapping (Sec. 2.3)), measurement fusion and raycasting (Sec. 2.4).

2.2. ∇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.

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

Trust-region methods such as Levenbarg-Marquardt (LM) are commonly used to optimize these objectives. These methods are not differentiable as at each optimization step, they involve recalibration of optimizer parameters, based on a *lookahead* operation over subsequent iterates [9]. 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 smooth gradients with respect to the optimizer hyperparameters.

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 [12] 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 [12]. $[\lambda_{min}, \lambda_{max}]$ 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)).

∇LM allows differentiable realizations of several SLAM components, such as dense visual odometry estimation [13] (Fig. 1(a)), and Iterative Closest Point (ICP) alignment [1].

2.3. Differentiable Mapping

The *mapping* process in SLAM involves a number of differentiable yet non-smooth operations (clipping, indexing, thresholding, new/old decision, active/inactive decision, etc.). We enforce differentiability in this process by the following corrective measures.

1. The surface measurement made at each valid pixel p in the live frame 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 *dif-*

Figure 2. **Qualitative results** on sequences from the ScanNet [4] dataset. 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 [5].

ferentiable fusion step (*c.f*. Sec 2.4).

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 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. The computational graph for this differentiable mapping process is illustrated in Fig. 1(c).

2.4. Other differentiable modules

We also repurpose existing differentiable modules for measurement fusion [8, 11] and raycasting [14].

3. 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* [11] algorithm that constructs TSDF-based volumetric maps, the *PointFusion* [8] algorithm that constructs surfel maps, and a pointcloud-based SLAM framework that we call *ICP-SLAM*. Fig. 2 shows qualitative reconstruction results on sequences from ScanNet [4], and Table 1 shows that the accuracy of these differentiable SLAM systems is similar to their non-differentiable counterparts.

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

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

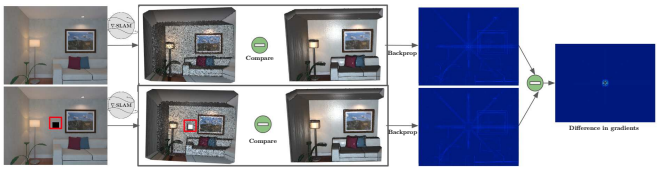

Figure 3. **Analysis of gradients**: ∇SLAM enables gradients to flow through to the input images. *Top*: An RGB-D image pair (depth not shown) is passed through ∇KinectFusion. The resulting map is compared with a precise (ground-truth) map. The comparision error is backpropagated through the SLAM system, 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.

4. Experiments and results

4.1. Differentiable Optimization

We design a test suite of nonlinear curve fitting problems to measure the performance of ∇LM (Sec 2.2) 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 2 how ∇LM performs similarly to LM (a slight performance drop is noticeable, due to smoothing).

4.2. 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. 3, the top row shows an RGB-D image differentiably transformed—

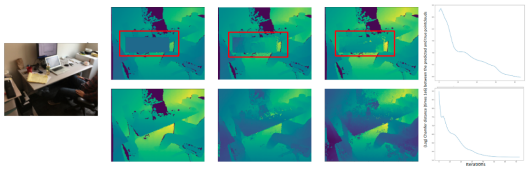

Figure 4. **End-to-end gradient propagation**: (*Top*): A chunk of a depth map is *chopped*. The resultant sequence is reconstructed using ∇PointFusion and the pointcloud is compared to a *clean* one reconstructed using the unmodified depth map. The Chamfer distance between these two pointclouds is used to define a reconstruction error between the two pointclouds, which is backpropagated through to the input depth map and updated by gradient descent. (*Bottom*): Similar to the Fig. 3, we show that ∇SLAM can *fill-in* holes in the depthmap by leveraging multi-view gradient information.

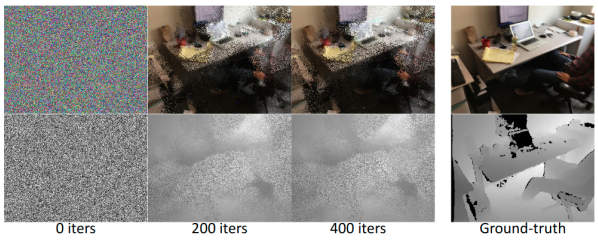

Figure 5. **RGB-D completion using end-to-end gradient propagation**: Three RGB-D images and a *noise image* are passed through ∇PointFusion, and compared to a clean reconstruction obtained from four RGB-D images. The reconstruction loss is used to optimize the *noise image* by gradient descent. We can 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.

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. Element-wise 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. 4, 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. 5.

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

5. Conclusion

We introduce ∇SLAM, a declarative, computational graph approach to SLAM. ∇SLAM enables gradient-based learning for localization and mapping based tasks by providing explicit gradients with respect to the input sensor observations. We showcase how the gradients propogate through the tracking, mapping, and fusion stages. Future efforts will focus on learning ∇SLAM components to optimize downstream task performance. ∇SLAM potentially enables a variety of self-supervised learning applications, by equipping gradient-based learning architectures with *spatial understanding*.

References

[1] Paul J Bes, Neil D McKay, et al. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992. 2

[2] Bert De Brabandere, Wouter Van Gansbeke, Davy Neven, Marc Proesmans, and Luc Van Gool. End-to-end lane detection through differentiable least-squares fitting. *arXiv*, 1902.00293, 2019. 2

[3] Jan Czarnowski, Tristan Laidlow, Ronald Clark, and Andrew Davison. Deepfactors: Real-time probabilistic dense monocular slam. In *IEEE RAL*, 2020. 2

[4] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, 2017. 3

[5] Angela Dai, Matthias Nießner, Michael Zollöfer, Shahram Izadi, and Christian Theobalt. Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface re-integration. *ACM Transactions on Graphics*, 2017. 3

[6] Ravi Garg, Vijay Kumar BG, Gustavo Carneiro, and Ian Reid. Unsupervised cnn for single view depth estimation: Geometry to the rescue. In *ECCV*, 2016. 2

[7] Stephen Gould, Richard Hartley, and Dylan Campbell. Deep declarative networks: A new hope. *arXiv:1909.04866*, 2019. 1

[8] Maik Keller, Damien Lefloch, Martin Lambers, Shahram Izadi, Tim Weyrich, and Andreas Kolb. Real-time 3d reconstruction in dynamic scenes using point-based fusion. In *International Conference on 3D Vision*, 2013. 1, 2, 3

[9] Michael Lampton. Damping–undamping strategies for the levenberg–marquardt nonlinear least-squares method. *Computers in Physics*, 1997. 2

[10] Raúl Mur-Artal and Juan D. Tardós. ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras. *IEEE TRO*, 33(5):1255–1262, 2017. 1

[11] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohli, Jamie Shotton, Steve Hodges, and Andrew W Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *ISMAR*, 2011. 1, 2, 3

[12] FJ Richards. A flexible growth function for empirical use. *Journal of experimental Botany*, 1959. 2

[13] Frank Steinbrücker, Jürgen Sturm, and Daniel Cremers. Real-time visual odometry from dense rgb-d images. In *ICCV Workshops*, 2011. 2

[14] Shubham Tulsiani, Tinghui Zhou, Alexei A. Efros, and Jitendra Malik. Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *CVPR*, 2017. 3

[15] Thomas Whelan, Michael Kaess, Hordur Johannsson, Maurice Fallon, John J Leonard, and John McDonald. Real-time large-scale dense rgb-d slam with volumetric fusion. *IJRR*, 2015. 1