

Single-view TSDF Mesh Noise Reduction under Virtual Light using Differentiable Rendering

PILJOONG JEONG, Gwangju Institute of Science and Technology, Republic of Korea

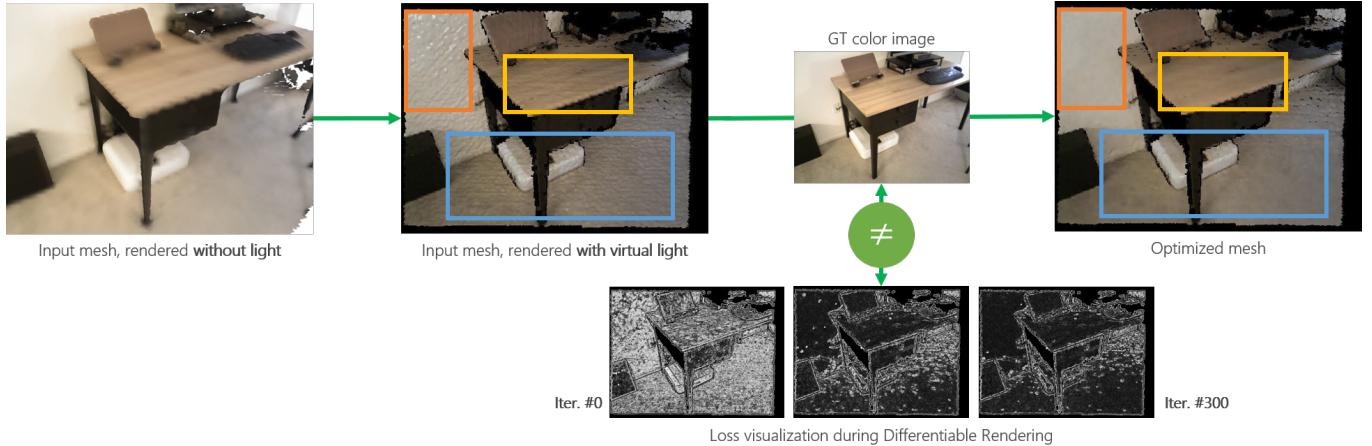


Fig. 1. Overview of our method. (From top left) We saturate noisy vertices by rendering input TSDF mesh with virtually placed light source. We extract common-shared geometric clue between rendered input mesh and target color image. Differentiable renderer iteratively minimizes loss, which is the difference between clues from rendered input mesh and target Ground Truth color image. Orange, Yellow, and Blue inset shows difference between input mesh and optimized mesh. Our method successfully reduces noise in mesh vertices. Furthermore, our provided video shows input mesh that is being optimized (right-side of the video) as iteration continues, as well as the visualization of loss (left-side of the video) that is being minimized: https://drive.google.com/file/d/10F_I89m5O-RWOIxocYoxG2QOkJc7YWIF/view?usp=sharing

Thanks to consistent evolution of SLAM (Simultaneous Localization and Mapping) and its related technologies, we can reconstruct geometric properties of where we are currently observing in real-time. Due to the limitation of current depth sensing hardware, however, we are generally able to obtain geometric features corrupted by noise. Color image is perceived as geometrically noise-free in terms of human vision-perception system, but to our best knowledge, encoding the information from Ground Truth for differentiable rendering under single view constraint is not discussed yet. In this report, we propose a bridge between the geometric information generated from color image and rendered mesh, so that differentiable renderer can optimize input i.e., noisy vertex position without any depth supervision. The key insight is that we can highlight noisy vertices by rendering mesh with virtually placed light sources. We compare our result with one of the state-of-the-art differentiable rendering method[1], and show our method outperforms previous method which requires prior depth information (silhouette image).

CCS Concepts: • Computing methodologies → Shape modeling; Mixed / augmented reality.

Author's address: PilJoong Jeong, Gwangju Institute of Science and Technology, 123 Cheomdangwagi-ro, Buk-gu, Gwangju, 61005, Republic of Korea, piljoong.jeong@gm.gist.ac.kr.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2021 Association for Computing Machinery.
0730-0301/2021/6-ART \$15.00
<https://doi.org/10.1145/nnnnnnnn.nnnnnnnn>

Additional Key Words and Phrases: Differentiable rendering, Mesh denoising

ACM Reference Format:

Piljoong Jeong. 2021. Single-view TSDF Mesh Noise Reduction under Virtual Light using Differentiable Rendering. *ACM Trans. Graph.* 1, 1 (June 2021), 7 pages. <https://doi.org/10.1145/nnnnnnnn.nnnnnnnn>

1 INTRODUCTION

To satisfy increasing demand of Augmented Reality (AR) technology, we required to obtain geometric information of observed real scene as precise as possible. As precision of geometric detail is increased, we can augment virtual objects with more consistency from computer graphics field (rendering) perspective, and we are able to estimate camera poses from perfect correspondences from computer vision field (SLAM) perspective.

Capturing detailed geometry is still remain as technical challenge due to hardware/computational limitation of SLAM. **PJ: TODO: actually, this is not true. Fine voxels can represent geometric detail but easily hindered by noise, whereas coarse mesh has resilience against noise but fails to represent where geometric detail is required e.g., edges** It is obvious that consumer-level depth sensing technology still has noisy observation. We can cope this by divide a real scene with detailed (fine) voxels as precise as possible when generating TSDF mesh. In this way, however, unnecessarily large number of triangles are generated even in the case of reconstructing simple geometry (e.g., plane). This directly affects to rendering performance, as rendering requires primitive traversal in order to appropriately propagate lighting information

for every single frame. Due to those limitations, we are compromised to use TSDF meshes from coarse voxels, which have geometric incorrectness including noise. Nevertheless, there is strong demand to perfect geometry captured from SLAM sequences.

Recent advances of Differentiable Rendering make it possible to optimize current input parameters by observing a set of given Ground Truth images. This seems promising to SLAM, as they naturally capture Ground Truth color images whereas generating input TSDF mesh corrupted by noisy measurements. However, it is unclear that how to interpret (perceptually encoded) geometric clues within color images, reflect those information into actual geometry to minimize its imperfection. Moreover, there is some limitations hinder directly applying previous differentiable rendering techniques into SLAM dataset. We explained such difficulties in 2.

Our main contribution is bridging the gap between perceptual noise-free geometric features from C and noisy geometric feature in M. To exploit this, we borrow a novel concept of image denoising using flashlight. Images taken with flashlight can hold additional features which are hard to detect from general geometric feature (e.g., depth, normal etc). For example, in [Eisemann and Durand 2004] [Petschnigg et al. 2004] flashy photography is used to enhance images taken from scene which has insufficient lighting condition. [Moon et al. 2013] is pioneering work that adopt image enhancement using flashlight on photorealistic rendering domain, by casting virtual flashlights to capture a scene’s reflective / refractive features, which are not stored in traditional G-buffers. Based on these approaches, we saturate noisy vertices by casting virtual light, which are never detected when rendered with mesh’s albedo only. Please refer the leftmost image (rendered without light) and its connected image (rendered with virtual light) in 1 for details. We demonstrate our results, compare with result from previous method, and show that our method outperforms previous result.

2 PREVIOUS WORKS

TSDF Mesh Noise Reduction in SLAM. Starting from pioneering work [Curless and Levoy 1996], estimating true depth from noisy measurements has been one of the major challenges in SLAM. Basically, [Curless and Levoy 1996] first introduced the definition of ‘fusion’, which is accumulating noisy world positions into a voxel. This acts as similar with spatial running average filter, therefore it is known that accumulating frames that captures same region can reduce noise incrementally. However, this takes a lot of time to converge depth values to a true mean, which interferes practical AR application experience. Therefore, various depth noise reduction techniques are applied to SLAM systems, including simple bilateral filter [Newcombe et al. 2011], merging depth information of neighboring frames either offline [Choi et al. 2015][Zhou et al. 2018], or online [Cao et al. 2018][Yang et al. 2020]. All mentioned previous works require multiple depth frames in order to generate reliable mesh which takes a long time, and they did not consider color images, which holds perceptive geometric information. Our method, in contrast, is able to infer geometric clue in color image, thus able to de-noise input mesh without multiple frames.

Differentiable Rendering. Differentiable rendering is the technique that optimizing input parameters by observing a set of target images

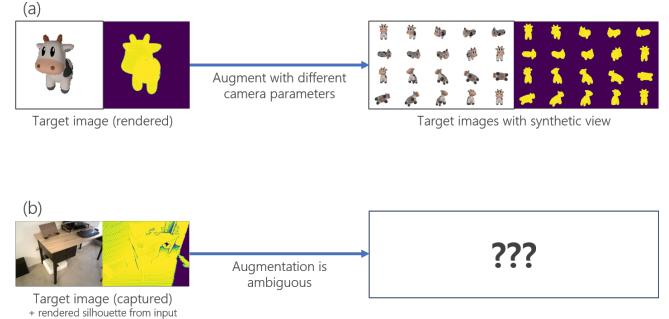


Fig. 2. Different setup between optimizing simple mesh and TSDF mesh from SLAM. Previous works aimed to optimize input geometry to set of target images captured with different camera view either synthetically[Ravi et al. 2020][Jatavallabhula et al. 2019][Laine et al. 2020][Nimier-David et al. 2019] or by taking calibrated real photographs[Nimier-David et al. 2019]. However, in the case of TSDF mesh from SLAM it is ambiguous since (1) input is not separated with its backgrounds, hence silhouette image is hard to generate (2) we cannot generate synthetic target views (3) it is hard to sample real images from SLAM sequences which captures a region that input mesh represents, since the labeled camera pose paired with target image is estimated value. It is well-known that camera poses from SLAM is inaccurate. (a) It is straightforward to generate synthetic target images if the target image can be rendered. (b) Unlike (a), it is much hard to augment target images in order to optimize indoor TSDF mesh.

via iteratively render a scene with current state of input parameters (i.e., forward pass), as well as propagate gradients along with computation graph (i.e., backward pass) built at forward pass. Due to the fact that it does not require any prior knowledge (e.g., pre-trained model trained from external dataset) other than target images, this seems an off-the-shelf optimizer for SLAM since we naturally capture target image during scanning process, whereas fused mesh is contaminated with noise due to its nature limitations. However, there are ambiguities that have to be considered in order to adapt differentiable rendering to optimize SLAM problems. 2 describes the different setup between simple mesh optimization and indoor TSDF mesh optimization. Our method bypasses those ambiguities via inferring geometric clue without necessity of silhouette information, as well as multiple target observations.

3 METHODS

We propose an optimization procedure that incrementally minimizes mesh vertex noises using differentiable rendering. Suppose we have vertices $V = \{V_i \in \mathbb{R}^3\} = \{V_0 \dots V_n\}$ within input mesh \mathcal{M} which is fused from input color C and depth \mathcal{D} with intrinsic K such that $\mathcal{M} = K^{-1}(C \oplus \mathcal{D})$. Here, \oplus denotes image registration operator. We pre-computed \mathcal{M} from popular off-line SLAM framework [Zhou et al. 2018] with voxel size 2cm. We define deformed vertices V_d , which has same shape as V initialized with zero values i.e., $V_d = \{\mathbf{0}_0, \dots, \mathbf{0}_n\}$. For every iteration, V_d is optimized so that resulting vertices $V_o = V + V_d$ approximate noise-free geometry which looks perceptually similar with C . **PJ: Emphasize minimizing difference between GT color and rendered color.** The main

problem is how to find common geometric representations between C and C to figure out which region is to be optimized (i.e., noisy).

Assumptions. Our goal is to minimize vertex noise of TSDF mesh \mathcal{M} fused from a single pair of indoor RGB-D images $C \oplus \mathcal{D}$. Given this scenario, we assume that we known camera intrinsic K and extrinsic $T = I \in \mathbb{R}^{4 \times 4}$. We will also consider the captured real scene is Lambertian, as all SLAM datasets are generated under Lambertian-dominant assumption to guarantee accurate camera pose estimate, unless additional consideration of non-Lambertian scene is introduced. **PJ: TODO: refer [Whelan, 2018]** We additionally constrain that there are no strong texture changes under geometrically close, flat surfaces.

3.1 Interpreting Perceptually Noise-free Geometry from Color Image

PJ: Too verbose! TODO: add proof Our key observation is strong radiance change within color images C and C , is only occurred from strong geometric displacements around a corresponding pixel. This is possible since we made two assumptions: Lambertian-dominant scene, and no strong texture changes under flat surface. Consider C is generated from perfect renderer which are capable of tracing full light transport, under perfect geometries. Although we cannot know which lighting conditions are used to render C , however, at least we know the lighting is somehow applied to perfect geometries since C is rendered, as well as we are able to perceive geometric information by just observing C . Using the fact that C is a discretization of continuous radiance function, we can intuitively notice that changing lighting condition never affects to local radiance change (unless the light is very close to the surface). From this intuition, we can render \mathcal{M} over virtually placed light source(s). If \mathcal{M} is noise-free enough, there would be no highlighted pixel in its rendering C . However, due to the fact that C is rendered with C from noisy measurement \mathcal{D} , there exist a set of saturated pixel, which are actually not saturated within C . Based on this observation, we decided to minimize gap between saturated pixels in C and unsaturated i.e., noise-free pixels in C . In order to do this, it is required to detect noise-free pixels in C , as well as noisy pixels in C , and set the difference as loss function so that our differentiable renderer properly minimizes gap.

3.2 Detecting Geometric Information given Color Image and Its Rendering

We simply detect noise-free surfaces within C by applying Scharr gradient kernel. Under the assumption that the scene is Lambertian-dominant and there is no sudden strong texture changes around the same surface, we found that this is enough to capture which region is smooth in terms of radiance (i.e., geometrically noise-free). In order to ensure robustness on strong texture changes one may combine with gradient from \mathcal{D} , but for this report we only experimented with gradients from C . Specifically, Scharr kernel for each axis over an image is defined as

$$G_x = \begin{pmatrix} -3 & 0 & 3 \\ -10 & 0 & 10 \\ -3 & 0 & 3 \end{pmatrix}, G_y = \begin{pmatrix} -3 & -10 & -3 \\ 0 & 0 & 0 \\ 3 & 10 & 3 \end{pmatrix} \quad (1)$$

, and our detected geometric changes over Ground Truth color image \tilde{G}_C is a Scharr gradient of I_C i.e., the intensity image from C . Since resulting gradient on some pixels have negative value of identical magnitude to positive values of neighboring pixel, we take its absolute value

$$\tilde{G}_C = \frac{1}{2} (|G_x(I_C)| + |G_y(I_C)|) \quad (2)$$

In order to detect noisy vertices over TSDF mesh, we propose Lightweight map to determine which vertex has stronger noise compare to \tilde{G}_C . Lightweight map I_{lw} is an image taken from identical camera setup to C , but holds how much a pixel corresponds to a hitpoint is affected by virtual light at a shading stage. I_{lw} is defined as

$$I_{lw} = \{I_{lw,i} \in \mathbb{R}^{W \times H}\}, I_{lw,i} = \frac{(x_i - p_0) \cdot n_i}{d(x_i, p_0) + \epsilon}, \quad (3)$$

where x_i, n_i is world position and normal of hitpoint with pixel index i , respectively. p_0 is position of virtual light, and $d(x_i, p_0)$ is Euclidean distance between hitpoint and light position. We obtain changing amount of each lightweight value \tilde{G}_{lw} by applying Scharr kernel over I_{lw} , similar with \tilde{G}_C .

$$\tilde{G}_{lw} = \frac{1}{2} (|G_x(I_{lw})| + |G_y(I_{lw})|) \quad (4)$$

We found that using geometric normal to calculate lightweight map saturate pixels around noisy vertices rather than shading normal, as shading normal smooths normal of each hit-point using neighboring vertex normal and its barycentric coordinates. In detail, pixels within same face have similar lightweight values since they are both geometrically close to each other, and they share same normal. Pixels that are geometrically close, but within different faces are highly likely to have similar values if two faces have near identical normal values. This is the case when two faces are considered as ‘flat’ to each other, meaning that shared vertices have no noise. As the vertex have bigger noise, the gap or normal between sharing two faces also gets bigger. This brings pixels in G_{lw} have large value where there is significant normal difference, meaning that the region has noisy vertex. Note that larger G_{lw} at a pixel means that the pixel has higher noise, therefore differentiable renderer can optimize the region more aggressively. This is illustrated in Figure 3. We observed that the intensity image of rendered scene with virtual light I_C serves similar role with I_{lw} as they properly reflect strong gradients around deviating normal. Therefore, we note that for all results we used I_C instead of I_{lw} .

3.3 Optimization using Differentiable Rendering

We define lightweight loss to minimize the difference between noise-free geometric information \tilde{G}_C and noise-detected rendered image \tilde{G}_C . We found that there is gradient value range inconsistency between \tilde{G}_C and \tilde{G}_C , as they are derived from different type of image i.e., color and geometry, respectively. We applied hyperbolic tangent kernel to each gradient image to ensure that both images have normalized value range, and we observed that this helped optimizer to find optimal without failure. Finally, we replace our target GT image from C to $C \oplus \mathcal{D}$, as \mathcal{M} follows holes where pixels in \mathcal{D} have

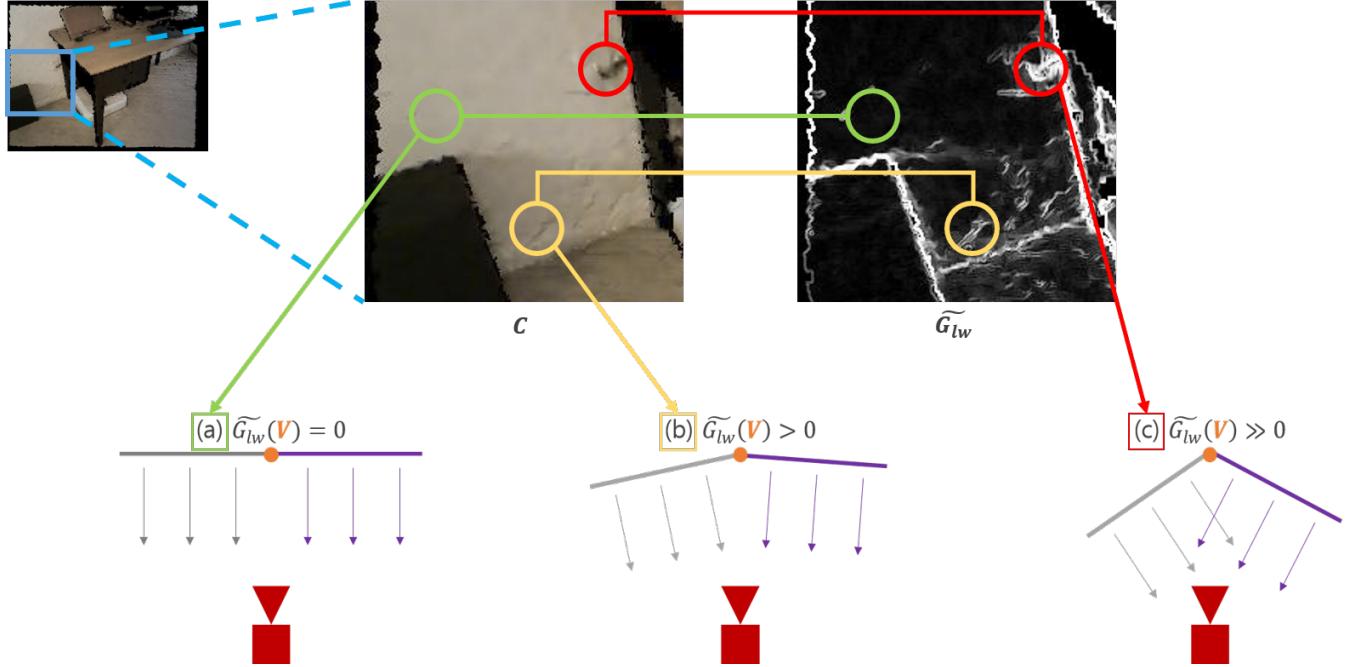


Fig. 3. Relationship between G_{lw} and actual noise of vertex V . We manually selected three pixels, where each have different magnitude of G_{lw} . We also illustrated 1D example of the relationship. Gray and Purple lines and arrows indicate each face and its geometric normal. (a) V is considered as noise-free since G_{lw} is evaluated as zero, meaning adjacent faces have identical normal value. (b) G_{lw} is increased as two faces have inconsistent normal. (c) as G_{lw} gets bigger, V is considered as highly-noisy vertex. From these examples, we can say that G_{lw} precisely indicates where noisy pixels exist.

zero value. Our final geometric gradients are:

$$G_C = \tanh(\tilde{G}_{C \oplus D}), G_{lw} = \tanh(\tilde{G}_{lw}), \quad (5)$$

where $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$. Our optimizer minimizes lightweight loss representing geometric difference, while penalizing vertices not to evolve too far from its initial position:

$$\mathcal{L} = w_{lw} \cdot L_{lw} + w_{pos} \cdot L_{pos}, \quad (6)$$

$$L_{lw} = \|G_C - G_{lw}\|_2^2, \quad (7)$$

$$L_{pos} = \|V - (V_o)\|_2^2 = \|V_d\|_2^2 \quad (8)$$

For all results, we used $w_{lw} = 0.01$ and $w_{pos} = 1.0$. Fig. 4 visualizes optimization procedure.

4 RESULTS

We have implemented our method on top of general differentiable renderer[Ravi et al. 2020]. For all tests, we used Intel Xeon E5-2687W CPU machine with 3.0GHz, and GeForce RTX 2080 TI for CUDA-optimized differentiable renderer. We optimized our mesh with Stochastic Gradient Descent method with learning rate = 1.0, and momentum = 0.9. We placed one virtual light source at the camera center for all experiments, although this method is up to a number of lights and its positions.

4.1 Comparison with Previous Work

We compare our method with standard differentiable rendering approach used in differentiable renderer[Liu et al. 2019]. We slightly

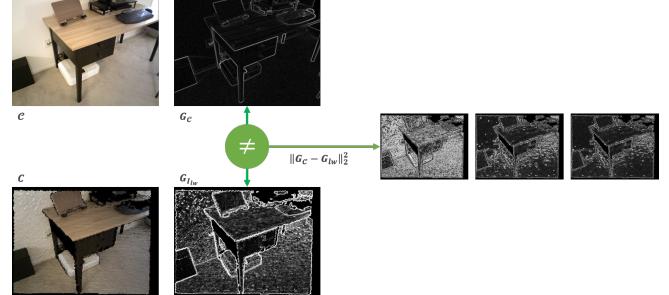


Fig. 4. Optimization procedure of our differentiable rendering. From generated target noise-free geometric clue G_C and input noisy geometric clue G_{lw} , we minimize L_2 distance between two clues. Note that we additionally penalize aggressive vertex evolve, however it is skipped in the figure.

modified its workflow in order to evaluate performance under identical conditions (please refer 2 for details). Specifically, we limit amount of target view to 1, as it is impossible to synthetically generate clean target using existing SLAM sequences. For silhouette image, we used silhouette of input TSDF mesh as silhouette itself does not take into account detailed depth information as they represent boundary of mesh PJ: TODO: to footnote(Hence, previous methods required silhouette images from different camera extrinsic in order to optimize the shape of input mesh). We follow losses and its weights as suggested in previous method[Ravi et al. 2020], which

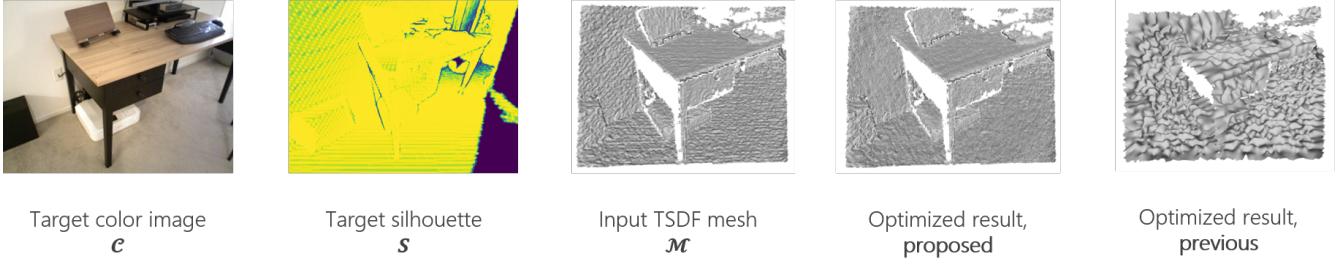


Fig. 5. Comparison of result using previous method and our method. For target silhouette, we generated it from input mesh by render mesh with silhouette renderer, and it is enough to use since silhouette image itself does not hold any depth information other than boundary information of target mesh. **Previous** result fails to converge to perceptive geometric shape hidden in target images, whereas **Proposed** result successfully reduces noise. Note that **Proposed** method only observes C in order to infer noise-free geometry, whereas Previous method requires both C and S .

is defined as:

$$\mathcal{L}_{prev} = w_{sil} \cdot L_{sil} + w_{rgb} \cdot L_{rgb} + w_{edge} \cdot L_{edge} + \\ w_{normal} \cdot L_{normal} + w_{laplacian} \cdot L_{laplacian},$$

where L_{sil} , L_{rgb} is silhouette, and color difference between input and target, respectively. L_{edge} , L_{normal} , and $L_{laplacian}$ stands for mesh edge length consistency, mesh normal orientation consistency, and mesh Laplacian loss, respectively. Note that mesh losses L_{edge} , L_{normal} , $L_{laplacian}$ is developed from traditional geometry processing field, thus they do not reflect geometric features that are shown in target images. For brevity, we skip detailed equations for each loss. We used $w_{sil} = 1.0$, $w_{rgb} = 1.0$, $w_{edge} = 1.0$, $w_{normal} = 0.01$, $w_{laplacian} = 1.0$ as authors suggested. Fig. 5 shows optimized mesh with loss from previous work. Resulting mesh preserves its silhouette information, as they provided silhouette loss to prevent vertices to evolve out of its silhouettes. However, vertices within silhouette images are evolved without any shape constraints supervised by target images. Those vertices are guided to satisfy input mesh's internal properties, as mesh losses does not consider geometric clues in target images. This results in optimized mesh failed to optimize, while only preserving boundary.

4.2 Validation of Proposed Method

In Figure 6, we plotted our loss graph to ensure that our loss function forms a convex function. Our proposed lightweight loss is monotonically decreased as the optimization continues. Note that our positional loss is not minimized, as we want positional loss to act similar with regularization term, which prevents evolving vertices getting not to far from its original position. Nevertheless, the total loss forms a convex function as the actual contribution of positional loss is small enough than lightweight loss. **PJ: TODO: validate with other scenes.**

4.3 Limitations & Future Directions

We summarize future directions of our method, including failure case and promising extension of proposed method.

Adding Robust Positional Constraint Loss. Although our method can drastically reduce mesh noise, we found that optimized mesh is not maintaining topologically correct structure. Specifically, we

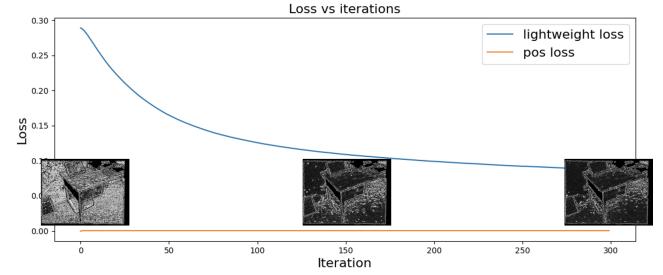


Fig. 6. Loss plot with loss images at specific iterations (here, 0, 150, and 300 is used). Our lightweight loss successfully minimizes vertex noise by observing target noise-free geometric information G_C . Note that L_{pos} never converges to zero, as they act as constraint to vertices not to evolve too far from its original position, thus have non-zero values among all iterations.

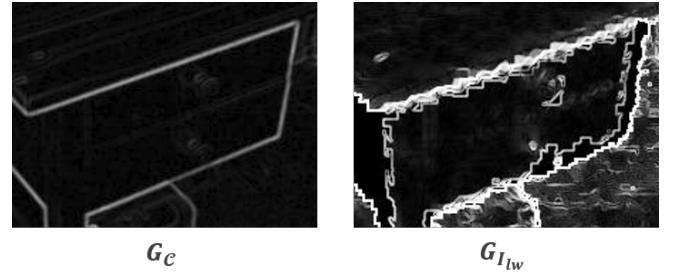


Fig. 7. Inset of each gradient. $G_{I_{lw}}$ is corrupted since geometric normal deviates around edge.

observed self-intersections between neighboring faces. This problem can be naturally arisen since we did not consider relationship between neighboring vertices in our final loss. Therefore, though vertices are evolved to have reduced noise, their evolving direction is totally random, lead to occur self-intersections between neighboring faces by corresponding (incorrectly evolved) neighboring vertices. We illustrated this failure case in Figure 7.

Edge Fitting. Although our method is able to minimize noisy vertices by treating G_C as noise-free geometric clue, however it

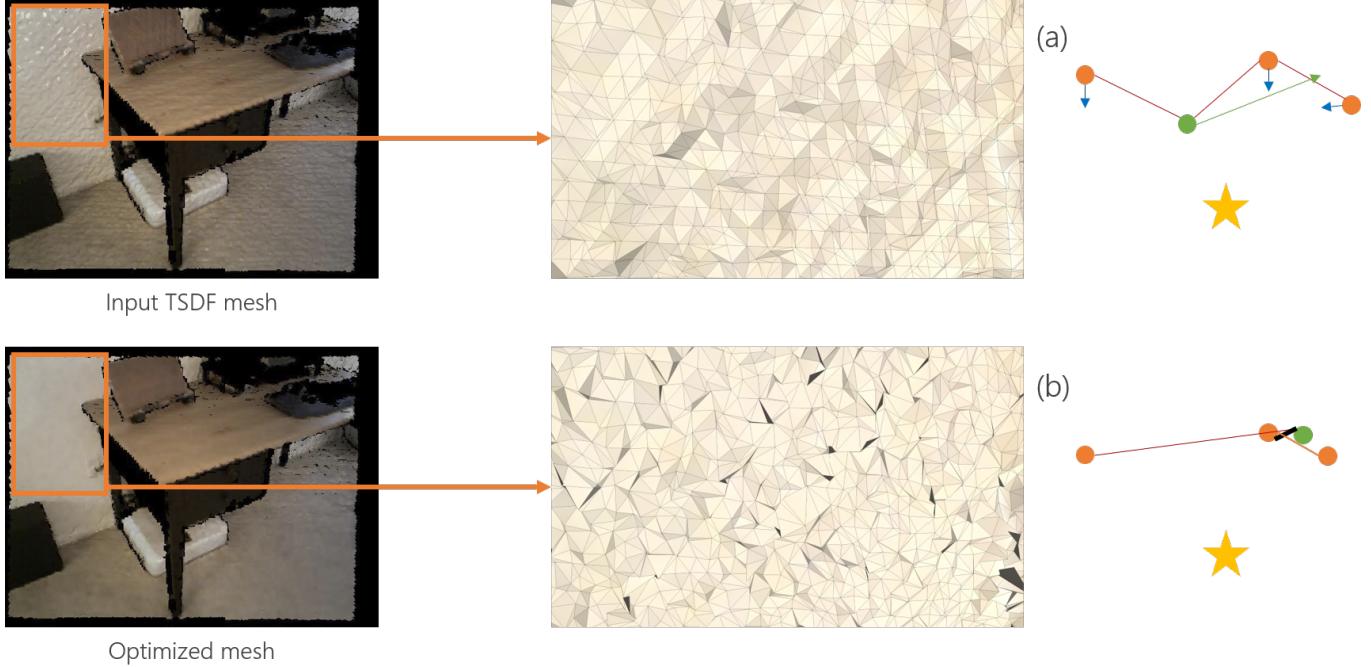


Fig. 8. Comparison of input mesh and optimized mesh. We zoomed up each geometry corresponds to image inset. **Black artifacts** are observed in zoomed view in optimized mesh. We found that those are self-intersection between neighboring faces, since our positional constraint L_{pos} does not consider distances between neighboring vertices. This phenomenon is illustrated at the rightmost images of each zoomed view. (a) For each iteration, vertices are optimized while only considering flatness of surfaces, not for topological correctness of geometry. Star, circle, and arrow indicates light position, vertices, and gradient (evolving direction after current optimization step), respectively. Here, **Green vertex** is optimized to **Green arrow** direction so that it occludes another face. (b) Consequently, there exists a self-intersection between neighboring faces (**Black line**) although the entire mesh noise is reduced, resulting self-occluded artifact.

failed to clean out noisy vertices around geometric edges. We found that there is gradient magnitude difference between two images, as on edges G_{lw} tends to have large deviation since it is the place where geometric normal hugely differs. Our current L_2 loss of gradients cannot reflect this case, as they are evaluated pixel-by-pixel. We visualized this case in Figure 8. Weighting color gradients by using gradients from depth, say G_D , or developing an erosion kernel for G_C to reliably cover G_{lw} seems worth trying.

5 CONCLUSION

In this report, we proposed a method that optimizes input TSDF mesh generated from single RGB-D pair, by exploiting noise-free geometric clues hidden in color image. Our method differs from most of previous methods as they required to optimize shape from multiple takes of target scene, which is hard to directly apply to SLAM dataset. We show that our method outperforms previous differentiable rendering setup. Future work on improving robustness in terms of optimizing edgy geometries, preventing topological correctness of input mesh would be beneficial to solve problem that is defined as ill-posed in SLAM.

REFERENCES

- Yan-Pei Cao, Leif Kobbelt, and Shi-Min Hu. 2018. Real-time high-accuracy three-dimensional reconstruction with consumer RGB-D cameras. *ACM Transactions on*

- Graphics (TOG)* 37, 5 (2018), 1–16.
 Sungjoon Choi, Qian-Yi Zhou, and Vladlen Koltun. 2015. Robust reconstruction of indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5556–5565.
 Brian Curless and Marc Levoy. 1996. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. 303–312.
 Elmar Eisemann and Frédo Durand. 2004. Flash photography enhancement via intrinsic relighting. *ACM transactions on graphics (TOG)* 23, 3 (2004), 673–678.
 Krishna Murthy Jatavallabhula, Edward Smith, Jean-François Lafleche, Clement Fuji Tsang, Artem Rozantsev, Wenzheng Chen, Tommy Xiang, Rev Lebaredian, and Sanja Fidler. 2019. Kaolin: A pytorch library for accelerating 3d deep learning research. *arXiv preprint arXiv:1911.05063* (2019).
 Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. 2020. Modular primitives for high-performance differentiable rendering. *ACM Transactions on Graphics (TOG)* 39, 6 (2020), 1–14.
 Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. 2019. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 7708–7717.
 Bochang Moon, Jong Yun Jun, JongHyebob Lee, Kunho Kim, Toshiya Hachisuka, and Sung-Eui Yoon. 2013. Robust image denoising using a virtual flash image for Monte Carlo ray tracing. In *Computer Graphics Forum*, Vol. 32. Wiley Online Library, 139–151.
 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.
 Merlin Nimier-David, Delio Vicini, Tizian Zeltner, and Wenzel Jakob. 2019. Mitsuba 2: A retargetable forward and inverse renderer. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 1–17.

- Georg Petschnigg, Richard Szeliski, Maneesh Agrawala, Michael Cohen, Hugues Hoppe, and Kentaro Toyama. 2004. Digital photography with flash and no-flash image pairs. *ACM transactions on graphics (TOG)* 23, 3 (2004), 664–672.
- Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. 2020. Accelerating 3d deep learning with pytorch3d. *arXiv preprint arXiv:2007.08501* (2020).
- Sheng Yang, Beichen Li, Yan-Pei Cao, Hongbo Fu, Yu-Kun Lai, Leif Kobbelt, and Shi-Min Hu. 2020. Noise-Resilient Reconstruction of Panoramas and 3D Scenes Using Robot-Mounted Unsynchronized Commodity RGB-D Cameras. *ACM Transactions on Graphics (TOG)* 39, 5 (2020), 1–15.
- Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. 2018. Open3D: A modern library for 3D data processing. *arXiv preprint arXiv:1801.09847* (2018).