Exploiting Color Image: Single-view Marching-cube Denoising using Differentiable Rendering

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

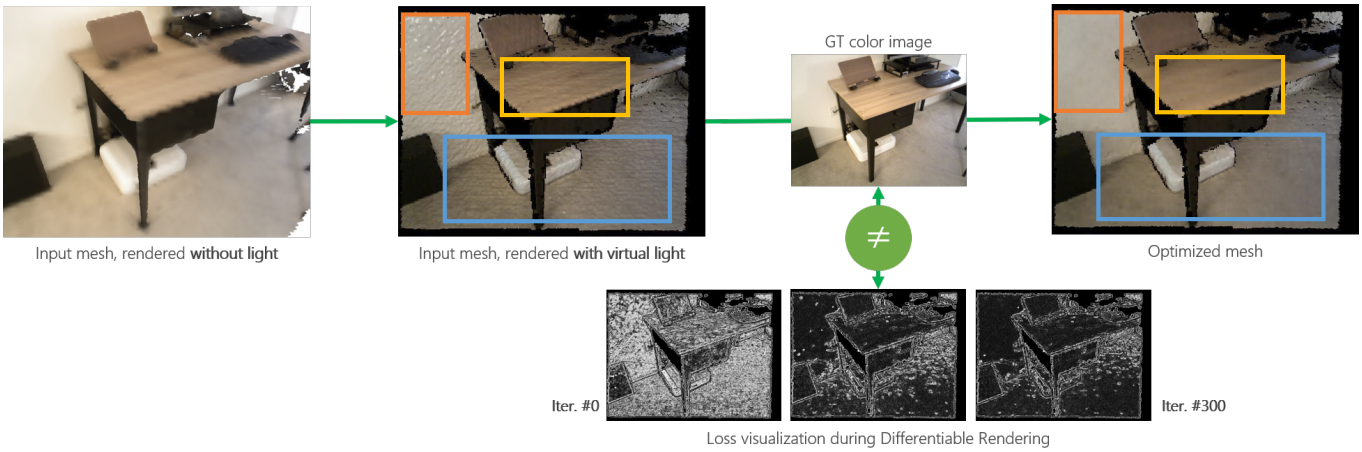

Fig. 1. Overview of our method. (From top left) We saturate noisy vertices by rendering input TSDF mesh with virtually placed light source. We extract common-shared geometric clue between rendered input mesh and target color image. Differentiable renderer iteratively minimizes loss, which is the difference 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-RWOIxocYoxG2QOkJc7YWlF/view?usp=sharing **PJ: TODO. change url to personal drive.**

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[Ravi et al. 2020], 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.

0730-0301/2021/7-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Additional Key Words and Phrases: Differentiable rendering, Mesh denoising

ACM Reference Format:
PilJoong Jeong. 2021. Exploiting Color Image: Single-view Marching-cube Denoising using Differentiable Rendering. *ACM Trans. Graph.* 1, 1 (July 2021), 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

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. It is obvious that consumer-level depth sensing technology still has noisy observation. We can cope this by divide a real scene with less detailed (coarse) voxels to smooting out noisy depth measurements when generating TSDF mesh. In this way, however, fails to retain geometric detail even in the case of reconstructing simple geometry (e.g., edge between planes). To preserve geometric detail, we still need to divide voxels as fine as possible where is geometrically complex. This directly affects to rendering performance, as rendering requires primitive traversal in order to appropriately propagates lighting information for every single frame. Due to those limitations, we are compromised to use TSDF meshes from acceptably 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 (Fig. 2).

Our main contribution is bridging the gap between perceptional noise-free geometric features from input color image and noisy geometric feature in input mesh. To exploit this, we interpreted input color image as a result of perfect renderer with perfect geometry. Under certain assumptions, we argued that radiance is consistent if a point and its neighbors are lie on a same plane. Generally, it is not true for virtually rendered image with input mesh, as the mesh is composed with noisy vertices. Therefore, we optimized input mesh to have radiance consistency i.e., noise-free geometry which is input color image have.

We demonstrate our results, compare with result from previous method, and show that our method outperforms previous result.

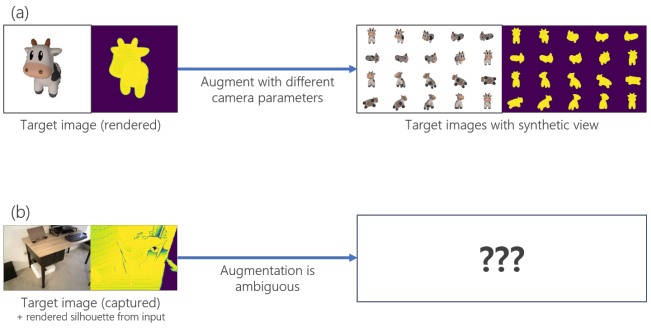

Fig. 2. Different setup between optimizing simple mesh and TSDF mesh from SLAM. Previous works aimed to optimize input geometry to set of target images captured with different camera view either synthetically[Ravi et al. 2020][Jatavallabhula et al. 2019][Laine et al. 2020][Nimier-David et al. 2019] or 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.

2 PREVIOUS WORKS

We clustered the related work into three categories. Note that any of those, are orthogonal to our work; Our proposed method can take output from any of those methods as input, and able to generate more plausible result.

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 accumulates 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. One may consider differentiable rendering technique to remedy noisy mesh, as during SLAM process we naturally capture a set of target color images while actual mesh is contaminated from a set of noisy measurements. Differentiable rendering optimize input parameters by observing a set of target images 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.

However, there are ambiguities that have to be considered in order to adapt differentiable rendering to optimize SLAM problems. Figure 2 describes the different setup between simple mesh optimization and indoor TSDF mesh optimization. **PJ: Elaborate how loss is determined in order to optimize mesh.** Our method bypasses those ambiguities via inferring geometric clue without necessity of silhouette information, as well as multiple target observations.

Shape Manipulation from Additional Features. **PJ: TODO. fill here: Geometry from image, Image enhancement from lighting?**

3 METHODS

We propose an optimization procedure that incrementally minimizes mesh vertex noises using differentiable rendering. Suppose we have vertices $V = \{V_i \in \mathbb{R}^3\} = \{V_0 ... V_n\}$ within input mesh $\mathcal{M}$ which is fused from input color $\mathcal{C}$ and depth $\mathcal{D}$ with intrinsic K such that $\mathcal{M} = K^{-1}(\mathcal{C} \oplus \mathcal{D})$. Here, $\oplus$ denotes image registration operator. We define deformed vertices V_d, which has same shape as V initialized with zero values i.e., $V_d = \{\mathbf{0}_0, ..., \mathbf{0}_n\}$. 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 $\mathcal{C}$. 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).

3.1 Exploting assumptions underlying existing SLAM datasets

We found two common aspects within C taken from most of SLAM dataset, which acts important role to consider C as geometric clue.
Lambertian-dominant. This naturally satiesfies since this is an assumption for vast majority of SLAM dataset, unless it is used for vaildating SLAM algorithm under non-Lambertian environment e.g., [Whelan et al. 2018]. In order to track accurate camera pose during SLAM, finding reliable feature correspondences are necessary. Generally, a point shown in two images that shares overlapped region C_0, C_1 is chosen, which satisfies photometric consistency e.g., $E_{photo} = \arg\min_x C_0\left(x + \overrightarrow{u}\right) - C_1(x)$. [Szeliski 2010] This is because it has been empirically considered as 'static anchor' on a given scene; a point on a surface never deforms during capturing stage, and its value is direction-invariant so that the point has consistent pixel value wherever an image is taken from.

Interpreting photometric consistency in computer graphics field is trivial; this indicates that a point is lie on a Lambertian surface, such that a bi-directional reflectance $f(x, w_i, w_o) = c$, where $c \leq 1$ is a constant.
No strong radiance changes within a plane. We found that there seldom have strong radiance changes within a plane among indoor SLAM datasets. Here, a plane can be a structural information of given scene (e.g., wall, floor, ceiling), or upper part of an object (e.g., desk, box). Although there may exists a case that a plane have strong radiance changes (e.g., chessboard pattern on floor), we decided to ignore them. In other words, any point within a plane have similar pixel value with neighbors lie on same plane. Note that our proposed method works even though this assumption does not met, however we decide to made it for the sake of ease of explanation.

3.2 Re-vising: How Geometry Affects to Color Image

In this section, we formally describe our intuition that enables input color image treated as noise-free geometric clue.

Let us consider C as a result from perfect renderer which is capable of tracing full light transport without any noise or outlier, given unknown lighting conditions and noise-free geometry. Based on Light Transport Equation(LTE) representation with respect to path integral form [Veach 1998], we represent C as a set of solution of LTE for each pixel:

$$\begin{aligned} C &= \bigcup_i^W \bigcup_j^H L_{i,j}\left(K^{-1}\cdot x \rightarrow p_0\right) \\ &= \bigcup_i^W \bigcup_j^H L_{i,j}\left(p_1 \rightarrow p_0\right), \end{aligned} \tag{1}$$

where $L_{i,j}$ is total radiance at a pixel, and $p_1 = K^{-1}\cdot x$ is a hitpoint corresponds to $L_{i,j}$.

To intuitively exploit relationship between C and noise-free hitpoint $x_{i,j}$, we select arbitrary pixel and its total radiance in (Eqn. 1) and expand radiance sum over path segments $\bar{\mathrm{p}}_n = \mathrm{p}_0\mathrm{p}_1 ... \mathrm{p}_n$ with $n+1$ vertices:

$$L_{i,j}\left(p_1 \rightarrow p_0\right) = P\left(\bar{\mathrm{p}}_1\right) + P\left(\bar{\mathrm{p}}_2\right) + \sum_{n=3}^{\infty} P\left(\bar{\mathrm{p}}_n\right), \tag{2}$$

Let us consider direct lighting term in (Eqn. 2) first:

$$\begin{aligned} P\left(\bar{\mathrm{p}}_2\right) &= \int_A f\left(\mathrm{p}_2 \rightarrow \mathrm{p}_1 \rightarrow \mathrm{p}_0\right)\cdot L_{e(i,j)}\left(\mathrm{p}_2 \rightarrow \mathrm{p}_1\right)\cdot G\left(\mathrm{p}_1 \leftrightarrow \mathrm{p}_2\right) \\ &= \int_A \Bigg[f\left(\mathrm{p}_2 \rightarrow \mathrm{p}_1 \rightarrow \mathrm{p}_0\right)\cdot L_{e(i,j)}\left(\mathrm{p}_2 \rightarrow \mathrm{p}_1\right) \\ &\qquad \cdot V\left(\mathrm{p}_1 \leftrightarrow \mathrm{p}_2\right)\cdot \frac{|\cos(\theta_1)||\cos(\theta_2)|}{||\mathrm{p}_1 - \mathrm{p}_2||^2}\Bigg] \end{aligned} \tag{3}$$

Note that we omitted path differential $dA(\mathrm{p}_2)$ for brevity.

Since we assumed Lambertian surface and emitted radiance term is free up to any geometric transformation, the term that affects to direct lighting value in (Eqn. 3) is **geometry term**: visilbility, incident angles, and distance between hitpoints. **PJ: TODO: parameterize hitpoint as a sum of barycentric coords of vertex positions within a face.**

We will drop any consideration of remaining terms in (Eqn. 2). We do not consider emitted radiance i.e., albedo term $P\left(\bar{\mathrm{p}}_1\right)$ since this represents pure physical quantity of a geometry have, thus itself is consistent up to any geometric displacement (e.g., additive noise on vertices). **PJ: Does it sound clear?** We additionally treat indirect lighting term $\sum_{n=3}^{\infty} P(\bar{\mathrm{p}}_n)$ as it is; we will consider that indirect lighting cannot make a pixel value strongly deviated with neighbors where is originally have similar value within neighbors when only direct lighting is applied. This seems feasible because (1) indirect lighting term uses identical rendering equation with direct lighting term except to hitpoints (2) its throughput is far smaller than direct lighting term, under Lambertian assumption (3) it is applied globally; to make a pixel saturated against neighbors indirect lighting have to be applied to only that pixel (4) in a photorealistic rendering domain, multiple bounces among diffuse material causes significant noise, hence often ignored[Moon et al. 2013].

Therefore, we conclude that radiance at a pixel $L_{i,j}(p_1 \rightarrow p_2)$ is mostly affected by **geometry term** in $P(\bar{\mathrm{p}}_2)$ i.e.,

$$\begin{aligned} L_{i,j}(p_1 \rightarrow p_2) \sim P(\bar{\mathrm{p}}_2) &\propto G(\mathrm{p}_1 \leftrightarrow \mathrm{p}_2) \\ &= V(\mathrm{p}_1 \leftrightarrow \mathrm{p}_2)\cdot \frac{|\cos(\theta_1)||\cos(\theta_2)|}{||\mathrm{p}_1 - \mathrm{p}_2||^2} \end{aligned} \tag{4}$$

We now examine direct lighting term in rendering C, where noisy geometry $\mathcal{M}$ is used.

$$P\left(\bar{\mathrm{p}}_2'\right) = c\int_A L_{e(i,j)}\left(\mathrm{p}_2' \rightarrow \mathrm{p}_1'\right)\cdot V\left(\mathrm{p}_1' \leftrightarrow \mathrm{p}_2'\right)\cdot \frac{|\cos(\theta_1)'|\,|\cos(\theta_2)'|}{\left\|\mathrm{p}_1' - \mathrm{p}_2'\right\|^2}$$

As $\mathcal{M}$ is composed with noisy vertices V, **geometry term** is highly likely to have different value against those from C. Thus, there may exist different radiance values between C and C though evaluated at an identical pixel position.

This naturally concludes that a loss minimizing radiance difference equals to matching **geometry term** in C similar with C.

However, simply minimizing color difference i.e., $||C - C||^2$ does not work, as the texture of $\mathcal{M} \sim C$ contains real light information, furthermore we will cast a virtual light to saturate noisy vertices.

Therefore, C have far different tone compared to C in a global manner. The main challenge is to detect geometric differences between images in a local manner, regardless of any radiance tone changes.

3.3 Detecting Geometric Information given Color Image and Its Rendering

In this section, we describe our method to detect geometric information in both C and C, based on our intuition in (Eqn. 4).

We showed that changes of value made in a pixel is proportional to geometric transformation of a corresponding hitpoint. Consider a pair of hitpoint and corresponding pixel, together with neighboring pixels and corresponding hitpoints lie on same plane. If pixel values are evaluated in C, they will have similar radiance values according to assumption. This can be expressed as:

$$|L(K^{-1} \cdot x_{i,j}) - L(K^{-1} \cdot x_{i+\delta_i, j+\delta_j})| < \epsilon, \tag{5}$$

where $\delta_i \neq 0, \delta_j \neq 0$ is pixel coordinate displacement indicating neighbor of a pixel.

Based on our intuition, we decided to apply image gradient kernel to detect geometric displacement. This is natural as image gradient saturates where pixel values are discontinuous along with a pixel and its neighbors. From our assumption and intuition, this directly indicates that saturated region equals to the position where is geometrically discontinuous. If we apply image gradient kernel $G(\cdot)$ to C, $G(C)$ have zero values where satisfies (Eqn. 5), and vice versa.

We simply detect noise-free surfaces within C by applying Scharr gradient kernel. In order to ensure robustness on strong texture changes, one may adapt gradient from $\mathcal{D}$; note that for this report we only experimented with gradients from C. Specifically, Scharr kernel for each axis over an image is defined as

$$G_x = \begin{pmatrix} -3 & 0 & 3 \\ -10 & 0 & 10 \\ -3 & 0 & 3 \end{pmatrix}, G_y = \begin{pmatrix} -3 & -10 & -3 \\ 0 & 0 & 0 \\ 3 & 10 & 3 \end{pmatrix}, \tag{6}$$

and our detected geometric changes over Ground Truth color image $\widetilde{G}_C$ 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

$$\widetilde{G}_C = \frac{1}{2}\left(|G_x(I_C)| + |G_y(I_C)|\right) \tag{7}$$

In order to detect noisy vertices on C regardless of shading method, we propose Lightweight map to determine which vertex has stronger noise compare to $\widetilde{G}_C$. 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

$$I_{\text{lw}} = \{I_{\text{lw,i}} \in \mathbb{R}^{W*H}\}, I_{\text{lw,i}} = \frac{(x_i - p_0) \cdot n_i}{d(x_i, p_0) + \epsilon}, \tag{8}$$

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

$$\widetilde{G}_{\text{lw}} = \frac{1}{2}\left(|G_x(I_{\text{lw}})| + |G_y(I_{\text{lw}})|\right) \tag{9}$$

We found that using geometric normal to calculate lightweight map saturate pixels around noisy vertices more than shading normal, as shading normal smooths normal of each hitpoint 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}.

3.4 Optimization using Differentiable Rendering

We define lightweight loss to minimize the difference between noise-free geometric information $\widetilde{G}_C$ and noise-detected rendered image $\widetilde{G}_C$. We found that there is gradient value range inconsistency between $\widetilde{G}_C$ and $\widetilde{G}_C$, 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 zero value. Our final geometric gradients are:

$$G_C = \tanh\left(\widetilde{G}_{C\oplus\mathcal{D}}\right), G_{lw} = \tanh\left(\widetilde{G}_{lw}\right), \tag{10}$$

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:

$$\begin{aligned} \mathcal{L} &= w_{lw} \cdot L_{lw} + w_{pos} \cdot L_{pos}, \\ L_{lw} &= ||G_C - G_C||_2^2, \\ L_{pos} &= ||V - (V_o)||_2^2 = ||V_d||_2^2 \end{aligned} \tag{11}$$

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. For $\mathcal{M}$, we pre-computed from popular off-line

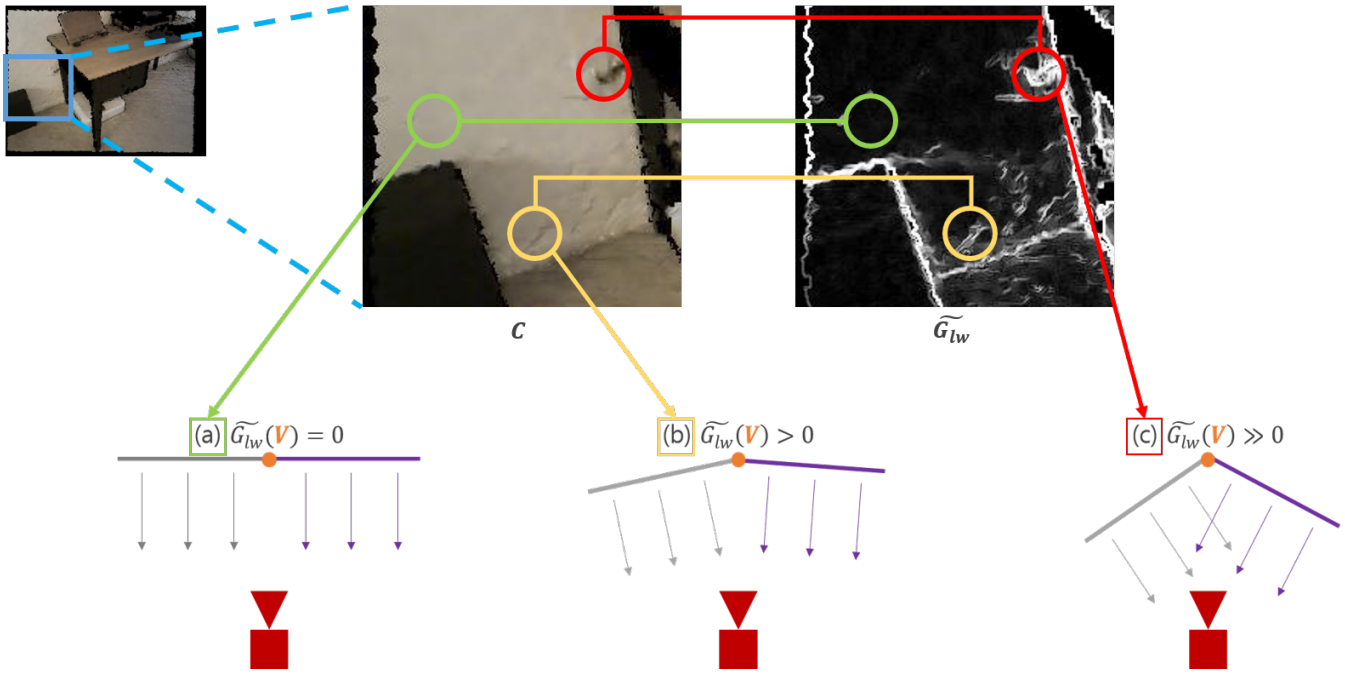

Fig. 3. Relationship between G_{lw} and actual noise of vertex V. We manually selected three pixels, where each have different magnitude of G_{lw}. We also illustrated 1D example of the relationship. **Gray** and **Purple** lines and arrows indicate each face and its geometric normal. **(a)** V is considered 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.

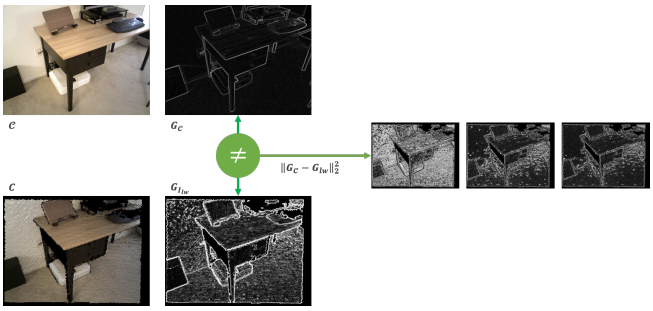

Fig. 4. Optimization procedure of our differentiable rendering. From generated target noise-free geometric clue G_C and input noisy geometric clue G_{lw}, we minimize L_2 distance between two clues. Note that we additionally penalize aggressive vertex evolve, however it is skipped in the figure.

Fig. 5. 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.

SLAM framework [Zhou et al. 2018] with voxel size 2cm. 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 Validation of Proposed Method

Validation of loss. In (Fig. 5), 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. We showed the effect of positional loss in (Fig. 6).

Validation with various scenes. We validated performance of our proposed method, by evaluating minimum eigenvalue of the region where we are able to perceive as 'flat', given C.

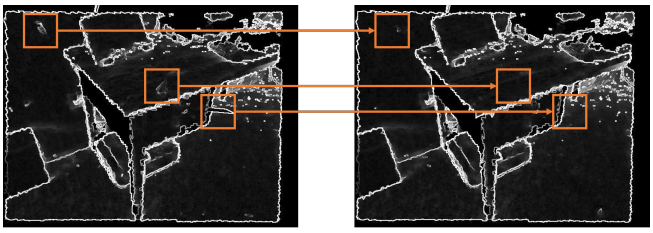

Fig. 6. Ablation study of positional loss, without loss and with loss, respectively. **Orange** inset highlights difference of vertex evolving aspect. Note that total loss of both result is converged. (Left) Some vertices are evolved in a wrong direction with respect to its perceived shape (Right) Our result with positional loss successfully maintain its shape, as the loss prevent vertices evolve not too far from its original position.

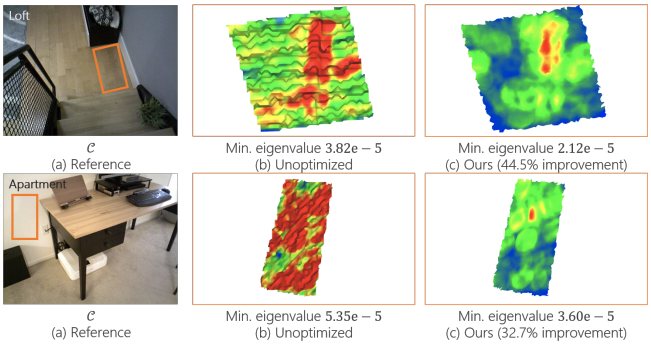

Fig. 7. Plane fitting validation of our result. Given scenes, we selected **Orange** region where is considered as "perceptually flat". We compared minimum eigenvalue between inset of input mesh (Middle) and optimized mesh (Right), and plotted heatmap of each result. Our method successfully minimize minimum eigenvalue within plane.

4.2 Comparison with Previous Work

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

4.3 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 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 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 $G_{\mathcal{D}}$, 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.

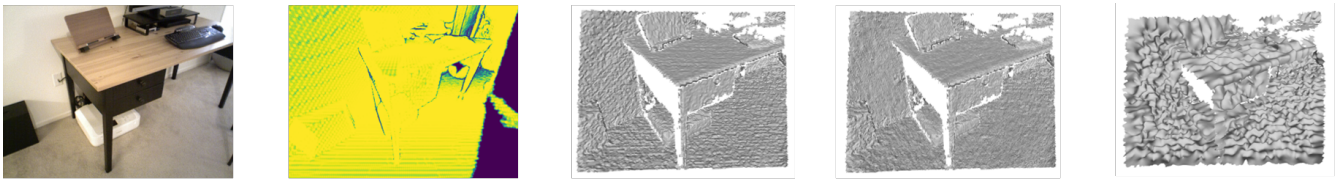

Fig. 8. Comparison of result using previous method and our method. For target silhouette, we generated it from input mesh by render mesh with silhouette renderer, and it is enough to use since silhouette image itself does not hold any depth information other than boundary information of target mesh. **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.

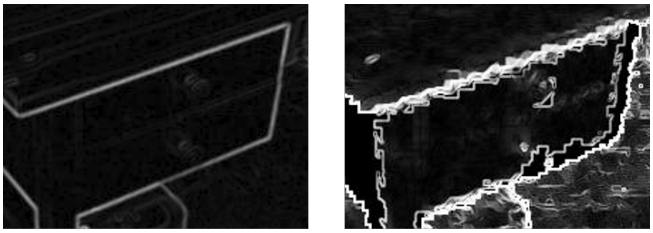

Fig. 9. Inset of each gradient. G_{I_W} is corrupted since geometric normal deviates around edge.

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.

Krishna Murthy Jatavallabhula, Edward Smith, Jean-Francois 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, JongHyeob 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.

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).

Richard Szeliski. 2010. *Computer vision: algorithms and applications*. Springer Science & Business Media.

Eric Veach. 1998. *Robust Monte Carlo methods for light transport simulation*. Stanford University.

Thomas Whelan, Michael Goesele, Steven J Lovegrove, Julian Straub, Simon Green, Richard Szeliski, Steven Butterfield, Shobhit Verma, Richard A Newcombe, M Goesele, et al. 2018. Reconstructing scenes with mirror and glass surfaces. *ACM Trans. Graph.* 37, 4 (2018), 102–1.

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).

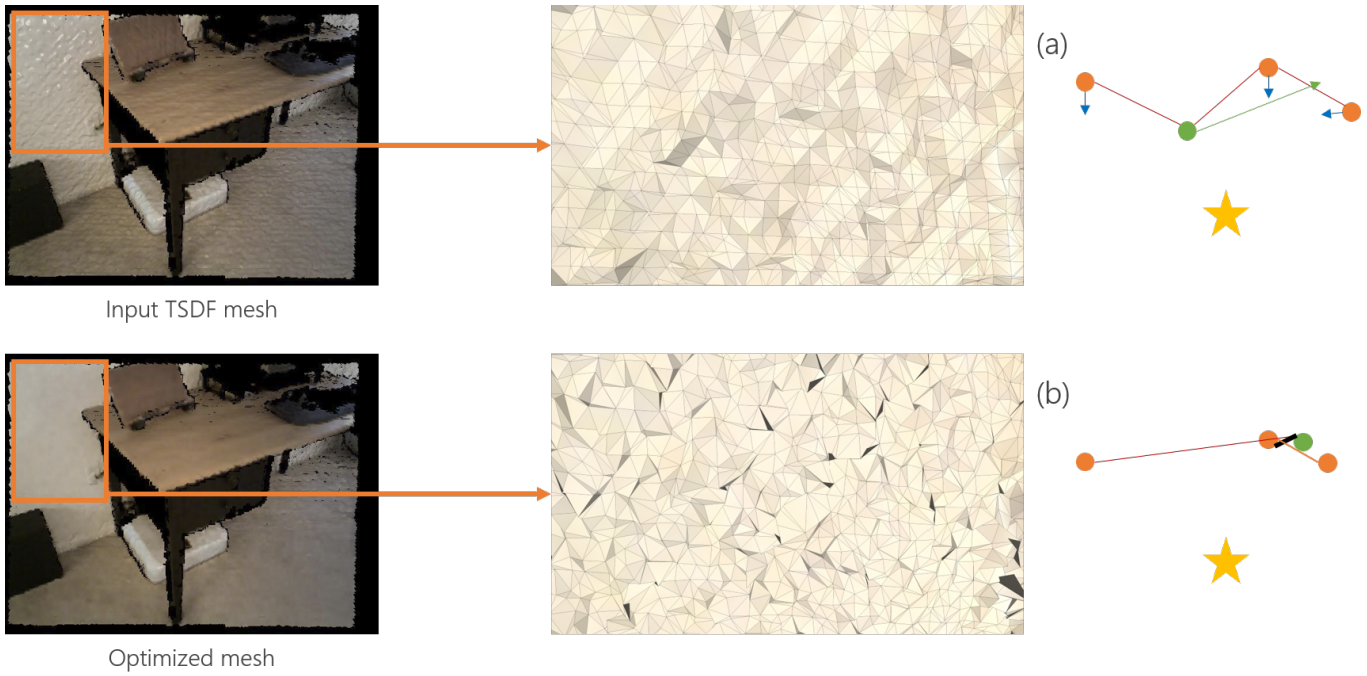

Fig. 10. Comparison of input mesh and optimized mesh. We zoomed up each geometry corresponds to image inset. **Black artifacts** are observed in zoomed view in optimized mesh. We found that those are self-intersection between neighboring faces, since our positional constraint L_{pos} does not consider 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.