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

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

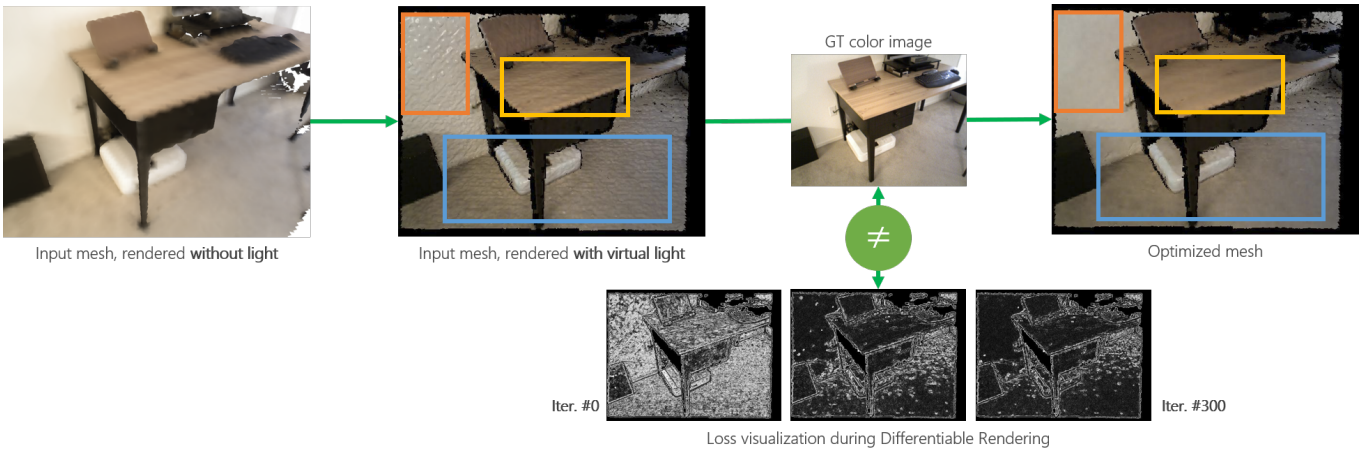

Fig. 1. Overview of our method. (From top left) We saturate noisy vertices by rendering input TSDF mesh with virtually placed light source. We extract common-shared geometric clue between rendered input mesh and target color image. Differentiable renderer iteratively minimizes loss, which is the difference 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-RWOIxocYoxG2QOkJc7YWlF/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.

0730-0301/2021/6-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. Single-view TSDF Mesh Noise Reduction under Virtual Light using Differentiable Rendering. *ACM Trans. Graph.* 1, 1 (June 2021), 6 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 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 propagates 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 Figure. 2.

Our main contribution is bridging the gap between perceptional 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 [2][3] flashy photography is used to enhance images taken from scene which has insufficient lighting condition. [4] 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 Figure 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 [5], estimating true depth from noisy measurements has been one of the major challenges in SLAM. Basically, [5] 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 [6], merging depth information of neighboring frames either offline [7][8], or online [9][10]. 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 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

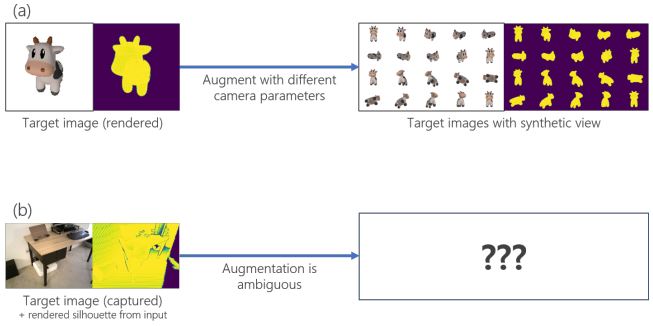

Fig. 2. Different setup between optimizing simple mesh and TSDF mesh from SLAM. Previous works aimed to optimize input geometry to set of target images captured with different camera view either synthetically[11][12][13][14] or by taking calibrated real photographs[14]. However, in the case of TSDF mesh 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.

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. Figure 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

Assumptions. Our goal is to minimize TSDF mesh noise fused from a single, Lambertian-dominant indoor RGB-D frame. Given the scenario, we assume that we know intrinsic, extrinsic parameters of camera used to capture the frame. Specifically, we pre-computed input TSDF mesh $\mathcal{M}$ from pair of GT color image C and depth image $\mathcal{D}$ with known intrinsic K, such that $\mathcal{M} = TSDF(K^{-}1(C \oplus \mathcal{D}))$. Here, $\oplus$ denotes image registration operator. We used popular SLAM framework [8] to generate $\mathcal{M}$ with voxel size as 2cm. Since $\mathcal{D}$ is noisy measurement of real scene geometry, $\mathcal{M}$, and its rendering C have slightly different appearance compare with C. We additionally constraint that there are no strong texture changes under geometrically close, flat surfaces (e.g., chessboard pattern on flat wall). This assumption is used to ensure that strong radiance change is only made from interaction between light and hit-point geometry within C. We will elaborate this assumption later.

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, 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 optimized vertices approximates

noise-free geometry similar with C. The main problem is how find common geometric representations across different image types (here, rendered $\mathcal{M}$ fused from $\mathcal{D}$ and C) to figure out which region is to be optimized (i.e., noisy).

PJ: Too verbose! TODO: add proof Our key observation is strong radiance change within 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. We are able to guarantee the first assumption since SLAM dataset is generated under the assumption. It is still open challenge as it is necessary condition for correspondences used to estimating relative camera pose, as well as current depth sensing hardware fails to measure depth values lie on non-Lambertian surfaces. By applying second assumption, we are able to intuitively know that there is no sudden radiance change around point on noise-free 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.

3.1 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

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

, 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

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

In order to detect noisy vertices over TSDF mesh, 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

$$I_{\text{lw}} = \{I_{\text{lw,i}} \in \mathbb{R}^{W*H}\}, I_{\text{lw,i}} = \frac{(x_i - p_0)\cdot n_i}{d\left(x_i, p_0\right) + \epsilon}, \tag{3}$$

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 $d\left(x_i, p_0\right)$ is Euclidean distance between hitpoint and light position. We obtain changing amount of each lightweight value $\widetilde{G}_{\text{lw}}$ by applying Scharr kernel over I_{lw}, similar with $\widetilde{G}_C$.

$$\widetilde{G}_{\text{lw}} = \frac{1}{2}\left(|G_x\left(I_{\text{lw}}\right)| + |G_y\left(I_{\text{lw}}\right)|\right) \tag{4}$$

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}.

3.2 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:

$$G_C = \tanh\left(\widetilde{G}_{C\oplus\mathcal{D}}\right), G_{lw} = \tanh\left(\widetilde{G}_{lw}\right), \tag{5}$$

where $\tanh\left(x\right) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$. Our optimizer minimizes lightweight loss representing geometric difference, while penalizing vertices not to

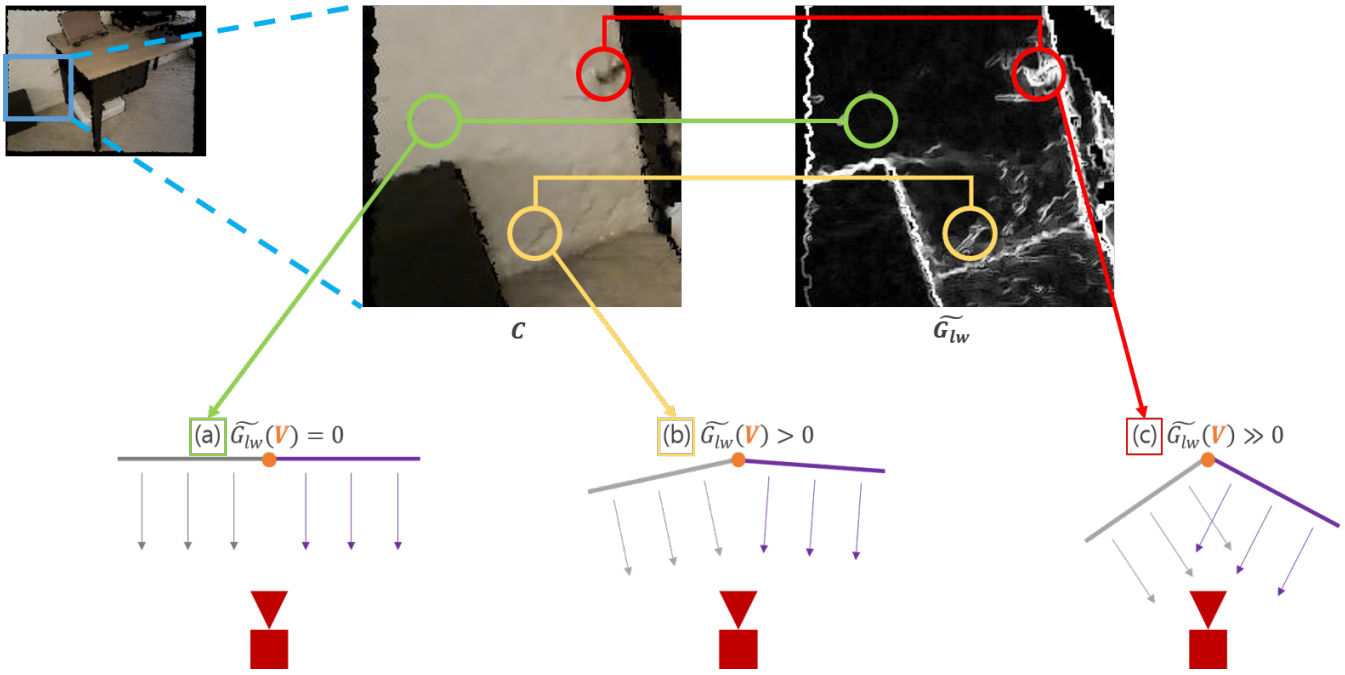

Fig. 3. Relationship between G_{lw} and actual noise of vertex V. We manually selected three pixels, where each have different magnitude of G_{lw}. We also illustrated 1D example of the relationship. Gray and Purple lines and arrows indicate each face and its geometric normal. (a) V is considered as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

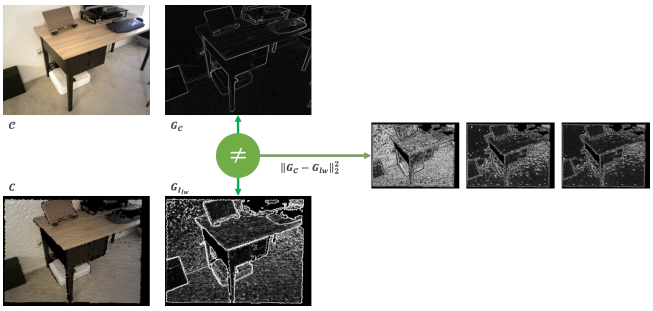

Fig. 4. Optimization procedure of our differentiable rendering. From generated target noise-free geometric clue G_C and input noisy geometric clue G_{lw}, we minimize L_2 distance between two clues. Note that we additionally penalize aggressive vertex evolve, however it is skipped in the figure.

evolve too far from its initial position:

$$\mathcal{L} = w_{lw} \cdot L_{lw} + w_{pos} \cdot L_{pos}, \tag{6}$$

$$L_{lw} = ||G_C - G_C||_2^2, \tag{7}$$

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

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[11]. 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[1]. We slightly modified its workflow in order to evaluate performance under identical conditions (please refer Figure 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[11], which is defined as:

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

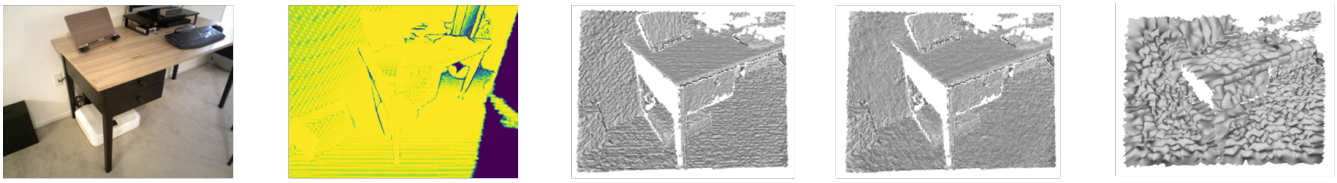

Fig. 5. Comparison of result using previous method and our method. For target silhouette, we generated it from input mesh by render mesh with silhouette renderer, and it is enough to use since silhouette image itself does not hold any depth information other than boundary information of target mesh. **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.

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: vaildate 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 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

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.

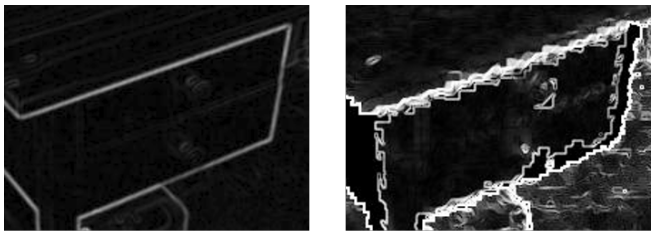

Fig. 7. Inset of each gradient. G_{lw} is corrupted since geometric normal deviates around edge.

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

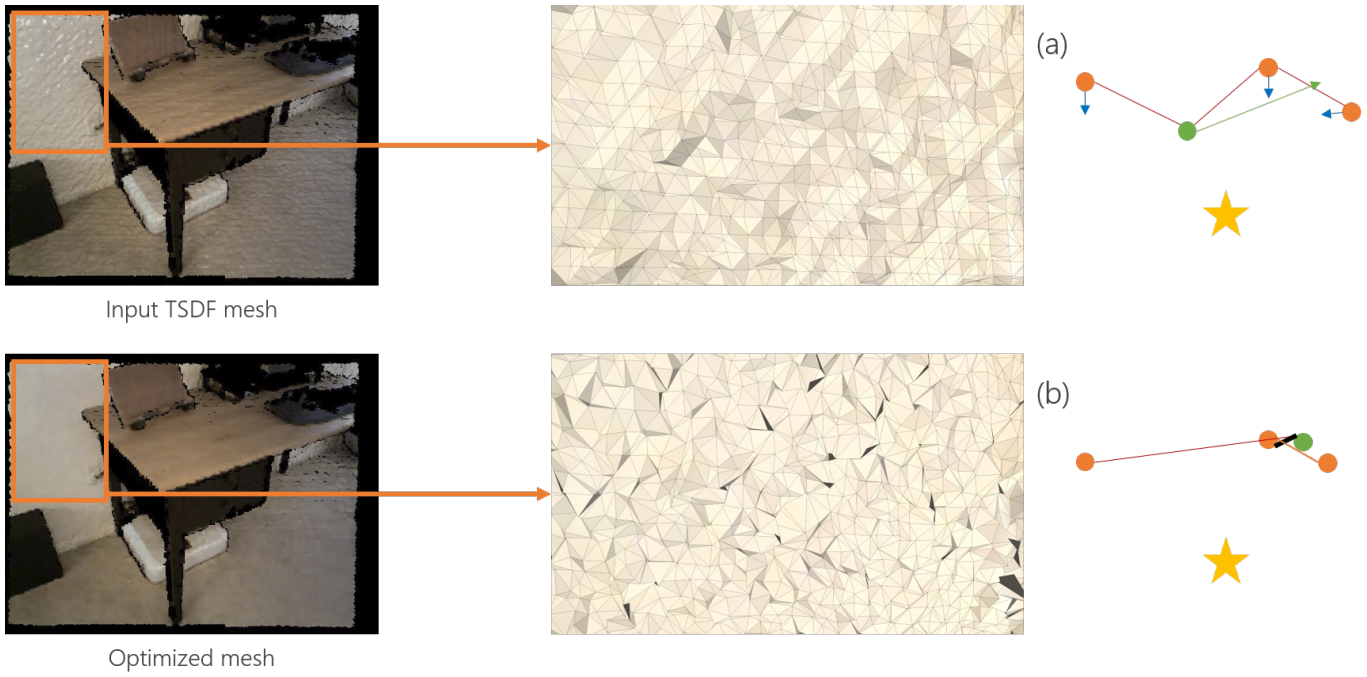

Fig. 8. Comparison of input mesh and optimized mesh. We zoomed up each geometry corresponds to image inset. Black artifacts are observed in zoomed view in optimized mesh. We found that those are self-intersection between neighboring faces, since our positional constraint L_{pos} does not consider distances 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.

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_{l_w} 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.