Statistical Gradient Filtering for Geometry Optimization Under Limited Observations
- Supplementary Material -

WONJONG JANG, POSTECH, South Korea
GWANGJIN JU, POSTECH, South Korea
SEUNGYONG LEE, POSTECH, South Korea

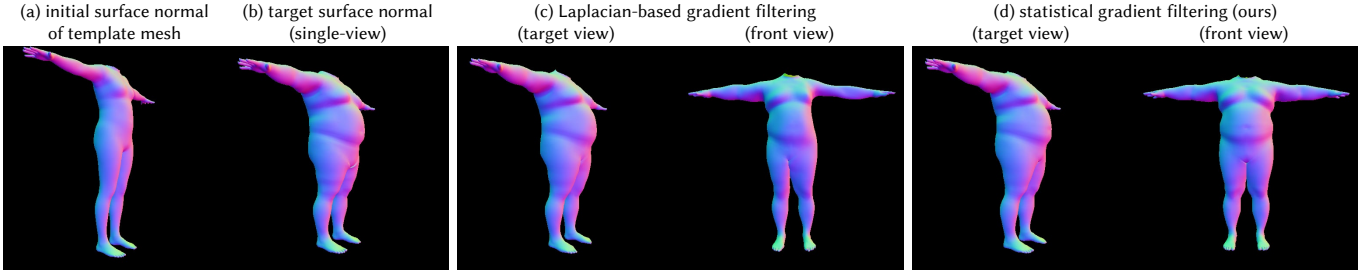

Fig. 1. Inverse rendering of full-body geometry. Laplacian-based gradient filtering [Nicolet et al. 2021] fits surface normals well for the target view, but relying solely on spatial smoothness leads to distorted geometry in regions that are unobserved from the target view. In contrast, our method incorporates statistical shape priors into gradient filtering, achieving accurate fitting for the target view while producing natural shapes in the unobserved regions.

1 Derivations

1.1 Algebraic Structure Preservation of Matrix Inversion

In the main paper, we claim that since $\mathbf{U}_k\mathbf{U}_k^T$ is idempotent, the inverse of a linear combination of $\mathbf{I}$ and $\mathbf{U}_k\mathbf{U}_k^T$ preserves the algebraic structure. Here we justify this claim.

Proposition 1.1. *Let $\mathbf{H} = \alpha\mathbf{I} + \beta\mathbf{P}$, where $\mathbf{P}$ is idempotent. If $\alpha \neq 0$ and $\alpha + \beta \neq 0$, then $\mathbf{H}^{-1}$ has the form $\mu\mathbf{I} + \nu\mathbf{P}$.*

Proof. Since $\mathbf{P}$ is idempotent, $\mathbf{I} - \mathbf{P}$ is also idempotent, and these two matrices are orthogonal: $\mathbf{P}(\mathbf{I} - \mathbf{P}) = \mathbf{0}$. We can decompose:

$$\mathbf{H} = \alpha(\mathbf{I} - \mathbf{P}) + (\alpha + \beta)\mathbf{P}. \tag{1}$$

Since $\mathbf{P}$ and $\mathbf{I} - \mathbf{P}$ project onto complementary subspaces, inversion acts independently on each:

$$\mathbf{H}^{-1} = \frac{1}{\alpha}(\mathbf{I} - \mathbf{P}) + \frac{1}{\alpha + \beta}\mathbf{P} = \frac{1}{\alpha}\mathbf{I} + (\frac{1}{\alpha + \beta} - \frac{1}{\alpha})\mathbf{P}, \tag{2}$$

which is a linear combination of $\mathbf{I}$ and $\mathbf{P}$. □

1.2 Equivalence of Exponent p and β Rescaling

We prove that adjusting the exponent p in the filtered update (Eq. 14 in our main paper) is equivalent to rescaling the hyperparameter β.

Proposition 1.2. *Applying exponent p with parameter β is equivalent to omitting the exponent p and using rescaled parameter $\beta' = (1 + \beta)^p - 1$.*

Proof. Let $\alpha = \frac{\beta}{1+\beta} \in [0, 1)$. The statistical gradient filter can be written as:

$$\mathbf{P}_{\text{stat}} = \alpha\mathbf{U}_k\mathbf{U}_k^T + (1 - \alpha)\mathbf{I}, \tag{3}$$

which can be rewritten as:

$$\mathbf{P}_{\text{stat}} = \mathbf{U}_k\mathbf{U}_k^T + (1 - \alpha)(\mathbf{I} - \mathbf{U}_k\mathbf{U}_k^T). \tag{4}$$

Since $\mathbf{U}_k\mathbf{U}_k^T$ and $\mathbf{I}-\mathbf{U}_k\mathbf{U}_k^T$ are orthogonal projectors onto complementary subspaces (i.e., their product is zero and each is idempotent), raising to the power p gives:

$$\mathbf{P}_{\text{stat}}^p = \mathbf{U}_k\mathbf{U}_k^T + (1 - \alpha)^p(\mathbf{I} - \mathbf{U}_k\mathbf{U}_k^T). \tag{5}$$

Expanding:

$$\mathbf{P}_{\text{stat}}^p = [1 - (1 - \alpha)^p]\mathbf{U}_k\mathbf{U}_k^T + (1 - \alpha)^p\mathbf{I}. \tag{6}$$

This has the same form as Eq. 3 with $\alpha' = 1 - (1 - \alpha)^p$. Converting back to β:

$$\beta' = \frac{\alpha'}{1 - \alpha'} = (1 + \beta)^p - 1. \tag{7}$$

□

2 Implementation Details

We implemented our method in PyTorch with CUDA acceleration. All experiments and performance evaluations were conducted on a desktop workstation equipped with an AMD Ryzen 7950X3D processor, an NVIDIA GeForce RTX 5090 GPU, and 192GB RAM. Throughout all experiments in our main paper, we use the mean shape of the FLAME face model [Li et al. 2017] as our template mesh, which consists of 5,023 vertices and 9,976 triangles.

We used UniformAdam [Nicolet et al. 2021] as the optimizer with step size 0.01, $\beta_1 = 0.9$, and $\beta_2 = 0.999$. For our statistical gradient filtering, we set $\beta = 30$. For regularization, we set $\beta = 10$. For Laplacian-based gradient filtering baselines [Jung et al. 2023; Nicolet

Authors' Contact Information: Wonjong Jang, POSTECH, South Korea, wonjong@postech.ac.kr; Gwangjin Ju, POSTECH, South Korea, gwangjin@postech.ac.kr; Seungyong Lee, POSTECH, South Korea, leesy@postech.ac.kr.

2025. ACM 1557-7368/2025/8-ART
https://doi.org/10.1145/3731427

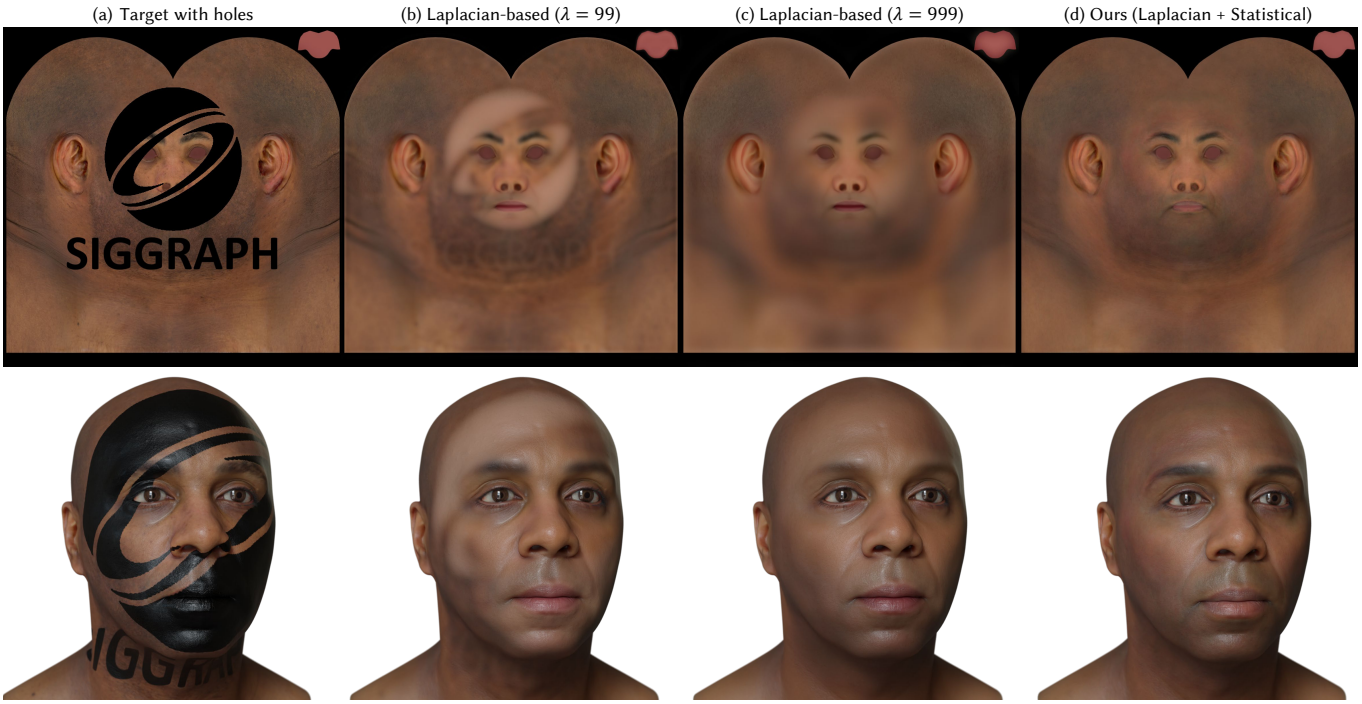

Fig. 2. Texture map inpainting. (a) Target texture map with large holes. (b) Laplacian-based gradient filtering [Nicolet et al. 2021] with small $\lambda = 99$ fails to inpaint large holes. (c) With a larger $\lambda = 999$, the method propagates surrounding color changes over large holes but produces blurred results. (d) Our method seamlessly fills large holes and synthesizes natural features such as eyebrows.

et al. 2021], we used $\lambda = 19$. These hyperparameters were fixed across all experiments. For the PCA-based method, we used a step size of 5.0, as it optimizes coefficients in the parameter space rather than directly updating vertex positions. For non-rigid registration on partial scans in Sec. 5.1 of the main paper, each partial scan contains on average 4,335 observed points used as optimization targets. For inverse rendering in Sec. 5.2 of the main paper, in the case of synthetic data, we perform initial rigid alignment to fit the template mesh to the target surface normal map before optimization. For in-the-wild images, we initialize the template mesh with PCA coefficients obtained by using Pixel3DMM [Giebenhain et al. 2025]. On average, 122,644 pixels per image are observed and contribute to the rendering loss.

Computational efficiency. Our statistical gradient filtering offers a computational advantage over Laplacian-based gradient filtering [Nicolet et al. 2021]. Laplacian-based gradient filtering requires solving a linear system involving $(\mathbf{I} + \lambda\mathbf{L})$ at each iteration. While this can be efficiently done via pre-factorization [Chen et al. 2008], our approach requires only matrix-vector multiplications, eliminating the need for a linear solver. Specifically, using matrix-vector multiplications instead of linear solves enables better parallel acceleration on the GPU than sequential sparse substitutions of a pre-factorized linear solve. Empirically, for a $1{,}024 \times 1{,}024$ grid (2^{20} vertices), Laplacian-based gradient filtering takes 20.53 ms per filtering step, whereas our method takes only 4.53 ms.

Convergence comparison. We compare the convergence behavior of parametric, regularization, Laplacian-based gradient filtering, and our method on the FLAME face model. As this mesh contains a relatively small number of vertices, the gaps between the overall runtime of all baselines and ours are less than one second. Nonetheless, our method converges to a lower error than the baselines. Moreover, to reach a given error, our method requires fewer iterations (Fig. 4 of our main paper), demonstrating that our filtering provides a better descent direction.

Wall-clock performance. We run 500 iterations for non-rigid template mesh registration and 3000 iterations for inverse rendering with all methods completing in approximately 20 seconds.

3 Additional Experiments

We provide additional experiments to validate our method on diverse optimization tasks beyond those presented in the main paper: inverse rendering for reconstructing full-body geometry from a single-view surface normal map, texture map inpainting for completing partial textures, and Gaussian Splatting optimization on meshes for novel view synthesis. The first two experiments demonstrate that our method is compatible with other PCA models than the FLAME face model [Li et al. 2017]: for full-body geometry, we use a PCA model computed from SMPL-X (400 bases) [Pavlakos et al. 2019]; for texture maps, we construct a PCA model (200 bases) from 1760 texture maps, obtained by augmenting 176 texture maps from 3DScanStore [3DScanStore 2025] with color jittering to simulate

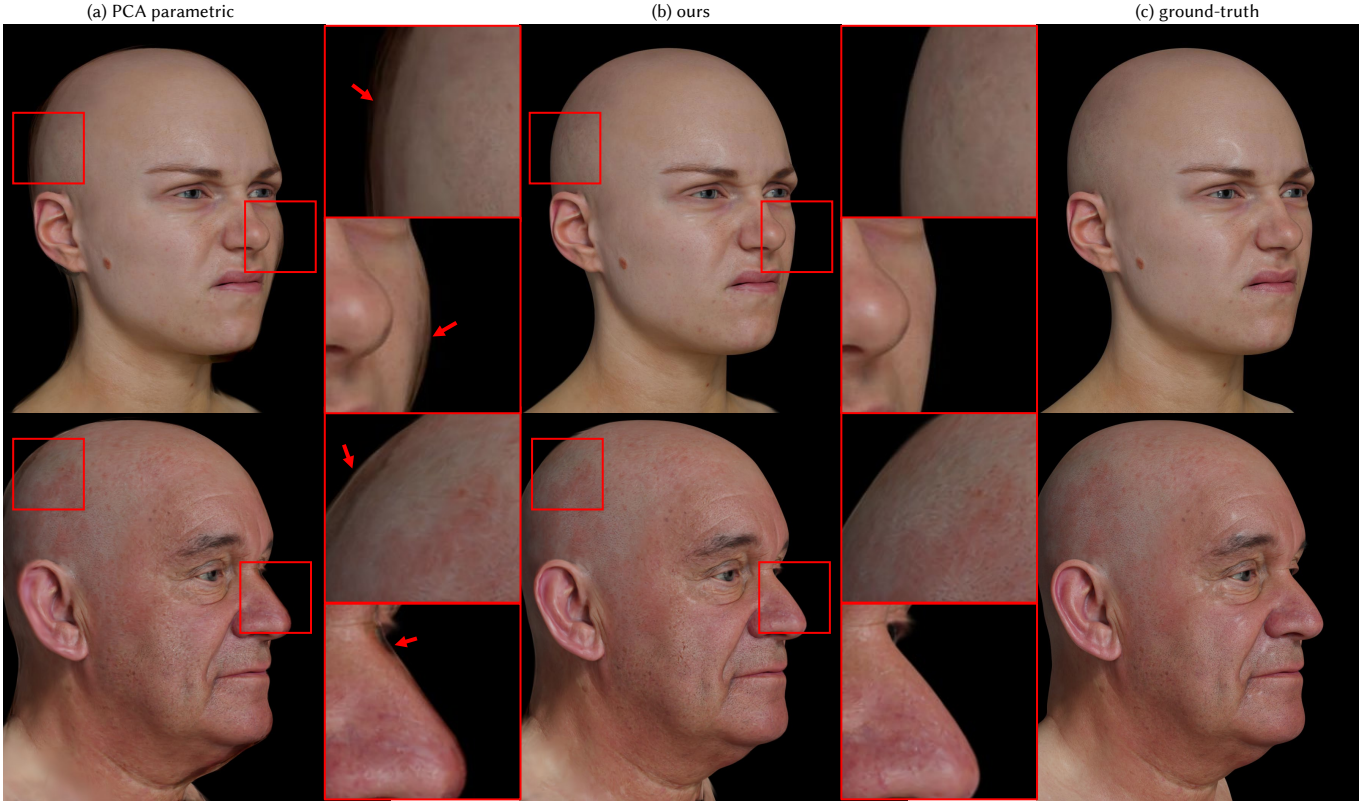

Fig. 3. Gaussian splatting optimization on 3D head meshes. PCA parameterization causes ghosting artifacts due to its limited solution space, which prevents accurate geometry fitting. Our method enables accurate geometry fitting, producing higher-quality renderings without such artifacts.

diverse skin tones. For Gaussian splatting optimization, we utilize the FLAME face model.

3.1 Inverse Rendering of Full-Body Geometry

We extend the inverse rendering experiment in Sec. 5.2 of the main paper to full-body mesh reconstruction. Given a single-view surface normal map of a target body, we deform the template body mesh by minimizing pixel-wise surface normal differences between rendered and target normal maps using a differentiable renderer [Laine et al. 2020] along with the same silhouette loss as in the main paper. As shown in Fig. 1, Laplacian-based gradient filtering [Nicolet et al. 2021] fits the target view well but yields distorted geometry in regions that are unobserved from the target view. In contrast, our method produces natural shapes in the unobserved regions despite using only a single-view observation.

3.2 Texture Map Inpainting

Given a partial texture image, we optimize the texture by minimizing the RGB loss over the visible region:

$$\mathcal{L}_{\text{RGB}} = \sum_{i \in \mathcal{V}} \|\mathbf{T}_i - \mathbf{T}_i^{\text{gt}}\|_1, \tag{8}$$

where $\mathcal{V}$ denotes the set of visible pixels, $\mathbf{T}_i$ is the optimized texture value at pixel i, and $\mathbf{T}_i^{\text{gt}}$ is the ground truth. The goal is to leverage gradient filtering to fill the masked regions with plausible content.

Since this task requires careful handling of seams, we employ a hybrid formulation that replaces the identity matrix $\mathbf{I}$ in the statistical gradient filtering with the Laplacian-based diffusion operator $(\mathbf{I} + \lambda \mathbf{L})^{-1}$, combining statistical priors with spatial smoothness.

As shown in Fig. 2, using the Laplacian-based operator alone [Nicolet et al. 2021] enforces only spatial smoothness, which is insufficient for large holes with small λ, while large λ leads to blurred results. Moreover, as seen in the eyebrow region, Laplacian-based gradient filtering merely diffuses surrounding color changes without synthesizing new features. In contrast, incorporating our statistical gradient filter not only fills large holes naturally but also synthesizes coherent features, such as plausible eyebrows, that harmonize with the surrounding context.

3.3 Gaussian Splatting Optimization on 3D Meshes

Recent Gaussian head avatars [He et al. 2025; Li et al. 2025; Luchuan Song 2025; Qin et al. 2024; Shao et al. 2024; Zhang et al. 2025; Zielonka et al. 2025] utilize PCA models from FLAME [Li et al. 2017], binding Gaussians onto 3D full-head meshes to constrain geometry and capture facial expressions. As in TransGS [Qin et al. 2024], we formulate 2D Gaussians on the UV space of a 3D mesh as surfels, where

Gaussian positions are parameterized by the underlying mesh and vertex positions of the mesh are updated through the rendering loss of Gaussian splatting.

We compare our method against the PCA-based method that optimizes PCA coefficients for Gaussian splatting on 3D meshes. For training data, we render 90 multi-view images from 3DScanStore assets using Blender Cycles [Iraci 2013], and adopt the standard Gaussian splatting loss functions [Kerbl et al. 2023]. As shown in Fig. 3, PCA parameterization limits the solution space, preventing accurate geometry fitting and resulting in ghosting artifacts around facial boundaries. Our method allows greater flexibility in geometry deformation, enabling precise fitting to target views and producing higher quality renderings without blur artifacts.

References

3DScanStore. 2025. *3DScanStore*. https://www.3dscanstore.com

Yanqing Chen, Timothy A. Davis, William W. Hager, and Sivasankaran Rajamanickam. 2008. Algorithm 887: CHOLMOD, Supernodal Sparse Cholesky Factorization and Update/Downdate. *ACM Trans. Math. Softw.* 35, 3 (2008).

Simon Giebenhain, Tobias Kirschstein, Martin Rünz, Lourdes Agapito, and Matthias Nießner. 2025. Pixel3DMM: Versatile Screen-Space Priors for Single-Image 3D Face Reconstruction. *arXiv preprint arXiv:2505.00615* (2025).

Yisheng He, Xiaodong Gu, Xiaodan Ye, Chao Xu, Zhengyi Zhao, Yuan Dong, Weihao Yuan, Zilong Dong, and Liefeng Bo. 2025. LAM: Large Avatar Model for One-shot Animatable Gaussian Head. In *Proc. ACM SIGGRAPH*.

Bernardo Iraci. 2013. *Blender cycles: lighting and rendering cookbook*. Packt Publishing Ltd.

Yucheol Jung, Hyomin Kim, Gyeongha Hwang, Seung-Hwan Baek, and Seungyong Lee. 2023. Mesh density adaptation for template-based shape reconstruction. In *Proc. ACM SIGGRAPH*.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. 3D Gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.* 42, 4 (2023).

Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. 2020. Modular Primitives for High-Performance Differentiable Rendering. *ACM Trans. Graph.* 39, 6 (2020).

Linzhou Li, Yumeng Li, Yanlin Weng, Youyi Zheng, and Kun Zhou. 2025. RGBAvatar: Reduced Gaussian Blendshapes for Online Modeling of Head Avatars. In *Proc. CVPR*.

Tianye Li, Timo Bolkart, Michael J Black, Hao Li, and Javier Romero. 2017. Learning a model of facial shape and expression from 4D scans. *ACM Trans. Graph.* 36, 6 (2017), 194–1.

ZHan Xu Yi Zhou Deepali Aneja Chenliang Xu Luchuan Song, Yang Zhou. 2025. StreamME: Simplify 3D Gaussian Avatar within Live Stream. In *Proc. ACM SIGGRAPH*.

Baptiste Nicolet, Alec Jacobson, and Wenzel Jakob. 2021. Large steps in inverse rendering of geometry. *ACM Trans. Graph.* 40, 6 (2021).

Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed AA Osman, Dimitrios Tzionas, and Michael J Black. 2019. Expressive body capture: 3d hands, face, and body from a single image. In *Proc. CVPR*.

Dafei Qin, Hongyang Lin, Qixuan Zhang, Kaichun Qiao, Longwen Zhang, Zijun Zhao, Jun Saito, Jingyi Yu, Lan Xu, and Taku Komura. 2024. Instant Facial Gaussians Translator for Relightable and Interactable Facial Rendering. *arXiv preprint arXiv:2409.07441* (2024).

Zhijing Shao, Zhaolong Wang, Zhuang Li, Duotun Wang, Xiangru Lin, Yu Zhang, Mingming Fan, and Zeyu Wang. 2024. SplattingAvatar: Realistic Real-Time Human Avatars with Mesh-Embedded Gaussian Splatting. In *Proc. CVPR*.

Jiawei Zhang, Zijian Wu, Zhiyang Liang, Yicheng Gong, Dongfang Hu, Yao Yao, Xun Cao, and Hao Zhu. 2025. Fate: Full-head gaussian avatar with textural editing from monocular video. In *Proc. CVPR*.

Wojciech Zielonka, Stephan J. Garbin, Alexandros Lattas, George Kopanas, Paulo Gotardo, Thabo Beeler, Justus Thies, and Timo Bolkart. 2025. Synthetic Prior for Few-Shot Drivable Head Avatar Inversion. In *Proc. CVPR*.