Laboratório VISGRAF

Instituto de Matemática Pura e Aplicada

Depth-Supervised 2D Gaussian Splatting For Geometric Reconstruction

Daniel Perazzo, Tiago Novello (supervisor)

Technical Report TR-24-08 Relatório Técnico

August - 2024 - Agosto

DEPTH-SUPERVISED 2D GAUSSIAN SPLATTING FOR GEOMETRIC RECONSTRUCTION

DANIEL PERAZZO, TIAGO NOVELLO, JOÃO PAULO, LUIZ VELHO
IMPA

1. Introduction

Implicit neural representations (INRs) [11] have profoundly impacted the field of 3D reconstruction from images with the introduction of neural radiance fields (NeRFs) [8]. NeRFs implicitly use a neural network to parameterize a scene's fuzzy geometry (density) and radiance. Using volume rendering, NeRFs can synthesize highly detailed novel views of real-world scenes. However, directly extracting detailed surfaces from such fuzzy density representations poses challenges due to insufficient surface regularization. Surface-based variants such as NeuS [10] address this issue by representing the surface as the zero-level set of a signed distance function (SDF), enabling geometric regularization by enforcing the Eikonal equation. Despite these promises, NeRF methods are still limited by the underlying characteristics of volume rendering. These require expensive stochastic sampling at rendering time, making it infeasible to render in real-time at high resolutions.

These limitations motivated novel scene representations, particularly the Gaussian splatting (GS) technique [5], which has significantly impacted the field. GS enables a highly effective novel view synthesis by parameterizing the scene explicitly using Gaussians with color and density parameters. This enabled real-time differentiable rendering at high resolutions through classic rasterization-based approaches, introduced in works such as EWA Splatting [16]. Despite being an excellent asset for view interpolation, modeling the surface geometry has become more challenging, highlighting an open avenue for further research.

In this report, we explore the 2DGS representation. We couple 2DGS with depth supervision, Neural SDF training, and back-face culling-based sampling. This report aims to outline the modeling and describe our experiments.

Date: August 8, 2024
E-mail address: daniel.perazzo@impa.br.
Key words and phrases. Gaussian splatting, neural implicit representations.

2. Method

2.1. **Modelling**

In this work, we approach the problem of 3D scene reconstruction from posed images $\{\hat{I}_i\}_{i=1}^m$. We follow the approach in 2DGS [3] and represent the scene as a collection of 2D Gaussians endowed with view-dependent colors. We use the world-to-screen transformations W_i and compare the resulting image I_i with the ground truth $\hat{I}_i$ to optimize the model parameters. We also use monocular depth $\hat{\mathbf{D}}$ extracted from the images $\hat{I}_i$, using the state-of-the-art depth estimator Depth-anything-v2 [13]. Our final aim is to obtain a 3D object representation with accurate geometric properties.

We consider a collection of N 2D Gaussians $\mathcal{g}_j = \{\mu_j, \Sigma_j, c_j, \alpha_j\}$, where μ_j are the *mean* values representing scene points, $\Sigma_j = [N_j, u_j, v_j]$ are the 2D Gaussian orientations, N_j are Gaussian normals and α_j is the opacity for each Gaussian. The two vectors u_j, v_j give the Gaussian *spreads*. Finally, c_j represents the colors evaluated for each viewing direction using spherical harmonics. These parameters will be obtained during the optimization.

To perform the rasterization, given a pixel point $x \in \mathbb{R}^2$, we need to find its corresponding opacity $\mathcal{G}_j(x) \in \mathbb{R}$ for the "splatted" surfel. We use the previously defined $\mathcal{g}_j$ parameters in 2DGS [3] to find this value.

We use the standard Gaussian Splatting rasterization equation, that is, the color of a pixel $x \in \mathbb{R}^2$ is given by a function $I : \mathbb{R}^2 \to \mathbb{R}^3$ given by:

$$I(x) := \sum_{k=0}^{N} \mathcal{G}_k(x)\alpha_k c_k \prod_{j=0}^{k-1}(1 - \alpha_j \mathcal{G}_j(x)) \tag{2.1}$$

This results in a rendered image I_j. We use a photometric loss $\mathcal{L}_{img}$ given by:

$$\mathcal{L}_{img} = \sum_{j=0}^{M} \|\hat{I}_j - I_j\|_1 \tag{2.2}$$

And also a SSIM [1] loss $\mathcal{L}_{SSIM}$:

$$\mathcal{L}_{SSIM} = \sum_{j=0}^{M} \texttt{SSIM}(\hat{I}_j, I_j) \tag{2.3}$$

As in 2DGS[3], we use geometry regularizers. To smooth the normals, we use a *smooth normal loss*, which penalizes the rendered normal map $\mathbf{N}$, which is not coherent about the depth map $\mathbf{D}$. If we define the normal map $\mathbf{N}$ for each pixel $x \in \mathbb{R}^2$ as:

$$\mathbf{N}(x) := \sum_{k=0}^{N} \mathcal{G}_k(x)\alpha_k N_k \prod_{j=0}^{k-1}(1-\alpha_j \mathcal{G}_j(x)) \tag{2.4}$$

Analogously, if every surfel has a distance to the camera as z_k, then we can render a depth map $\mathbf{D}$ for each pixel $x \in \mathbb{R}^2$ as:

$$\mathbf{D}(x) := \sum_{k=0}^{N} \frac{\mathcal{G}_k(x)\alpha_k z_k \prod_{j=0}^{k-1}(1-\alpha_j \mathcal{G}_j(x))}{\sum_{i=0}^{N} \mathcal{G}_i(x)\alpha_i \prod_{j=0}^{i-1}(1-\alpha_j \mathcal{G}_j(x)) + \epsilon} \tag{2.5}$$

with ϵ being a small number for numerical stability. Given this rendered depth map $\mathbf{D}$, extracting a normal map $\hat{\mathbf{N}}$ is possible.

Next, we align both normals to align the extracted normal map from the depth map with the normals rasterized from the surfels.

$$\mathcal{L}_{nrml} = \sum_{k=0}^{M} 1 - \left\langle \hat{\mathbf{N}}, \mathbf{N} \right\rangle \tag{2.6}$$

We also used the disparity regularization that forces the Gaussians to aggregate. This regularization enables much better geometry and diminishes the presence of floaters. If we define ω_k as:

$$\omega_k = \mathcal{G}_k(\mathbf{u}(x))\alpha_k \prod_{j=0}^{k-1}(1-\alpha_j \mathcal{G}_j(\mathbf{x})) \tag{2.7}$$

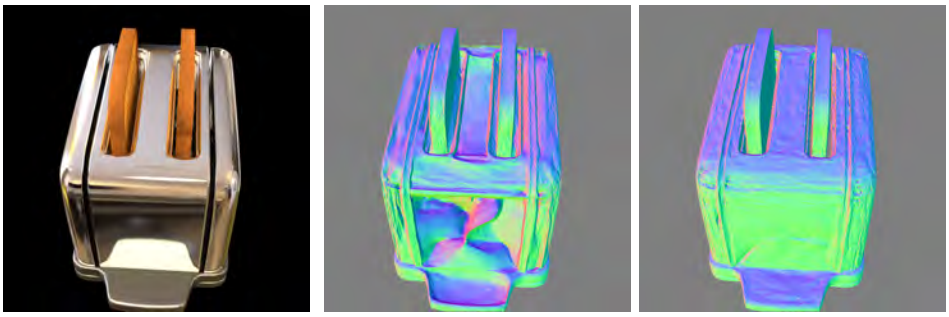
and also define the depth of each Gaussian as $z_k \in \mathbb{R}$, we have that the $\mathcal{L}_{disp}$ is defined as:

$$\mathcal{L}_{disp} = \sum_{k=0}^{N} \sum_{j=k+1}^{N} \omega_k \omega_j |z_j - z_k| \tag{2.8}$$

However, standard 2DGS can produce floaters and inaccurate geometric artifacts. To alleviate this issue, we perform depth supervision. The basic idea is that if we have a high-quality metric depth map, we can use a traditional L1 loss to align the depth maps. That is, suppose we have a ground-truth $\hat{\mathbf{D}}$, and we have a rendered depth map $\mathbf{D}$; then, we can align both of these maps using an L1 loss:

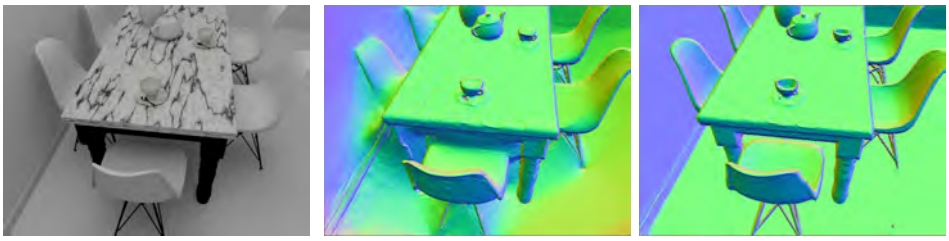
$$\mathcal{L}_{dep} = \sum_{k=0}^{M} \|\mathbf{D}_k - \hat{\mathbf{D}}_k\|_1 \tag{2.9}$$

We get good results for geometry reconstruction and view-synthesis if we perform this. This supervision can enormously improve the results for areas such as specular highlights, as shown in Figure 1. It also enables much greater detail for the surfaces, as in Figure 2 and, in some cases, can aid in the reconstruction of objects that are present in few views, as in Figure 3. As seen in these images, depth supervision can provide detail and fix the geometry for regions with specular highlights. However, obtaining high-quality metric data for most images can be challenging, although some feed-forward networks can obtain good-quality metric depth maps [12, 14].



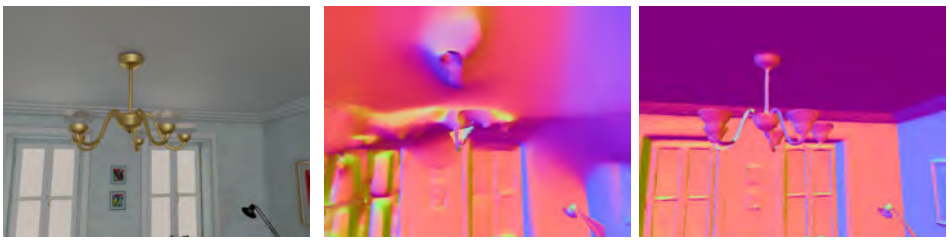
(A) Ground-truth image for "toaster" scene. (B) Predicted normals no depth supervision. (C) Predicted normals with depth supervision.

FIGURE 1. Comparison of predicted normals with and without depth supervision. Notice how, without depth supervision, the method can get lost in the specular highlights.



(A) Ground-truth image for "Breakfast" scene. (B) Predicted normals no depth supervision. (C) Predicted normals with depth supervision.

FIGURE 2. Comparison of predicted normals with and without depth supervision. Notice how the hard edges and the details are much clearer on the case with depth supervision.



(A) Ground-truth image for "Breakfast" scene. (B) Predicted normals no depth supervision. (C) Predicted normals with depth supervision.

FIGURE 3. Comparison of predicted normals with and without depth supervision. Notice how, without depth supervision, the chandelier does not appear. This issue arises since the chandelier appears in a few images.

Instead, we use a much weaker supervision signal: high-quality relative depth maps. These can be obtained using techniques such as Depth-anything-v2, which we will use. These relative depths are close to the ground-truth images up to a scale α and shift β parameters.

We use the same approach as in MonoSDF [15], where for each depth image pair $\mathbf{D}_k, \hat{\mathbf{D}}_k$, we solve a linear least-squares problem to find the α, β parameters:

$$\alpha_k, \beta_k := \arg\min_{\alpha,\beta} \|\hat{\mathbf{D}}_k - \alpha \mathbf{D}_k - \beta\|_2 \tag{2.10}$$

This has a closed-form solution, as it is a standard least-squares linear regression. After solving this problem and finding these parameters α_k, β_k for each depth image $\mathbf{D}_k$:

$$\mathcal{L}_{depth} = \sum_{k=0}^{M} ||\beta_k + \alpha_k \mathbf{D}_k - \hat{\mathbf{D}}_k||_2 \tag{2.11}$$

We define the final loss $\mathcal{L}$ function as the contribution of two terms, $\mathcal{L}_{data}$ and $\mathcal{L}_{reg}$, that is, a data term and a regularization term:

$$\mathcal{L} = \mathcal{L}_{data} + \mathcal{L}_{reg} \tag{2.12}$$

Where:

$$\mathcal{L}_{data} = \lambda_{img}\mathcal{L}_{img} + \lambda_{dep}\mathcal{L}_{dep} + \lambda_{SSIM}\mathcal{L}_{SSIM} \tag{2.13}$$

And:

$$\mathcal{L}_{reg} = \lambda_{disp}\mathcal{L}_{disp} + \lambda_{nrml}\mathcal{L}_{nrml} \tag{2.14}$$

We also perform a decay on the depth loss. Given the current iteration as **iter** and the total number of iterations as N_{iter} and an initial value for λ'_{dep}. We compute a new weight for each iteration by:

$$\lambda_{dep} := \lambda'_{dep}(1 - \frac{\mathbf{iter}}{N_{iter}}) \tag{2.15}$$

We use Adam [6] for the optimization and use the same adaptation strategies as in 3DGS [5].

2.2. **Back-face Culling**

In our framework, we also added a new technique inspired by the traditional idea of back-face culling. In the original 2DGS, the surfels are not oriented. However, this can have downsides since it depends on the representation in relation to the input camera pose views to orient the surface's normals. Due to this, we must learn how we can orient these surfels. To this end, we implemented a technique called back-face culling. Given a pixel in the image x, we define the direction of the ray that "shoots" from the camera center passing through the pixel x as $\mathbf{v} \in \mathbb{R}^3$. The intuition is that, assuming that the surfel is oriented, for a given surfel g_j, this

surfel should only be visible if $\langle V, N_j \rangle$ is positive. Implementation wise, we update α_j in the following manner:

$$\alpha_j := \begin{cases} 0 & \text{if } \langle V, N_j \rangle < 0 \\ \alpha_j & \text{otherwise} \end{cases}$$

We show it on the diagram in Figure 4

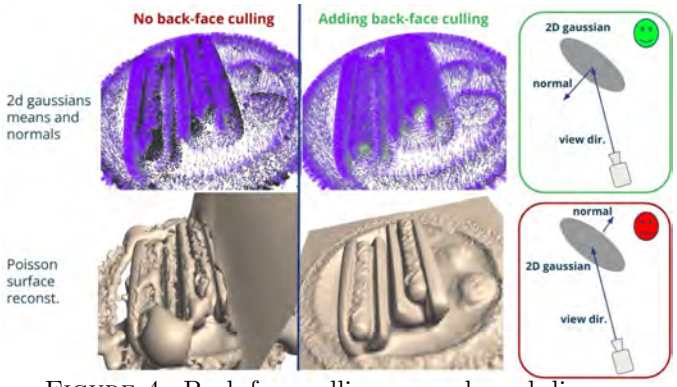


FIGURE 4. Back-face culling example and diagram

FIGURE 5. Visualization of surfels (bright means the normal is facing the camera, dark otherwise) of results of back-face culling. On the left, we have no back-face culling; on the right, we add back-face culling. Green means the surfel points to the camera, and black means otherwise. As seen in this image, adding back-face culling forced the surfels to be consistently aligned, as expected.

This modification forces the surfels to be aligned, as shown in Figure 5. Back-face culling is necessary for the surfaces to be consistent. This inconsistency can cause problems in tasks such as relighting [2] or if the user wants to extract a mesh from this representation using a technique such as Poisson Surface Reconstruction [4]. Using back-face culling forced the surfels to have a consistent orientation.

2.3. **Neural SDF**

In our representation, we also incorporated a Neural SDF due to its ability to model the geometry of objects. As previous works showed [7, 9, 10], Neural SDFs are a powerful tool to model continuous object geometry, being able to represent fine details and also able to be used to compute geometric attributes such as curvature [9].

A *neural SDF* is a neural network $f : \mathbb{R}^3 \to \mathbb{R}$, with parameters θ, having the property of being the *signed distance function* (SDF) of its zero-level set $f^{-1}(0)$. In other words, it satisfies the *Eikonal equation* $\|\nabla f\| = 1$. Here, we approach the problem of training f such that $S := f^{-1}(0)$ approximates the scene surface from *posed images* $\{I_i\}$.

FIGURE 6. Neural SDF fitted using the Gaussian surfels. This was performed during training of both the surfels and the neural SDF.

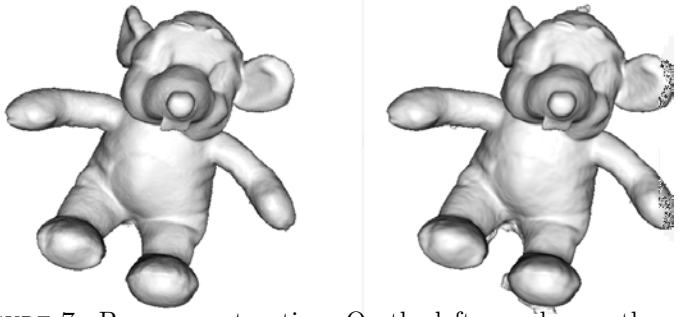


FIGURE 7. Bear reconstruction. On the left, we observe the mesh generated by the 2DGS technique, and on the right, we see a reconstruction created using the neural SDF as in Novello et al. [9].

To force g_j be a sample of f we ask for $\nabla f(\mu_j) = N_j$. The Gaussian normals align with the level set normals, indicating that g_j are *level set elements* of the

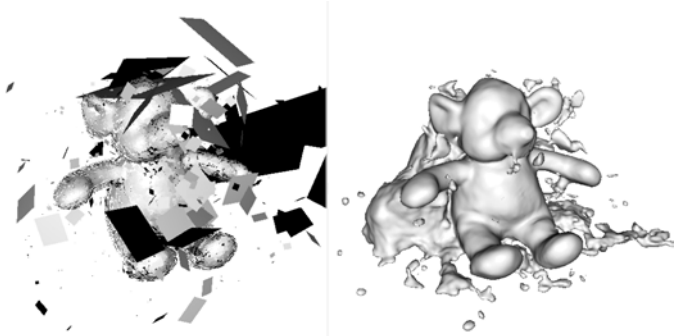


FIGURE 8. Bear reconstruction. In this case, we reconstruct directly from the surfels generated by 2DGS using i3d [9]. On the left, we see the surfels, and on the right, we see the reconstructed SDF.

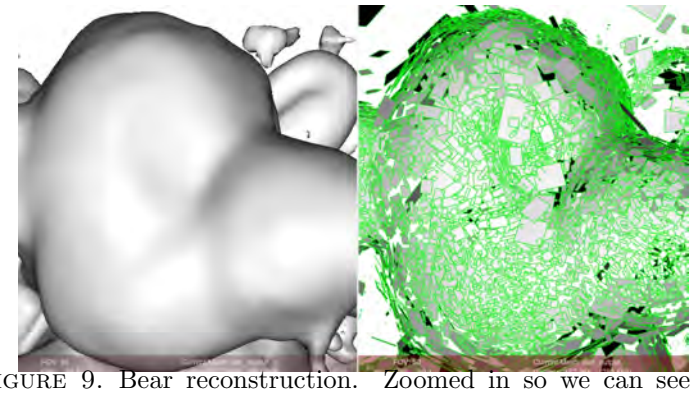


FIGURE 9. Bear reconstruction. Zoomed in so we can see the distribution of the surfels.

neural SDF f. These can be done in parallel while training the Gaussians. Figure 6 shows an example of training the SDF and the surfels in parallel. However, another approach is to train the SDF after the surfels g_j are trained. In this case, we can, for example, train with the extracted mesh. The reconstruction results using the techniques described in Novello et al. [9] are in Figure 7. In Comparison, we get a much worse result if we train the SDF with the surfels as in Figure 8. We can also observe the distribution of surfels more clearly in Figure 9. As can be seen, the surfels are tangent to the SDF, as expected.

3. Depth Evaluation

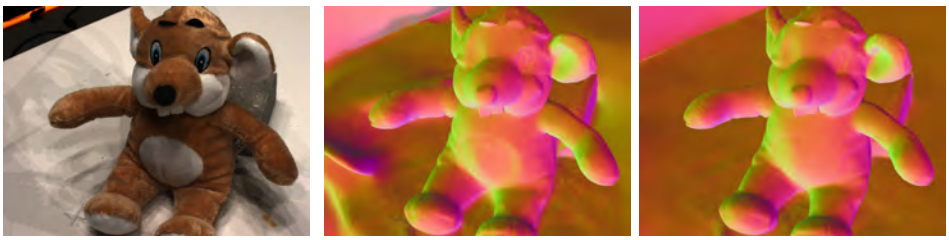
In this section, we evaluate the impact of depth supervision on 3D reconstruction. The following tests will be performed without the Neural SDF and without backface culling. We tested with 10,000 iterations, roughly 4 minutes on an RTX 4090 GPU. To test it, we measured the Chamfer Distance in Table 1, showing that our results were better than the original 2DGS while being better in most cases.

(A) Ground-truth image for "scan65"

(B) Predicted normals no depth supervision.

(C) Predicted normals with depth supervision.

FIGURE 10. Comparison of predicted normals with and without depth supervision. Notice how the depth supervision improved the holes in the skull, especially on the teeth and nose.

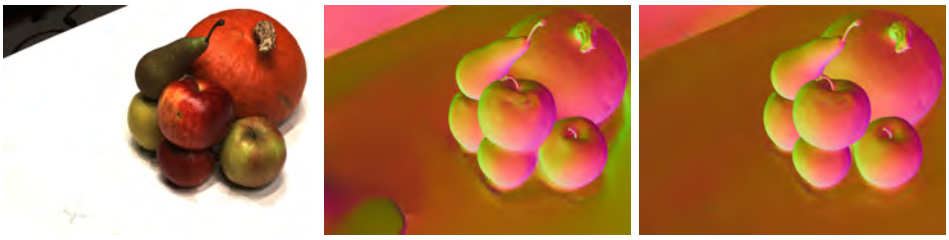


(A) Ground-truth image for "scan105"

(B) Predicted normals no depth supervision.

(C) Predicted normals with depth supervision.

FIGURE 11. Comparison of predicted normals with and without depth supervision. Notice how depth supervision improved the fine details in this scene in regions such as the ears and the details on the belly of the animal.



(A) Ground-truth image for "scan63"

(B) Predicted normals no depth supervision.

(C) Predicted normals with depth supervision.

FIGURE 12. Comparison of predicted normals with and without depth supervision. Notice how the depth supervision improved the delimitation of the fruits. As it is possible to see, there is much more detail in the delimitation of the boundaries of the fruits. The apples seem to "merge" without depth supervision instead of being separated.

Qualitatively, we can observe that depth supervision improved the geometry for challenging scenes. For example, in scenes with holes or other details, we can observe that depth supervision improved the result, as seen in Figure 10, especially on the teeth and nose of the skull. Depth supervision also improved other details, such as the belly of the "Bear" in Figure 11. Although not the objective of the reconstruction, it should be noted that depth supervision improved the reconstructed background considerably, with the desk on this object being of much higher quality. Finally, depth supervision enabled boundaries to be established between the cluttered objects, as seen in Figure 12.

Evaluated Scene	**Without Depth**	**With Depth**
scan24	0.7266	**0.6373**
scan37	0.9935	**0.9289**
scan40	**0.5805**	0.6852
scan55	**0.4309**	0.4643
scan63	1.1854	**1.0940**
scan65	1.0724	**0.9694**
scan69	0.8551	**0.7757**
scan83	0.8359	**0.7124**
scan97	1.0798	**0.8734**
scan105	0.6205	**0.5421**
scan106	0.6174	**0.5522**
scan110	1.3468	**1.0723**
scan114	0.5097	**0.4774**
scan118	0.6661	**0.5954**
scan122	0.5248	**0.5153**
Mean	0.8030	**0.7264**

TABLE 1. Comparison of Results with Depth Supervised and Without Depth using Chamfer Distance. Lower values indicate that the mesh is closer to the ground truth.

4. Conclusion

Our experiments demonstrated, although in its early steps, the impact of using an approach based on depth maps with 2DGS. We also showed how we can use the surfels to obtain an SDF and, finally, how back-face culling can improve in this setting. Although this report still needs to address many future avenues of research and open problems, we hope that it serves as a compilation of some experiments regarding Gaussian splatting for geometry reconstruction.

References

[1] Illya Bakurov, Marco Buzzelli, Raimondo Schettini, Mauro Castelli, and Leonardo Vanneschi. Structural similarity index (ssim) revisited: A data-driven approach. *Expert Systems with Applications*, 189:116087, 2022.

[2] Jian Gao, Chun Gu, Youtian Lin, Hao Zhu, Xun Cao, Li Zhang, and Yao Yao. Relightable 3d gaussian: Real-time point cloud relighting with brdf decomposition and ray tracing. *arXiv preprint arXiv:2311.16043*, 2023.

[3] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–11, 2024.

[4] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, volume 7, 2006.

[5] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4):1–14, 2023.

[6] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[7] Zhaoshuo Li, Thomas Müller, Alex Evans, Russell H Taylor, Mathias Unberath, Ming-Yu Liu, and Chen-Hsuan Lin. Neuralangelo: High-fidelity neural surface reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8456–8465, 2023.

[8] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

[9] Tiago Novello, Guilherme Schardong, Luiz Schirmer, Vinícius da Silva, Hélio Lopes, and Luiz Velho. Exploring differential geometry in neural implicits. *Computers & Graphics*, 108:49–60, 2022.

[10] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *arXiv preprint arXiv:2106.10689*, 2021.

[11] Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. Neural fields in visual computing and beyond. In *Computer Graphics Forum*, volume 41, pages 641–676. Wiley Online Library, 2022.

[12] Lihe Yang, Bingyi Kang, Zilong Huang, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything: Unleashing the power of large-scale unlabeled data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10371–10381, 2024.

[13] Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything v2. *arXiv:2406.09414*, 2024.

[14] Wei Yin, Chi Zhang, Hao Chen, Zhipeng Cai, Gang Yu, Kaixuan Wang, Xiaozhi Chen, and Chunhua Shen. Metric3d: Towards zero-shot metric 3d prediction from a single image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9043–9053, 2023.

[15] Zehao Yu, Songyou Peng, Michael Niemeyer, Torsten Sattler, and Andreas Geiger. Monosdf: Exploring monocular geometric cues for neural implicit surface reconstruction. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[16] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Ewa splatting. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):223–238, 2002.

IMPA - VISGRAF, Rio de Janeiro, Brazil

E-mail address: daniel.perazzo@impa.br