Report on "Texture-GS: Disentangling the Geometry and Texture for 3D Gaussian Splatting Editing"

Reviewer Fernando
firstauthor@i1.org

Archaeologist Diana Aldana
IMPA
diana.aldana@impa.br

Hacker Leoanrdo Nanci
UFF
leonardonanci@id.uff.br

PhD Student Victor Ferrari
UFF
victorferrari@id.uff.br

1. Reviewer

Sugestão para o Revis@r: ler as orientações do CVPR. Tentem manter o relatório dentro de 8 páginas.

1.1. Summary

Descreva brevemente o método e sua contribuição para *visão computacional* ou *computação gráfica*. Forneça sua avaliação sobre o escopo/magnitude da contribuição do artigo. Segue uma sugestão de estrutura para o resumo.

- Problema abordado;
- Motivação;
- Resumo do método;
- Lista de contribuições.

Descreva o método e faça uma ánalise justificando cada formula utilizada. A exposição está clara? Como poderia ser melhorada?

1.2. Strenghts

- Ideias interessantes validadas através de experimentalmente e de forma teórica, novas ferramentas, resultados impressionantes, ...
- O que alguém da área aprenderia lendo o paper?

1.3. Weaknesses

- Falta de experimentos (quais?)
- Alegações enganosas e erros
- Difícil de reproduzir (Participação do Hacker)

1.4. Evaluation

Dê uma classificação geral do trabalho e do artigo em uma escala contínua de 1 a 5, onde 1 é o pior e 5 é o melhor. Especificamente: 1 = Rejeitar, 2 = Possivelmente rejeitar, 3 = Duvidoso, 4 = Possivelmente aceitar, 5 = Aceitar.

Deve ficar claro quais dos pontos positivos e negativos foram mais considerados.

2. Archeaologist

Disentangling textures and texturing 3D objects are classical problems in computer vision. Currently, one of the most common approaches to pass a texture to an object it is to represent an object by a mesh and use an uv-parametrization. In general, it is desirable for it to be a bijective function between the 3D space and a parametric domain. However, it is not straightforward as to how to define this function for other representations of 3D objects such as NeRF [2] or 3D Gaussian Splatting (3DGS) [1].

In that sense, there have been many works tackling the problem of texturing NeRFs [3, 5, 7]. In particular, we highlight NeuTex [6] as the main inspiration of Texture-GS. This work also aims to learn a parametrization of the space using a bijective function F_{uv}. Specifically, this function is trained so that it approximates to a bijective function using the loss term

$$\mathcal{L}_{cycle} = \sum w_i || F_{uv}^{-1}(F_{uv}(\mathbf{x}_i)) - \mathbf{x}_i ||_2^2.$$

Note that this loss also appears in Texture-GS. Note that this function is trained using exclusively the position in the space. On the other hand, the view-dependent information given by the angles (θ, ϕ) are given as an input to the function F_{tex} along with the (u, v) coordinates, which returns the color in that coordinate. Observe that this function stores the original texture using the spherical topology. Finally, the geometry is encoded in an opacity function F_σ. The complete pipeline is shown in Figure 1.

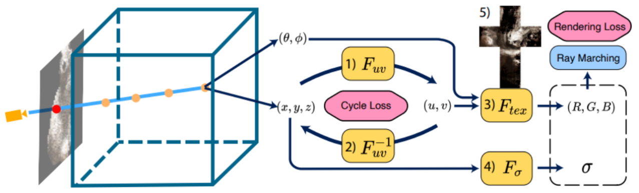

Figure 1. NeuTex [6] method pipeline.

Six months later was uploaded in arxiv the work GStex [4]. It tackled the same problem of decoupling appearance from geometry, but used a different scheme to show improvements compared to Texture-GS. It trained a *texture map/tiling* for each gaussian, which would be used to compute the diffuse (non-constant) term of the spherics harmonics that defined the color. This scheme allowed for bigger variations in the color domain, reconstructing more detailed objects under similar conditions that Texture-GS (see Figure 2)

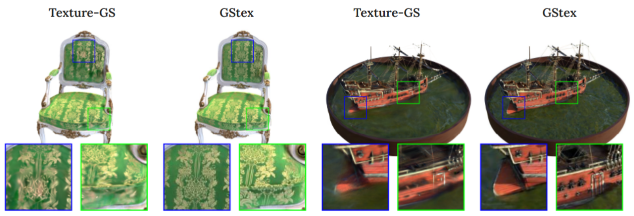

Figure 2. Comparison between Texture-GS and GStex

3. Code and experiments

The authors make the source code associated with the article available at the following repository: `https://github.com/slothfulxtx/Texture-GS`.

The repository is well-structured, due to its folder scheme and modules that divide the files according to their responsibility in the pipeline. The README is clear and includes simple commands, thanks to the provided scripts and YAML to configure the different executions.

In general, the equations presented in the article are implemented clearly. However, there are discrepancies in certain parts of the code. For example, in the computation of Equation 5, based on the Chamfer Distance, the import of the function that computes the distance is commented out without apparent reason (recently fixed). Additionally, in the implementation of Equations 4 and 6, the neural networks ϕ and ϕ^{-1} receive an extra parameter geo_emb of type $torch.embedding$, which is not mentioned in the article, and whose origin and function could not be deduced.

The greatest difficulty encountered in reproducing the method was installing the dependencies; specifically, installing compatible versions of the CUDA environment with the Torch and PyTorch3D packages. The issue was resolved by configuring a ”clean” environment on the Google Colab online notebook platform.

However, due to the limitations of using the platform's hardware acceleration, it was not possible to complete all steps of the pipeline proposed by the authors. The steps of extracting a 3DGS model and training the texture mapping functions were carried out. However, the texture extraction exceeded the available GPU access time. Based on the execution time of the third step and the displayed logs, the code seemed to be able to fulfill the task, given enough time.

For the experiments presented in class, pre-trained models available in the aforementioned repository were used. Initially, I tested the *re-texture* script, which enables inserting a new image to texture the model. As shown in Fig. 3 (left), the generated results were satisfactory, with the altered texture maintaining the shades of the original model. Finally, I experimented zeroing the Jacobian used to approximate the texture points around the center of each Gaussian. The goal was to analyze the level of detail captured by the texture relative to that recorded by the Gaussians. As observed in Fig. 3 (right), much of the object's lighting was lost, leading to the conclusion that the Gaussians are responsible for coarse modeling, while the texture represents the finer characteristics.

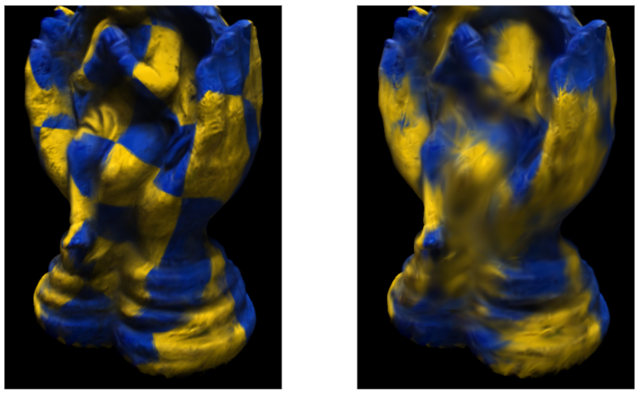

Figure 3. Experiments results: Re-rextured (left) and Zeroed Jacobian (right)

4. PhD project

We propose two different approaches that could be used based on the studies of TextureGS. The first proposal is to develop an application to improve the 3D artist workflow. This application would help with the challenges mentioned in the TextureGS paper, such as the complexity of editing conventional 3D representations. Based on the paper's results, which decouples geometry and appearance by mapping 2D textures onto 3D surfaces, the tool would provide a platform for real-time editing. Artists could modify textures and view the results, leveraging the real-time rendering capabilities of consumer-level GPUs, similar to the performance shown in the Texture-GS paper.

The application would include automated tools for creating UV maps, reducing the need for manual adjustments. Combining a UV mapping module with the one used in Texture-GS would automatically generate the UV coordinates, improving efficiency in the creation process. Additionally, the tool would integrate with existing software such as Blender, Maya, or Unreal Engine, allowing artists to make adjustments and export their work. The result would be an optimized pipeline that enhances creative flexibility, enabling texture swapping and fine-grained customization in a more accessible and efficient manner.

The second proposal focuses on improving texture manipulation by compartmentalizing textures based on the specific features of the 3D object, such as materials, regions, or geometric properties. As described in the paper, the method of disentangling geometry and texture could be extended to generate distinct textures for different object attributes, such as metallic surfaces, translucent parts, or opaque areas. This would allow for a more granular approach to texture editing.

The system would use machine learning techniques to automatically identify and classify different object parts based on their physical and visual characteristics. Once classified, it would enable the export of modular textures optimized for different editing software, improving the performance of the rendering pipeline. Additionally, this approach would support more complex operations, allowing artists to adjust individual regions without affecting others, thus providing detailed control over the object's appearance. For example, a 3D object with both a glass surface and a metal frame could have its textures separated and edited independently, preserving the integrity of the design while providing flexibility for fine-tuned adjustments.

Both proposals build on the advancements introduced in Texture-GS, offering solutions that simplify the 3D artist pipeline and improve performance and accessibility. These innovations would benefit gaming, animation, and virtual reality industries by simplifying workflows and enabling more efficient editing.

5. Conclusões

Apresente as conclusões, sugestões de título e um resultado ausente que o artigo poderia ter incluído.

References

[1] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023. 1

[2] 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. 1

[3] Keihachiro Moriyasu. *An elementary primer for gauge theory*. World Scientific, 1983. 1

[4] Victor Rong, Jingxiang Chen, Sherwin Bahmani, Kiriakos N Kutulakos, and David B Lindell. Gstex: Per-primitive texturing of 2d gaussian splatting for decoupled appearance and geometry modeling. *arXiv preprint arXiv:2409.12954*, 2024. 2

[5] Pratul P Srinivasan, Stephan J Garbin, Dor Verbin, Jonathan T Barron, and Ben Mildenhall. Nuvo: Neural uv mapping for unruly 3d representations. In *European Conference on Computer Vision*, pages 18–34. Springer, 2025. 1

[6] Fanbo Xiang, Zexiang Xu, Milos Hasan, Yannick Hold-Geoffroy, Kalyan Sunkavalli, and Hao Su. Neutex: Neural texture mapping for volumetric neural rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7119–7128, 2021. 1, 2

[7] Fangneng Zhan, Lingjie Liu, Adam Kortylewski, and Christian Theobalt. General neural gauge fields. *arXiv preprint arXiv:2305.03462*, 2023. 1