2D Gaussian Splatting for Geometrically Accurate Radiance Fields

Reviser
Esteban Wirth
esteban.wirth.97@gmail.com

Archaeologist
Esteban Wirth
esteban.wirth.97@gmail.com

Hacker
Leonardo Mendonça
IMPA
leonardo.mendonca@impa.br

PhD Student
Leonardo Mendonça
IMPA
leonardo.mendonca@impa.br

1. Review

1.1. Summary

The 2D Gaussian Splatting for Geometrically Accurate Radiance Fields centers itself around the improvement of the 3DGS [4] paper which predates it. Its objective is to improve the recreation of the scene by having a better control of depth calculation and having a more accurate method to model surfaces without overly restricted conditions or highly dense cloud points.

In order to do this they borrow the Gaussian equations of [4]

$$G(p) = exp(-\frac{1}{2}(p-p_k)^T\Sigma^{-1}(p-p_k)) \tag{1}$$

Where the covariance matrix in equation 1 is defined as $\Sigma = RSS^TR^T$ with R is a rotation matrix and S a scaling matrix. Still from the 3DGS paper [4] they adapt this representation in the object world to the camera image as

$$\Sigma' = JW\Sigma W^TJ^T \tag{2}$$

Where J and W are matrices that rotate and project respectively the scene onto the image.

To adapt these equations to the 2D case they cut out the third row and column. Unfortunately the reasons as to why they do this are not explained and at face value seem mathematically wrong given the Gaussians that are being used are not normalized so this operation doesn't project correctly the 3D gaussians into 2D ones.

They use this 2D gaussians in order to define a view dependent color value for points in space.

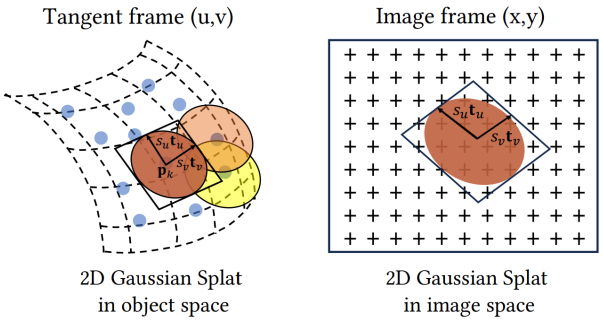


Figure 1

$$c(x) = \sum_{k=1}^{K} c_k\alpha_k G_k^{2D}(x)\prod_{j=1}^{k-1}(1-\alpha_j G_j^{2D}(x)) \tag{3}$$

Equation 3 determines the color value of the center of each gaussian which accumulates oppacity defined as α.

After determining this they begin the modelling of the surfels in object and image space as shown in Figure 1.

They then introduce the following two equations.

$$P(u,v) = p_k + s_ut_uu + s_vt_vv = H(u,v,1,1)^T \tag{4}$$

$$H = \begin{bmatrix} s_ut_u & s_vt_v & 0 & p_k \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5}$$

The above equations are dimensionally incoherent. The idea they are conveying is they can embed model the 2D

gaussians using the center and variation vectors through a linear transformations and embedding the variation vectors in 4D space.

They then proceed to define a second way to model a gaussian

$$G(\mathbf{u}) = exp(-\frac{u^2 + v^2}{2}) \quad (6)$$

abusing notation and giving no context as to why they have now 2 different definitions for the gaussians.

To create a way to relate the image space to the object space they define an embedding and projection of the points as follows

$$\mathbf{x} = (xz, yz, z, 1) = WP(u, v) = WH(u, v, 1, 1) \quad (7)$$

Where z is determining the depth to the point from a certain view point. To determine the intersection from a point (x, y) in image space to a point in object space they define the following elements in the 4D embedding space. They represent a ray passing through this point as the intersection of the two orthogonal plains that contain pass through the coordinates x and y. They claim that in 4D embedding space this can be modelled as $h_x = (-1, 0, 0, x)^T$ and $h_y = (0, -1, 0, y)^T$. There are no in depth arguments as to why this is true or mathematically rigurous. By using the previous transformations they are able to create an efficient method to calculate the transformation from image to object space.

$$h_u = (WH)^T h_x \quad h_v = (WH)^T h_y \quad (8)$$

By having orthogonal vectors they are able to develop an efficient way of calculating these values even though they are not fully explained in the paper.

To handle degenerate solutions they include a minimum width that the gaussians have to have in order to not loose them due to being two dimensional. Giving rise to the final gaussian function that will define the color function that was based on [4].

$$\hat{G}(\mathbf{x}) = max\{G(\mathbf{u}(\mathbf{x}), G(\frac{x - c}{\sigma}\} \quad (9)$$

This equation abuses notation and has no explanation for the value of σ that is being used.

After this they introduce two new loss functions which are the biggest improvements of the previous models. The depth loss function defined as

$$L_d = \sum_{i,j} \omega_i \omega_j |z_i - z_j| \quad (10)$$

Where ω represents the blending of opacity without consideration of color. This loss functions allows the gaussians to stick together and create coherent objects.

The other innovative loss function that they develop was the normal loss function which compares the normal created by the gaussians to the normal calculated through finite differences over the image. This allows for a consistency between the normals observed in the cameras and the normals rendered in the scene.

Finally they combine these two loss functions with the color loss function taken from [4] to create the total loss function.

The results of experiments where good as compared with other state of the art models.

In conclusion this paper innovated over the loss functions and created a working model that rendered scenes in a very good way. Unfortunately the lack of mathematical rigorous makes the paper unacceptable as a scientific development. For this it is advised the paper is resubmitted with a more technical and thorough mathematical explanation of their methods.

2. Archaeologist

2 Dimensional Gaussian Splatting is one of the precursors of the usage of Gaussian's to render three dimensional scenes. It builds from the original idea of 3DGS and advances to a better method to determine depth and model scenes by using two dimensional Gaussians. Particularly it draws inspiration from two fundamental papers.

First it cites and takes the main idea of 3DGS [4]. Specifically it uses the notion of using Gaussians to model a scene with view dependent properties. Furthermore it inherits one of the techniques used to assure convergence from [4] which is densification that modifies the number of Gaussians by splitting or merging them.

From this base idea it combines the notion of modelling the surface of a volume via two dimensional Gaussians from [7]. Where they take the mathematical intuitions of using surfels to model an object but improve from it by calculating the normal of the surfel as the cross product over the minimum and maximum change directions of each gaussian. This allowed [3] to improve the results particularly require less dense point clouds to model an object. They managed to do this by also including two new loss functions that assure a better depth reconstruction and consistency between the modelled normals of the gaussians and

the observed normals of the cameras.

From these innovations other papers have been able to grow and develop. For example [6] developed a model to create 4D scenes from generated videos. This are videos where the viewer can move through the scene while the video is occurring. It used the loss functions of 2DGS to be able to create these scenes and model them appropiately. To be able to adapt the 2DGS model to create a 4D model they included a distortion element that allowed the scene to vary through time.

3. Code and experiments

We have executed the 2DGS code, made available by the authors in https://github.com/hbb1/2d-gaussian-splatting. Despite the installation instructions present in the *README.md*, there were some difficulties:

1. The *environment.yml* file was not complete: we had to add the python modules *setuptools* and *matplotlib* for the installation and code to work without errors.
2. The 2DGS documentation fails to mention CUDA, whose installation is necessary for training and inference to work properly. We also went through a process of trial-and-error in order to install the right version of CUDA and the correct C++ kernels.

In spite of the problems with the installation, the execution of the code is made relatively straightforward by the command line interface implemented by the authors. Most of the hyperparameters can be set through command line arguments when calling *train.py*, as well as input and output names, image resolution and so on.

3.1. Code analysis

The most performance-heavy parts of the training process, notably the gaussian rendering, are done inside a CUDA-optimized C++ script, which employs the user's GPU for fast computing. The python files, however, are responsible for training, pruning and often calling the CUDA C++ files required to do so.

There were some differences between the methodology presented in the paper and the actual code implementation.

The densification strategy, mentioned only in passing in the text, is taken directly from 3DGS [4], and uses mostly the same hyperparameters. Between the training iterations *densify_from_iter* (default 500) and *densify_until_iter* (15000), the model will periodically split or clone certain gaussians, depending on their scale. After *opacity_reset_interval* (3000) epochs, the gaussians with opacity lower than *opacity_cull* (0,05) are removed, while the remaining ones have opacity reset to 0,01 (hardcoded). These hyperparameters are given without justification in [4] and not mentioned at all in [3].

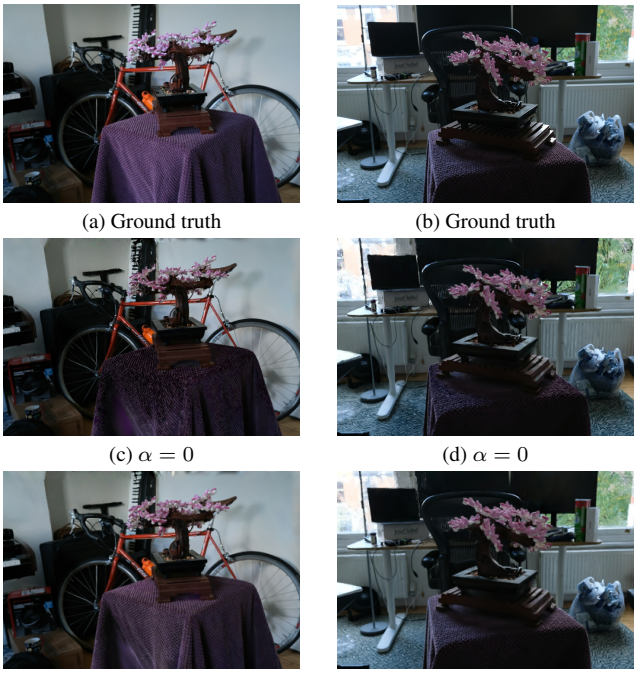
The number of spherical harmonics for anisotropic coloring in each gaussian *sh_degree* (set to 3) is another "hidden" hyperparameter. The number of iterations before applying depth and normal regularization, respectively 3000 and 7000, are not only unmentioned in the text, but also hardcoded, which hinders future experiments to study the effect of these parameters.

3.2. Experiments

3.2.1 Depth distortion regularization

This regularization loss term seeks to minimize the distance between the depth of different gaussians intercepted by the same ray, therefore grouping the gaussians near the physical surface of the object. The paper [3] suggests using the weight parameter $\alpha = 1000$ for bounded and $\alpha = 100$ for unbounded scenes. However, the code sets the default value of this parameter to zero, and, in the authors' own experiments, the value of α used changes from dataset to dataset. In the MipNeRF360 [1] dataset, this regularization is not used at all, and in Tanks and Temples [5], α is set to either 10 or 100, depending on the scene, and regardless of whether it is bounded or unbounded.

As such, in order to investigate the importance of this weight parameter, we have trained the model on the Bonsai scene of the MipNeRF360 dataset with resolution 390×260, setting alternatively $\alpha = 0$ (as used in the authors' tests) and $\alpha = 1000$ (textually recommended in the paper for bounded scenes such as this one). Results are shown in fig. 2. One may observe in these tests that there are artifacts near the upper edge of frame 0, which is likely not covered by the training views, and a significant distortion in the glass windows of view 12, which illustrates 2DGS's inadequacy for modeling semi-transparent materials. There is no apparent difference, however, between the rendered images using $\alpha = 0$ and $\alpha = 1000$. This suggests that perhaps the depth distortion regularization does not have as strong an impact as suggested by the authors.



(a) Ground truth (b) Ground truth

(c) $\alpha = 0$ (d) $\alpha = 0$

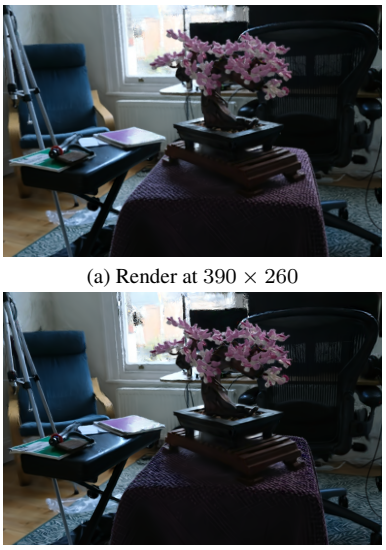
(e) $\alpha = 1000$ (f) $\alpha = 1000$

Figure 2. MipNeRF360 Bonsai scene, frame 0 (left) and 12 (right), after 30000 training iterations, rendered after training with different values of depth distortion weight α.

3.2.2 Training and rendering resolution

After training in 390×260 resolution, we rendered the optimized scene in a higher resolution of 1559×1039. The goal of this experiment was to investigate the method's performance for novel-view synthesis in higher resolutions than the one it was trained on. Since the results were very similar for the reconstructions with and without the depth distortion, we use here $\alpha = 0$, the same value used by the authors in their evaluation of the MipNeRF360 dataset. Results are shown in fig. 3.

We conclude that reconstruction quality is directly connected to the rendering resolution: in the high-resolution render, the image is characterized by very thin gaussians, inconsistent with the scene geometry, which are not visible in low resolution and were therefore not penalized during training.



(a) Render at 390×260

(b) Render at 1559×1039

Figure 3. Comparison between images rendered with different resolutions, after being trained on 390×260 resolution. MipNeRF360 Bonsai scene, frame 15

4. Projeto de doutorado

4.1. 2D Half-Gaussian Splatting

The paper showed very positive results for surface reconstruction and novel-view synthesis using planar gaussians, which can be used to accurately model a great variety of smooth surfaces.

However, there are many human-made objects and structures that possess sharp edges, or that are better represented by manifolds with boundary. We propose therefore a modification of the method by replacing the 2D gaussians used in the paper with half-gaussians, defined by the distribution:

$$G(u,v) = \begin{cases} 2e^{-\frac{u^2+v^2}{2}} \text{ , if } v \geq 0 \\ 0 \text{ , if } v < 0 \end{cases} \tag{11}$$

This modified shape allows one to economically represent surfaces with a common edge or surfaces whose boundary is a 1D manifold. An expected difficulty, however, is the fact that eq. 11 has a discontinuity at $v = 0$. In order for the function to be differentiable, we will need to define a smoothing function between the two half-planes $v < 0$ and $v > 0$, in order to allow for backpropagation. The "sharpness" of this smoothing function can be either set as a hyperparameter or as a trainable parameter for each half-gaussian, but systematic ablation studies are necessary to decide which strategy is more performant.

4.2. Mixed 2-3D Gaussian Splatting

Despite the good results for opaque surfaces, 2DGS reports poor reconstruction quality for translucent materials, which we can also observe in fig. 2. However, these materials can be accurately represented by volumetric (3D) gaussians, as seen in 3DGS [4] and EWA Splatting [9]. Therefore, we propose a mixed approach, where 2D and 3D gaussians coexist and can be jointly optimized to accurately capture both opaque surfaces and translucent volumes.

A natural objection is the matter of initialization: each point in the initial point cloud, produced by COLMAP, should initialize to either a 2D or 3D gaussian, but it is no evident that there is a way to determine beforehand what dimension each gaussian should have. One possible approach is to initialize every gaussian as 3D, and convert each of them to 2D if the ratio between its 3 dimensional scales reaches a certain threshold (i.e. if the gaussian becomes "flat enough").

We may also cite the drawback that, once we mix the 2D and 3D gaussians, we lose 2DGS's ability to obtain surface normals directly, which was useful for normal regularization and for mesh extraction. On the other hand, one can use of the approaches defined in GOF [8] or SuGaR [2], both of which obtained good results in characterizing the normal of a 3D gaussian.

Once these obstacles have been overcome, we should have a robust representation for novel-view synthesis in scenes that mix opaque and semi-transparent objects.

5. Conclusions

In conclusion even though the paper was successful in developing a working model the method used throughout it both in the code and math where unsatisfactory. It would have been great to see a result over noisy data to see the sturdiness of the model under more likely conditions when used in real life.

References

[1] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. *CVPR*, 2022. 3

[2] Antoine Guédon and Vincent Lepetit. Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5354–5363, 2024. 5

[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. 2, 3

[4] 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, 3, 5

[5] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics*, 36(4), 2017. 3

[6] Yikai Wang, Xinzhou Wang, Zilong Chen, Zhengyi Wang, Fuchun Sun, and Jun Zhu. Vidu4d: Single generated video to high-fidelity 4d reconstruction with dynamic gaussian surfels, 2024. 3

[7] Wang Yifan, Felice Serena, Shihao Wu, Cengiz Öztireli, and Olga Sorkine-Hornung. Differentiable surface splatting for point-based geometry processing. *ACM Transactions on Graphics*, 38(6):1–14, 2019. 2

[8] Zehao Yu, Torsten Sattler, and Andreas Geiger. Gaussian opacity fields: Efficient adaptive surface reconstruction in unbounded scenes. *ACM Transactions on Graphics*, 2024. 5

[9] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Ewa volume splatting. In *Proceedings Visualization, 2001. VIS'01.*, pages 29–538. IEEE, 2001. 5