# Reducing Overfitting in 3D Gaussian Splatting using Depth Supervision

**Tygo Hendrik Bertus Spanhoff**[1]

**Supervisor: Xucong Zhang**[1]

[1]**EEMCS, Delft University of Technology, The Netherlands**

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 23, 2024

Name of the student: Tygo Hendrik Bertus Spanhoff
Final project course: CSE3000 Research Project
Thesis committee: Xucong Zhang, Michael Weinmann

## Abstract

3D Gaussian Splatting (3DGS) is a method for representing 3D scenes, but is prone to overfitting when trained with limited viewpoint diversity, often resulting in artifacts like floating Gaussians at incorrect depths. This paper addresses this issue by introducing 3D Gaussian Splatting with Depth, which incorporates depth supervision from RGB Depth (RGB-D) cameras into the training process. By using depth data to guide the placement of Gaussians, the proposed method aims to reduce artifacts. Through quantitative and qualitative analysis, this paper demonstrates that depth-supervised Gaussian splatting mitigates overfitting artifacts, particularly in outdoor scenes with a mediocre camera point diversity. The depth-supervised model is able to reduce the depth loss by a factor of three times without substantially increasing the loss on regular views.


## 1 Introduction

The representation of 3-dimensional scenes is a fundamental challenge in the fields of computer graphics and computer vision. The ability to turn real-world objects into 3D representations on computers to generate novel views can be applied in media such as films and video games [10], for robots and navigation systems [10], and for medical applications such as CT scans [8].

First introduced in 2020, Neural Radiance Field (NeRF) [7] is a novel view synthesis method that represents 3-dimensional scenes using a Deep Neural Network. NeRF's ability to render photorealistic views in a reasonable time sparked significant interest in the field of radiance field rendering. Although revolutionary, NeRF came with some significant disadvantages, one of the main ones being that the representation as a Multi-Layer Perceptron requires an entire model to be retrained in order to make changes to a scene.

In July of 2023, Kerbl et al. introduced a novel way of representing 3D scenes using 3D Gaussians, which allowed for realtime rendering of scenes captured with multiple photos [5]. This technique was generally faster in training and rendering than previous techniques such as NeRF, and the representation also allowed for changes to the scene without having to retrain the model.

3D Gaussian Splatting (3DGS) is particularly prone to overfitting due to its sensitivity to limited viewpoint diversity in the training data. When training a Gaussian splat, it is possible for some parts of the scene to be visible in only one or a select amount of training frames. As such, the model will place a Gaussian of the corresponding color in the scene such that the rendered splat will look similar to the training frame when rendered from the same camera position, as can be seen in figure 1. However, since the part of the scene is only shot from one angle, the model could place the Gaussian at any distance from the camera. This could cause Gaussians to appear at the wrong depth, floating in space when rendered from a novel view, as can be seen in figure 1.

Figure 1: The white Gaussians representing the sky look decent on one view, but can be seen floating in mid-air when rendering the scene from a novel view.

This paper investigates a potential solution for this problem by training a Gaussian splat using images from RGB-D cameras. RGB-D cameras have corresponding depth data for every frame, and these cameras are becoming increasingly more available both as separate devices and included in mobile phones [2]. It extends the original paper by investigating if, how, and how much Gaussian splatting can further be improved if depth information is available. The paper will provide a qualitative comparison between RGB Gaussian splatting and RGB-D Gaussian splatting by investigating floating Gaussians in novel views. As such, the main research question is: How to exploit additional depth information from RGB-D measurements in 3DGS?

This will be investigated by introducing 3D Gaussian Splatting with Depth (3DGSw/Depth), a version of 3DGS that renders depth maps from the Gaussian Splat and supervises them with the depth ground truth from RGB-D cameras. This system will then be evaluated by analyzing the loss during training, and by comparing renders from models trained with both 3DGSw/Depth and 3DGS.

## 2 Related Work

3DGS uses Structure-from-Motion (SfM), particularly COLMAP [9], and gradient descent to create a Gaussian Splat from a set of 2D RGB images. Before the Gaussian Splat is created, COLMAP uses features such as corner points to find correspondence between the input images and is then able to estimate the 3D structure in the scene, as well as camera positions and intrinsic parameters. COLMAP outputs a sparse point cloud, which is then used to initialize a sparse cloud of Gaussians, where each Gaussian has a position (mean), covariance matrix, and opacity alpha. Then, a stochastic gradient descent procedure is used. For multiple camera positions per iteration, the Gaussians are projected to 2D and blended using alpha-blending to result in a rendered image. These renders are then compared to the original image using the L1 loss function, which is the absolute difference, together with a D-SSIM term 2. The Gaussians can be cloned, removed, or changed in every iteration in order to converge to a minimum loss by gradient descent.

Since the original paper on 3DGS has been published, multiple researches have advanced on the technique using depth information. Luiten et al. have used RGB-D cameras in order to obtain sparse point clouds without having to use a lot of camera positions like SfM requires [6]. Their research also briefly describes the rendering of depth maps from Gaussian splats by replacing the color component of each Gaussian by the depth value of its center. Depth information has also been used to improve the training of a Gaussian model [4], although the depth information in this research is not measured by a camera but instead estimated using a monocular depth estimation technique [1] combined with the sparse point cloud generated from COLMAP. This research will be a combination of sorts, using depth information from RGB-D cameras to directly supervise the training process of the Gaussian Splat.

## 3 Methodology

The original 3DGS model will be compared with the proposed model that incorporates depth information, 3DGSw/Depth. The original model will use gradient descent on the loss function L (Figure 2) as described in the original paper [5] to create a Gaussian Splat from a set of images provided by the Red-wood dataset [3]. The proposed model will use these same images with their corresponding depth maps to train a Gaussian Splat. The gradient descent procedure for the proposed model will use a combination of the loss function L and a proposed depth loss function that is derived from the dissimilarity between the ground truth depth map and the depth map that is rendered from the Gaussian splat.

$$\mathcal{L} = (1 - \lambda)\mathcal{L}_1 + \lambda\mathcal{L}_{\text{D-SSIM}}$$

Figure 2: The loss function from the original paper [5]

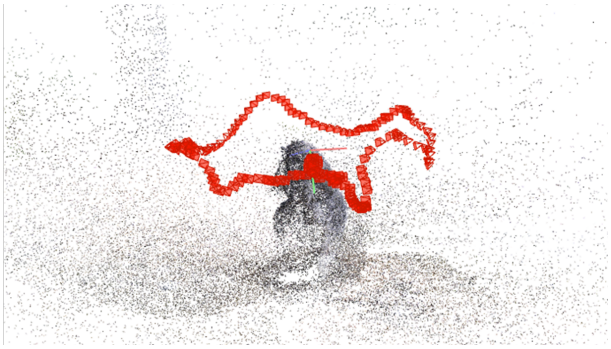

Figure 3: The original RGB-D images are taken from similar positions.

### 3.1 Supervising depth during Gaussian Splatting

The 3DGSw/Depth model contains extra depth functionality that is true to the original Gaussian Splat model in terms of operation. During training, the model will render both the image and a depth map (Figure 4) from the Gaussian splats. The depth map will be rendered by splatting the depth of every Gaussian on the image, similarly to how regular views are rendered from a Gaussian Splat. The depth of every Gaussian can be reused from the original model, as 3DGS already keeps track of their depths in order to determine the render order. The resulting depth map will be a black and white image, where lighter pixels correspond to a larger distance from the camera (Figure 4).

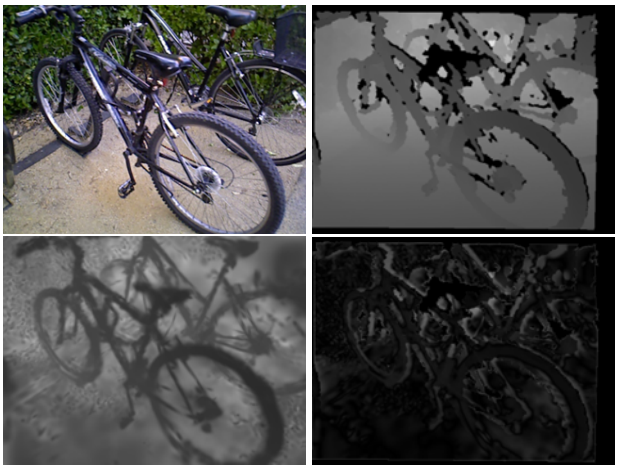

Figure 4: The depth-supervised Gaussian splatting pipeline represented in images. From top-left to bottom-right: The RGB image, the depth map from the camera, the depth map rendered from the Gaussian splat, and the depth loss map.

The rendered depth map and ground truth depth map will be preprocessed before the loss can be calculated. Firstly, the ground truth depth map (Figure 4) does not contain data for every pixel as RGB-D cameras are not perfect (Figure 5). Most RGB-D cameras use stereo-vision, which is unable to estimate the depth of objects that are only visible to one of the cameras. RGB-D cameras also have difficulty with large planes of just one color, especially in outdoor lighting when

the infrared dot patterns are not visible to the camera. RGB-D cameras also have problems estimating the depth of the sky. The parts of the depth maps that do not have data are completely ignored, neither counting as wrong nor correct. To achieve this, both the ground truth and rendered depth maps will be masked so pixels with no ground truth value are removed (Figure 6).

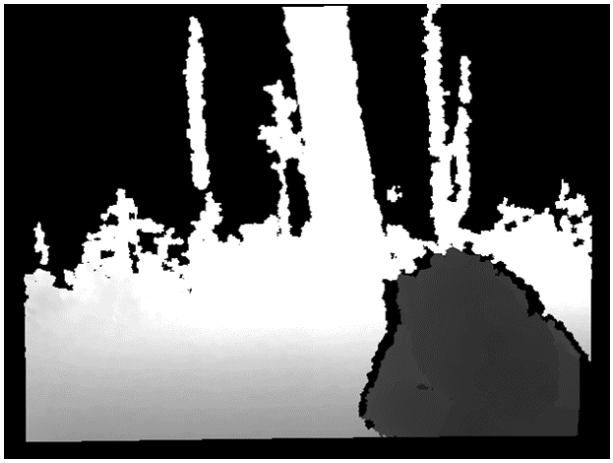

Figure 5: RGB-D images do not contain depth values for every pixel, such as right next to objects or in the sky.

The values of the rendered depth map will be on a different range than of the original depth map, as Structure from Motion and Gaussian Splatting do not maintain real-world distances. However, both depth maps are 0 for objects that are at 0 distance from the camera, and both depth ranges are linear, meaning that a depth value is twice as high when something is twice as far away. As such, the depth ranges only differ by a scaling factor and the depth maps can be compared after dividing both by their own means (Figure 6).

The depth loss will be equal to the mean of the absolute difference between the ground truth and the rendered depth map (Figure 6). Similarity measures that take into account structure, such as Structural Similarity Index Measure (SSIM), are unusable in this case where there are black spots in the depth map. The total loss will be a weighted average of the depth loss and the original loss function.

```
# Filter pixels for which there is no depth data
mask = gt_depth > 0
gt_depth = gt_depth[mask]
rendered_depth = rendered_depth[mask]

# Divide both maps by their own means
gt_depth = gt_depth / gt_depth.mean()
rendered_depth = rendered_depth
    / rendered_depth.mean()

# Calculate the depth loss
diff = gt_depth - rendered_depth
depth_loss = torch.abs(diff).mean() / 2

# Compute a weighted average of the original
# loss and the depth loss
final_loss = depth_weight * depth_loss +
    (1.0 - depth_weight) * original_loss
```

Figure 6: The Python code used to calculate the depth loss between the ground truth and rendered depth maps.

## 4 Results

### 4.1 Experimental setup

For this experiment, scans from the Red-Wood dataset [3] in the 'Sculpture' category were used. Although the type of object does not matter for this research, the 'Sculpture' category was chosen because of its diversity; it contains 458 object scans of varying shapes in both indoor and outdoor scenes. To keep the sample size manageable, this research only uses the first 96 object scans starting with the lowest object id's in the dataset. The scans in the dataset could contain thousands of frames, and the depth maps were sometimes shot at a slightly different framerate than the RGB frames. For this research, all object scans were subsampled to 200 frames by selecting frames at regular intervals, effectively retaining every nth frame. COLMAP was then used to create a sparse point cloud from these 200 images. The COLMAP procedure then kept the minimal amount of images necessary for a reconstruction, resulting in some subset of less than 200 images.

The Gaussian splat was trained on these images and the sparse point cloud using the *–eval* flag included in the original Gaussian Splatting repository, which split the images into a training and test set. Every object scan was used for the training of two Gaussian Splats: one of them uses the loss function from the original 3DGS paper, and the other is trained with a loss function that incorporates the depth loss with weight 0.5, which will be called 3DGSw/Depth. This weight was chosen after training several models with a range of weights from 0 to 1.0. Weights between 0.25 and 0.75 drastically decreased the depth loss while barely impacting the RGB loss, so the centrally positioned weight of 0.5 was chosen (Figure 7). The Gaussian Splats were trained in 2000 iterations to keep the computing time and resources manageable. During the training procedure, the depth and RGB losses were tracked in a csv file. After the training procedure, the similarity between the RGB renders and RGB ground truth was calculated using the SSIM, Peak Signal-to-Noise Ratio (PSNR), and Learned Perceptual Image Patch Similarity (LPIPS) similarity metrics. Finally, the Gaussian Splats were visually inspected to gain a deeper understanding of the effects that the depth information has on the training of the model.

### 4.2 Metrical results

Figure 8 shows that introducing a depth loss factor greatly reduces the depth loss while only slightly increasing the RGB loss. This shows that a local optimum of a combination of these functions lies closely to a local optimum of the original loss function, which suggests that the depth loss and original loss functions are not in conflict with each other. The

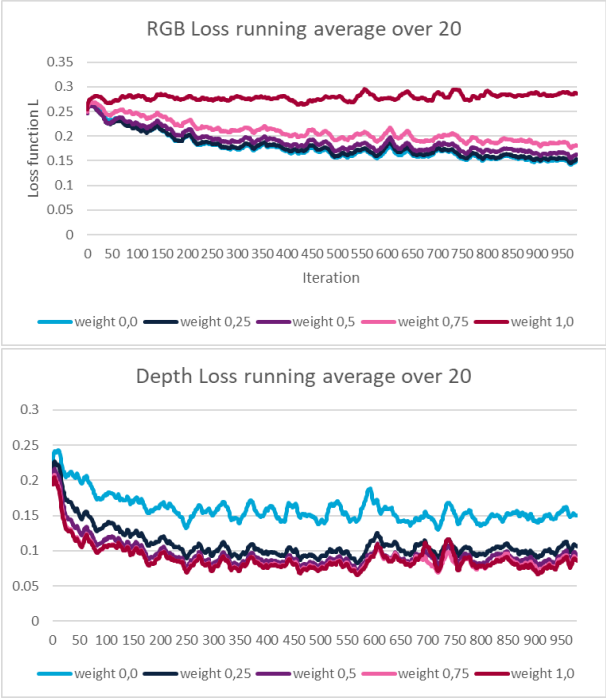


Figure 7: Loss on RGB images (top) and depth maps (bottom) during training of a Gaussian Splat on the bicycle scene, per weight factor of depth loss (20-iteration moving average).

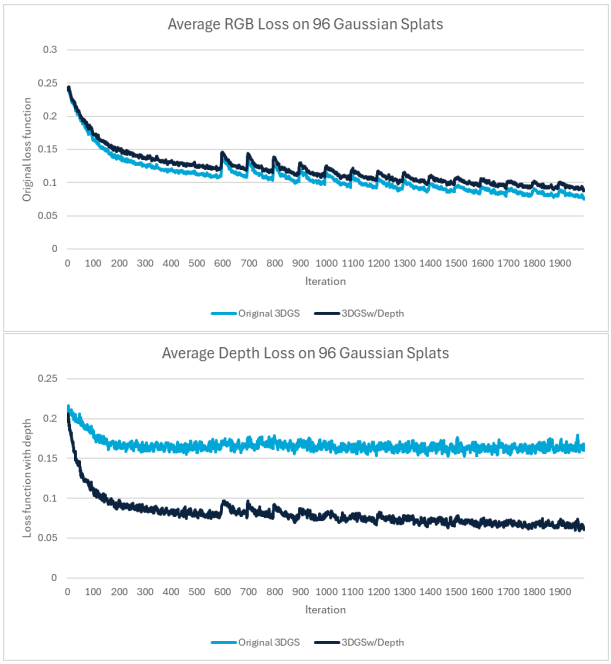


Figure 8: Loss on RGB images (top) and depth maps (bottom) during training of 192 Gaussian Splats on 96 objects with depth weights 0.0 and 0.5. The spikes that occur with intervals of 100 iterations are caused by densification, which is the splitting and cloning of Gaussians during 3DGS [5].

figure additionally shows that the original 3DGS model does optimize the depth, even without explicitly accounting for a depth loss in its loss function. However, this depth loss seems to stagnate after 200 iterations, at a loss value that is around three times higher than with 3DGSw/Depth.

Figure 9 shows the average SSIM, PSNR, and LPIPS scores of 3DGS and 3DGSw/Depth on a test set of RGB images for every of the 96 objects. A higher similarity is indiciated by a higher SSIM, a higher PSNR, and a lower LPIPS, which means that the original 3DGS outperformed 3DGSw/Depth with every metric. This could indicate that 3DGSw/Depth does not reduce overfitting. However, the objects were not scanned systematically which introduced a huge dependency on the camera positions, as can be seen in Figure 3. As such, the test set is similar to the training set and might not be able to account for overfitting.

| | 3DGS | 3DGSw/Depth |
|---|---|---|
| SSIM | 0.764 | 0.743 |
| PSNR | 22.51 | 21.52 |
| LPIPS | 0.345 | 0.377 |

Figure 9: The average SSIM, PSNR, and LPIPS on 96 Gaussian Splats trained using the original 3DGS and 96 that include a depth factor.

### 4.3 Visual results

Because the quantitative analysis is unable to capture completely how depth-supervision affects 3DGS, the resulting 3D models are visually compared for both 3DGS and 3DGSw/Depth to gain a deeper understanding of 3DGSw/Depth. All trained models are available upon request as a 7.5 GB Zip-file, this section will highlight views from several models that conjointly explain the differences between 3DGSw/Depth and 3DGS.

On indoor scenes such as in figure 10 and 11, 3DGSw/Depth slightly reduces the amount of floating Gaussians in the scene. However, the amount of floating artifacts remains high as the objects were filmed from especially close by, which prevented the RGB-D cameras from gaining a lot of depth data due to the pitfalls of stereo vision (Figure 5).

Scenes that are decently covered by the camera positions benefit greatly from the additional depth-supervision. Figure 12 is of high quality with 3DGS and depth-supervision reduced the semi-transparent floating artifacts. Especially outdoor scenes such as in figures 13 and 14 seem to benefit, which might be caused by the more extreme difference in depth compared to indoor scenes. 3DGSw/Depth removed several floating Gaussians in figure 14 and the depth-supervision improved figure 13 from near-perfect to perfect by removing black Gaussians that were floating in the sky.

In some cases, 3DGSw/Depth had a negative impact on the scene. This was caused by a mismatch between the RGB and depth data in those particular models, resulting in slightly different camera positions. In figure 15, the house in the distance was pulled closer to the cameras because of this mismatch. Figure 16 shows the incorrect depth loss map that resulted from a mismatched ground truth and render.

Figure 10: An indoor scene with mediocre camera coverage contains slightly less floating Gaussians when trained according to 3DGSw/Depth. Redwood scan 1312 [3], 3DGS on the left and 3DGSw/Depth on the right.

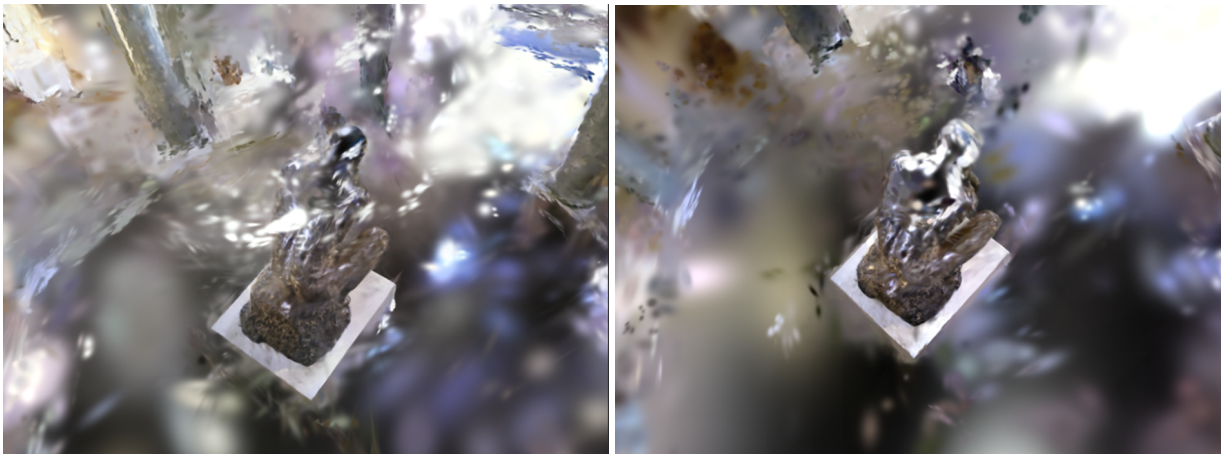

Figure 11: An indoor scene with bad camera coverage contains slightly less floating Gaussians when trained according to 3DGSw/Depth. Redwood scan 1313 [3], 3DGS on the left and 3DGSw/Depth on the right.

Figure 12: An indoor scene with good camera coverage contains slightly less floating Gaussians when trained according to 3DGSw/Depth. Redwood scan 1325 [3], 3DGS on the left and 3DGSw/Depth on the right.

Figure 13: An outdoor scene with good camera coverage contains few floating Gaussians with 3DGS but no floating Gaussians when trained according to 3DGSw/Depth. Redwood scan 2159 [3], 3DGS on the left and 3DGSw/Depth on the right.

Figure 14: An outdoor scene trained with 3DGSw/Depth contains less floating Gaussians than a model trained with 3DGS. Redwood scan 6380 [3], 3DGS on the left and 3DGSw/Depth on the right.

Figure 15: An outdoor scene with mediocre camera coverage is decent when trained with 3DGS but almost unrecognizable when trained with 3DGSw/Depth. Redwood scan 2172 [3], 3DGS on the left and 3DGSw/Depth on the right.

Figure 16: The RGB image and depth map were shot from slightly different camera positions, resulting in a high depth loss. From left to right: The ground truth depth, the rendered depth, and the depth loss map.

# 5 Responsible Research

To make this research reproducible, the experimental setup has been described in great detail. The paper describes the exact object scans that were used, together with a reference to the dataset. The steps taken to preprocess the data and train the Gaussian Splats are exhaustively described, together with the commands used, so that the reader will be able to reproduce every step exactly.

Furthermore, this paper has been carefully written to minimize the bias. For example, results that did not align with the rest of the results were not just left out. Instead, these results were prompts to rethink and explain the limitations, such as can be seen in the text regarding figure 9. When one of the models showed how 3DGSw/Depth performed worse than 3DGS (Figure 15), this was not left out but used as an opportunity for further research to build upon.

This research has been made verifiable by providing access to data such as the trained Gaussian Splats, which allows anyone to move around in the models and verify the correctness of the renders used in this paper, as well as check that the used models accurately represent the various other models. Since the data is several gigabytes, they are too big to be hosted indefinitely. As such, they are stored locally and available upon request. To accommodate for this, my email has been provided on the title page of this paper.

# 6 Conclusion and Discussion

## 6.1 Conclusion

This research investigated how additional depth information from RGB-D cameras can be exploited to enhance 3DGS and found that incorporating depth data significantly enhances the model performance by reducing overfitting artifacts. This was done by rendering depth maps from the Gaussian Splat during the training process, comparing this render with the ground truth from the RGB-D camera, and factoring in this dissimilarity as a loss term for the gradient descent procedure that Gaussian Splats are trained with. The results showed that this technique drastically reduces the depth loss without substantially increasing the loss on the RGB images, indicating that these function complement each other. Furthermore, this research provided a visual comparison of renders from Gaussian Splats that were trained with and without depth-supervision that showed how depth-supervision reduces floating artifacts, especially in outdoor scenes. However, depth-supervision could have a big negative impact on 3DGS when the provided depth data is incorrect or does not align with the RGB images.

## 6.2 Limitations

While depth-supervision does seem to be able to reduce overfitting artifacts in Gaussian splats, there are limitations. Firstly, this research provides a qualitative comparison, which might be subjective and might not generalize well. Secondly, the dataset used in this paper consists of object scans which are RGBD videos shot while moving around some object. Object scans are just one of the use cases of 3DGS, which makes the research hard to generalize to different kinds of scenes. One of the bigger limitations was the lack of diversity in camera positions, making it hard to split the data into a training set and test set such that the level of overfitting could be measured. This caused lower similarity scores for 3DGSw/Depth even though the visual analysis showed less overfitting.

## 6.3 Future Research

There are numerous ways in which this research could be expanded. This research showed that depth data and RGB data could negatively impact a Gaussian Splat when they are slightly misaligned. Further research could work on this problem by using COLMAP to calculate separate camera positions for the RGB and the depth data, in order to evaluate the depth and RGB renders in the Gaussian Splat from their own positions. Additionally, the system presented in this research could be evaluated more thoroughly by using datasets with diverse camera positions. This would allow more independence between the training and test set, so that an accurate quantitative review of both systems can be presented.

# References


[1] Shariq Farooq Bhat, Reiner Birkl, Diana Wofk, Peter Wonka, and Matthias Müller. Zoedepth: Zero-shot transfer by combining relative and metric depth, 2023.

[2] Nicholas Brunetto, Nicola Fioraio, and Luigi Di Stefano. Interactive rgb-d slam on mobile devices. *CVLab - Dept. of Computer Science and Engineering, University of Bologna*, 2024.

[3] Sungjoon Choi, Qian-Yi Zhou, Stephen Miller, and Vladlen Koltun. A large dataset of object scans. *arXiv:1602.02481*, 2016.

[4] Jaeyoung Chung, Jeongtaek Oh, and Kyoung Mu Lee. Depth-regularized optimization for 3d gaussian splatting in few-shot images, 2024.

[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), July 2023.

[6] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis, 2023.

[7] 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, 2020.

[8] Emmanouil Nikolakakis, Utkarsh Gupta, Jonathan Vengosh, Justin Bui, and Razvan Marinescu. Gaspct: Gaussian splatting for novel ct projection view synthesis, 2024.

[9] Johannes L. Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[10] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, David Mcallister, Justin Kerr, and Angjoo Kanazawa. Nerfstudio: A modular framework for neural radiance field development. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Proceedings*, SIGGRAPH '23. ACM, July 2023.