How can we reduce the effect of noise on 3D Gaussian Splats?

A Study on Deblurring and Recoloring Techniques to Enhance 3D Reconstructions

Tijmen Meijer
Supervisor(s): Xucong Zhang
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: Tijmen Meijer
Final project course: CSE3000 Research Project
Thesis committee: Dr. Xucong Zhang, Dr. Michael Weinmann

An electronic version of this thesis is available at http://repository.tudelft.nl/.

Abstract

Multi-view image recognition is crucial for numerous applications such as autonomous vehicles and robotics, where accurate 3D reconstructions from 2D images are essential. However, the presence of various noise factors like motion blur, variable lighting, and changes in the field of view can significantly degrade the quality of these reconstructions. Through a series of experiments using simulated data created in Blender, I explore the effectiveness of different de-noising methods, including AI-based and traditional image processing techniques. My findings indicate that specific adjustments in the Gaussian Splatting data preprocessing can significantly mitigate the adverse effects of noise, leading to more accurate and reliable 3D reconstructions. The study contributes to the field by providing a detailed analysis of noise impact and proposing viable solutions to enhance the fidelity of multi-view image reconstructions in noisy environments. The code and resources developed for this project are available in the GS-preprocessor repository on GitHub. [1]

1. Introduction

Multi-view image recognition is pivotal in advancing fields such as autonomous navigation and robotics, where it enables the creation of three-dimensional models from multiple two-dimensional images. This technology is essential for machines to understand and interact with the physical world more effectively. However, the accuracy and reliability of these models are often compromised by various forms of noise, such as motion blur, lighting changes, and field of view variations encountered in real-world scenarios. Addressing these issues is crucial for enhancing the performance and dependability of 3D reconstructions, which are fundamental to applications in autonomous vehicles and robotic manipulation.

While previous research has extensively explored Gaussian Splatting techniques, recognized for their ability to handle image distortions and improve 3D image reconstructions, there remains a significant research gap. Notable studies, such as those on motion blur restoration [3] and color consistency [14], have demonstrated the effectiveness of these methods in reducing artifacts caused by camera shake and lighting inconsistencies. However, the specific impact of different types of noise, particularly motion blur, variable lighting, and field of view changes on the accuracy of 3D reconstructions, has not been sufficiently addressed. This gap persists despite its critical relevance to practical applications in dynamic environments [12].

This paper seeks to address the critical unanswered questions regarding the influence of noise on the effectiveness of Gaussian Splatting in multi-view image recognition. Specifically, it investigates: (1) the effect of motion blur on the accuracy of 3D reconstructions, and (2) the impact of changing light conditions on recognition capabilities. This research systematically investigates these aspects, providing empirical evidence on how these noise factors affect the reconstruction accuracy.

[1] https://github.com/surftijmen/GS-preprocessor

2. Related Works

The field of 3D Gaussian splatting has seen significant advancements, particularly in addressing challenges related to deblurring and color consistency. These advancements are crucial for achieving high-quality real-time rendering and accurate scene reconstruction. In this section, I review several state-of-the-art methods that contribute to the development of effective deblurring frameworks, robust color correction techniques, and foundational concepts essential for 3D Gaussian splatting [6].

2.1. Image Deblurring Techniques

Deblurring 3D Gaussian Splatting[8] introduces a deblurring framework that utilizes a differentiable rasterizer and 3D Gaussians. This method employs a multi-layer perceptron (MLP) to adjust the parameters of 3D Gaussians, allowing for real-time sharp image rendering from blurry inputs . Similarly, *DeblurGS: Gaussian Splatting for Camera Motion Blur* [11] focuses on optimizing sharp 3D Gaussian splatting from motion-blurred images by integrating auxiliary parameters that simulate the blur operation into the Gaussian splatting pipeline. The *BAD-Gaussians* [15] method leverages explicit Gaussian representation and handles severe motion-blurred images with inaccurate camera poses to achieve high-quality scene reconstruction. This approach models the physical image formation process of motion-blurred images and jointly optimizes the Gaussians and camera trajectories, enabling superior rendering quality and real-time capabilities .

2.2. Color Consistency and Correction

Color consistency and correction are crucial aspects of 3D Gaussian splatting. The method described in *Efficient and Robust Color Constancy Using Gray Pixels*[4] addresses lighting inconsistencies in images by using gray pixels to achieve robust color constancy under various lighting conditions, which is vital for maintaining color fidelity in 3D Gaussian splatting . Ensuring color consistency across multiple images is tackled by the technique in *Improving Color Consistency across Multiple Images via Global Optimization on the Color Remapping Curves*[9], which provides a robust solution through global optimization, significantly enhancing the visual quality of the rendered images . Additionally, *Color Correction by Gamma Intensity Transfor-*

mation[10] offers an effective method for correcting color inconsistencies caused by various imaging conditions, useful for pre-processing images to ensure uniform color distribution before applying Gaussian splatting .

2.3. Foundational Concepts and AI-Based Methods

The foundational concepts of Gaussian splatting are thoroughly detailed in *3D Gaussian Splatting for Real-Time Radiance Field Rendering*[6], which explains the methodology behind Gaussian splatting and its applications in real-time rendering. For motion deblurring, the approach outlined in *Deep Multi-Scale Convolutional Neural Network for Dynamic Scene Deblurring*[7] presents a deep learning model that effectively handles motion blur in dynamic scenes, providing a significant improvement in image clarity.

3. Methodology

The methodology of this research involves generating and analyzing noise-affected images to understand the impact on the effectiveness of 3D Gaussian Splats in image recognition tasks. My method involves creating three distinct sets of images: normal (unaltered), noised (with added blur, or lighting variations), and denoised (noise-reduced versions). The latter image set is generated by applying my own denoise method to the blurred image set. This setup allows us to create controlled environments with varying degrees of blur and lighting. These image sets will be generated using 3D models rendered in Blender [2], which allows for precise control over artificial blur parameters and environmental conditions. The 3 image sets will then be converted into 3D Gaussian splats to perform analysis.

3.1. Noise Generation and Image Sets Construction

To generate the image sets, I used a python script within Blender to export 64 different camera angles of the corresponding object to images. This process allows to easily make adjustments to the 3D objects and export them in the correct format. Figure 1 presents one of the exported frames. I use the default Blender cube to test the deblurring methods, since this simple object can be easily compared for the analysis.

Motion Blur: Motion blur is simulated using Blender's *CompositorNodeBlur* to replicate camera shake. By adding this tree node with a custom blur size, 25 in my case, it is possible to generate the same set of images but all with a blurred filter applied. Motion blur is simulated by applying this in the x- and y direction.

Lighting Variations: Lighting conditions are altered in Blender to simulate different times of day or weather conditions. This includes changes in color temperature to test the robustness of the light improvement techniques under varying illumination. In Fig. 1 there is an example of the original blue cube with a strong yellow light applied, which makes it appear as a green.

Figure 1. Examples of generated results: original image, image with motion blur, image with (strong) lighting variation.

3.2. Deblurring Motion Blur

According to previous research [1, 11], there are several deblurring techniques to consider for minimizing the effect of motion blur. In this study, I evaluate three methods: Pylops Deblur, Wiener Deblur, and Deblur-GAN.

Pylops Deblur Pylops [13] is a Python library for large-scale optimization problems, which includes tools for image deblurring. The Pylops Deblur method uses iterative algorithms to solve the inverse problem of motion blur. It formulates the deblurring process as a linear inverse problem and applies regularization techniques to stabilize the solution.

Wiener Deblur The Wiener Deblur [1] method utilizes the Wiener filter, a classical approach to image deblurring that minimizes the mean square error between the estimated and true images. The Wiener filter operates in the frequency domain and requires knowledge of the power spectra of the noise and the original image. This method is effective for linear and stationary blur, but it unfortunately only works with gray scaled images.

Deblur-GAN Deblur-GAN[7] is an advanced deep learning approach that uses Generative Adversarial Networks (GANs) to restore sharp images from motion-blurred inputs. This method consists of two networks: a generator that attempts to produce a sharp image from a blurred input and a discriminator that tries to distinguish between real sharp images and the generator's output. The adversarial training process helps the generator learn to create more realistic deblurred images.

3.3. Deblurring Varying Light Conditions

To simulate the impact of different lighting conditions, such as those during sunset, multiple images of the same object are generated in Blender[2] with varied lighting hues .

These images are used to test various algorithms and Python scripts aimed at ensuring color consistency across the images. This part of the research seeks to understand how changes in light color influence the recognition rates in 3D Gaussian Splats and to develop methods to compensate for these variations effectively.

Numpy Mean Method This method standardizes the color profiles of images using PIL and NumPy[5]. It computes the mean color values of a reference image and applies this color tint to another image, ensuring that both images share the same color theme. This normalization of color tones is essential for improving feature matching across views, reducing reconstruction artifacts, and enhancing the visual quality of the 3D model. Color consistency not only aids in achieving more reliable reconstructions but also enhances the applicability of Gaussian splatting in fields where high-quality visual outputs are required, such as in virtual reality and digital heritage preservation. Given an original image with mean color values μ_{original} and a recolored image with mean color values $\mu_{\text{recolored}}$, the output image array I_{output} is computed as:

$$I_{\text{output}} = I_{\text{recolored}} \cdot \frac{\mu_{\text{original}}}{\mu_{\text{recolored}}}, \tag{1}$$

where $I_{\text{recolored}}$ is the array of the recolored image and $\frac{\mu_{\text{original}}}{\mu_{\text{recolored}}}$ is the ratio of the mean color values of the original and recolored images.

Colormatcher The Colormatcher[14] method uses the ColorMatcher library to transfer the color profile from a reference image to other images. This script loads a reference image and applies its color profile to all images in the specified directory, using the Monge-Kantorovitch Linear (MKL) method for color transfer. This method transfers the color characteristics from a reference image to a target image by solving an optimal transport problem, which minimizes the cost of transporting the color distribution of the target image to match that of the reference image [4]. The goal is to find a transport map T that minimizes the cost function $c(x, y)$, typically the Euclidean distance between color vectors. The optimal transport problem is defined as:

$$T^* = \arg\min_T \int_X c(x, T(x)) d\mu(x). \tag{2}$$

In this context, T represents the transport map I aim to determine, $c(x, y)$ denotes the cost function, μ is the color distribution measure of the target image, and ν is the color distribution measure of the reference image.

Generally, the objective is to adjust the colors in the target image so that its color distribution becomes as close as possible to the color distribution of the reference image, minimizing the total "cost" of this adjustment.

3.4. Preprocessing

With these findings I introduce the *GS-preprocessor*, a preprocessing noise removal method to optimize Gaussian Splatting blurred models. Given blurry multi-view observations, my goal is to restore a sharp 3D scene using the previously mentioned methods.

4. Experiment

In this section, I describe the experimental setup so that the research can be reproduced. I will also explain the evaluation methods used to assess the effectiveness of various deblurring and recoloring techniques.

4.1. Experiment Setup

The experiment is setup in two stages. Initially, synthetic data is used, specifically the cubes generated in Blender as mentioned before, to test the effectiveness of the different deblurring and recoloring methods. This controlled environment allows for precise manipulation of variables such as motion blur and lighting conditions. Following the simple synthetic object tests, the methods are applied to more realistic (but still synthetic) data to evaluate their performance in practical scenarios. This consists of a sequence of images taken from a 3D chair object with first a motion blurred camera and then with a point light which changes its color from white to red over time.

4.2. Evaluation Methods

To compare the effectiveness of different recoloring methods, I utilized three metrics on the images: Delta E, Structural Similarity Index (SSIM), and Peak Signal-to-Noise Ratio (PSNR). These metrics provide a comprehensive evaluation of color accuracy, structural similarity, and image quality, respectively. I then use the Cumulative Distribution Function (CDF) so I can compare the performance of the deblur methods within the resulting gaussian splats.

Delta E (CIE76) Delta E (CIE76) is used to measure the color difference between the original and recolored images in the LAB color space. It is calculated using the following formula:

$$\Delta E_{76} = \sqrt{(L_2 - L_1)^2 + (a_2 - a_1)^2 + (b_2 - b_1)^2}, \tag{3}$$

where L_1, a_1, b_1 are the LAB components of the original color and L_2, a_2, b_2 are the LAB components of the recolored color. This formula computes the root of the sum of the squared differences in lightness (L) and color opponent dimensions (a and b), providing a numerical measure of perceptual color distance.

Structural Similarity Index (SSIM) The Structural Similarity Index (SSIM) measures the similarity between two images. The formula for SSIM is:

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}, \quad (4)$$

where μ is the average, σ^2 is the variance, σ_{xy} is the covariance of x and y, $C_1 = (0.01L)^2$, $C_2 = (0.03L)^2$, and L is the dynamic range of the pixel values. This formula will be used to estimate the performance of the recoloring algorithm, and the values will be used to compare which method works best.

Peak Signal-to-Noise Ratio (PSNR) Peak Signal-to-Noise Ratio (PSNR) is used to measure the quality of the reconstructed image compared to the original. It is calculated using the following formula:

$$\text{PSNR} = 20 \log_{10} \left(\frac{\text{MAX}_I}{\sqrt{\text{MSE}}} \right). \quad (5)$$

The Mean Squared Error (MSE) is given by:

$$\text{MSE} = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i,j) - K(i,j)]^2, \quad (6)$$

where I is the original image, K is the compressed image, and m and n are the dimensions of the images.

Cumulative Distribution Function (CDF) The Cumulative Distribution Function (CDF) is used to analyse the distribution of distances between the points in the blurred, deblurred, and original point clouds. The CDF, $F(x)$, for a given distance x, is defined as the probability that a point in the deblurred point cloud is within distance x of a point in the original point cloud. Mathematically, the CDF is expressed as:

$$F(x) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}(d_i \leq x), \quad (7)$$

where d_i is the distance between the i-th point in the deblurred point cloud and its nearest neighbor in the original point cloud, N is the total number of points, and $\mathbb{1}$ is the indicator function, which is 1 if $d_i \leq x$ and 0 otherwise.

By comparing the CDFs of the blurred and deblurred point clouds with the original point cloud, I can quantitatively assess the effectiveness of the deblurring algorithm. A CDF curve that is closer to the upper-left corner of the plot indicates better performance, as it shows that a higher proportion of points in the deblurred point cloud are closer to their corresponding points in the original cloud.

5. Results and Discussion

The results after applying both methods to remove noise are presented in this section. The original, blurred and deblurred images together with the corresponding gaussian splats are firstly shown for motion blur. After that there is a statistical analysis for the recoloring method. These results will be evaluated and compared against each other.

5.1. Motion Blur

Figure 2 presents the results from the 3 deblurring methods. Both Wiener deblur and PyLops deblur only give relevant results with gray scaled images.

Figure 2. (a) Wiener deblur, (b) PyLops deblur, (c) Deblur-GAN.

The Wiener deblur method [1] approach slightly reduces the effect of motion blur but introduces unwanted vertical lines as shown in the red square in the first image in Fig. 3. This side effect makes the algorithm difficult to use properly for removing blur, since it creates a new type of noise.

The Pylops deblur method performs somewhat better, almost fully reducing the motion blur effect. However, it struggles with deblurring corners, altering the shape of objects as highlighted in the red areas of the second image in Fig. 3. Although this side effect is minimal, the visual difference in the corners alters the object too much to be an effective noise reduction method for Gaussian splatting. This method is further explained in appendix B.

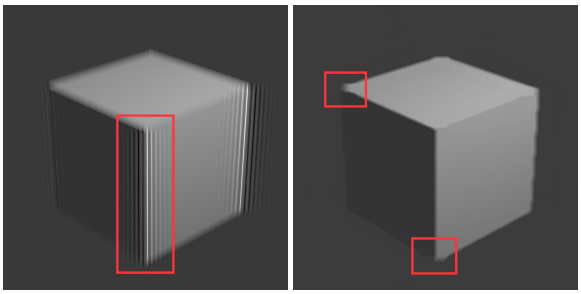

Figure 3. Wiener- and PyLops deblur error indication. Wiener deblur introduces unwanted vertical lines, PyLops struggles with deblurring corners.

The effectiveness of deblur-GAN in removing motion blur is clearly demonstrated in Fig. 4, presented together with the original blurred version. Both of these point clouds

have the same input dataset with 64 images. On the right picture I have applied the Deblur-GAN algorithm to deblur the input dataset. In the Appendix in Fig. 9 there are more results of deblurred realistic objects.

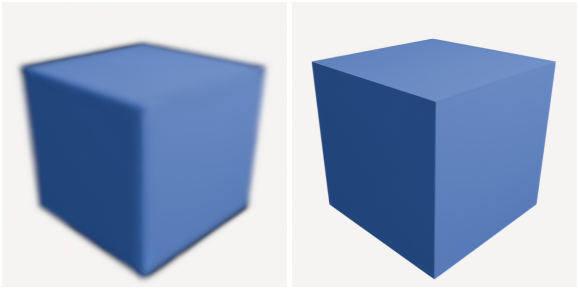

Figure 4. Gaussian splatting point clouds: (a) Blurred image set, (b) Deblurred image set with Deblur-GAN.

The processed 3D Gaussian splats are significantly sharper compared to the blurred image set. This method restores most of the details lost due to motion blur, which is evident when comparing the deblurred images to the original blurred set.

To effectively compare the Deblur-GAN result with the original point cloud, there is an illustrated CDF in Fig. 5 of the distances from the points in the blurred and deblurred point clouds to their nearest neighbors in the original point cloud. The blue curve representing the deblurred point cloud. The deblurred point cloud exhibits a notably improved distribution of distances to the original point cloud, with most points being closer to the original than those in the blurred point cloud. This shift in the CDF curve towards the origin for the deblurred point cloud indicates that the deblurring method effectively reduces noise and artifacts, restoring the point cloud more accurately to its original structure.

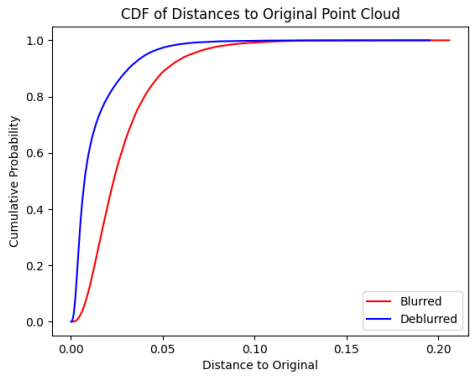

Figure 5. Cumulative Distribution Function (CDF) of distances to the original point cloud. The red curve represents the blurred point cloud, while the blue curve represents the deblurred point cloud.

Visually comparing the deblurred versions to the original one further underscores the effectiveness of this method in preserving image quality, making it the superior motion deblurring technique.

5.2. Varying Light Conditions

The quantitative comparison of the recoloring methods on the default cube using Delta E, SSIM, and PSNR metrics is summarized in Tab. 1. The visualization of these results are displayed in Fig. 6.

Method	ΔE	SSIM	PSNR
colored	27.24	0.94	15.49 dB
numpy	4.03	0.99	33.66 dB
color matcher	7.04	0.89	28.31 dB

Table 1. Quantitative comparison of recoloring methods, which indicate that both the numpy method and the Color matcher method result in an improved recoloring of the image.

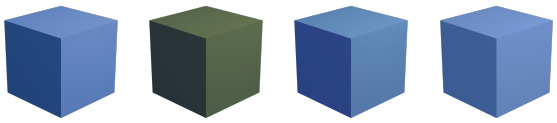

Figure 6. Examples of generated images of a synthetic cube under varying light conditions: (a) Original image, (b) Input image with altered lighting, (c) Image recolored using NumPy method, (d) Image recolored using Color Matcher method.

The results of applying the same comparison method on the realistic chair model are shown in Tab. 2 and Fig. 7.

Method	ΔE	SSIM	PSNR
colored	1.46	0.995	36.08 dB
numpy	1.23	0.97	37.40 dB
color matcher	0.55	0.997	39.21 dB

Table 2. Quantitative comparison of recoloring methods for the chair object.

Figure 7. Examples of generated images of a synthetic chair under varying light conditions: (a) Original image, (b) Input image with altered lighting, (c) Image recolored using NumPy method, (d) Image recolored using Color Matcher method.

Since it might be challenging to notice the differences between the objects in the figures above, refer to the zoomed-in versions of these images provided in the appendix for a more detailed view in C.

The quantitative analysis using Delta E, SSIM, and PSNR metrics reveals a positive performance. As intended for both objects, the colored version shows a poor comparison to the original image since they have a relatively high Delta E. For the synthetic cube, the NumPy method shows superior color accuracy and structural similarity, as evidenced by a low Delta E of 4.03, high SSIM of 0.99, and PSNR of 33.66 dB. For the realistic chair model however, all methods improve, with the color matcher method achieving the best results: a Delta E of 0.55, SSIM of 0.997, and PSNR of 39.21 dB. Visual comparisons in Figures 5 and 6 support these metrics, demonstrating the color matcher's superior ability to handle lighting variations while preserving image fidelity. These findings highlight the importance of choosing advanced recoloring methods for applications requiring high color accuracy and image quality under diverse lighting conditions. In the appendix in Fig. 11 the effect of applying the color matcher algorithm is presented after applying a red light on the chair object. In this figure you can clearly see the positive impact of applying this method.

6. Limitations & Future Works

This section discusses the limitations encountered in this study and proposes future research directions to address these challenges and improve the robustness and effectiveness of multi-view image reconstructions using 3D Gaussian Splats.

6.1. Limitations

While the methods and results presented in this study show significant improvements in handling motion blur and lighting variations in multi-view image reconstructions, there are several limitations that need to be acknowledged.

Motion Blur Reconstruction Despite the effectiveness of the deblurring techniques evaluated, motion blur cannot be fully reconstructed. The inherent loss of information during the motion blur process means that some details are irretrievably lost. Even advanced methods like DeblurGAN [7] can only approximate the original image based on learned patterns, which might not always represent the actual scene accurately.

AI-based Deblurring The use of AI-based deblurring methods, such as Deblur-GAN, relies on training data to predict and restore blurred images. This introduces a limitation as the restored image is not the actual data but a plausible reconstruction based on the model's training. The performance of these models heavily depends on the quality and diversity of the training data, which might not cover all possible real-world scenarios.

Recoloring Imperfections Recoloring methods, including the Numpy Mean Method and Colormatcher, aim to standardize color profiles across images. However, these methods are not always perfect and can sometimes introduce artifacts or fail to achieve exact color consistency. Factors such as varying lighting conditions and the intrinsic color properties of the objects being imaged can affect the efficacy of these recoloring techniques [12, 14].

6.2. Future Works

Future research can address these limitations through several avenues:

Enhanced Motion Blur Models Developing more sophisticated models that can better estimate and restore lost information due to motion blur is a crucial area for future research. This could involve hybrid approaches that combine traditional deblurring techniques with AI-based models to leverage the strengths of both.

Training Data Diversification For AI-based deblurring methods, expanding the diversity and quality of training datasets can help improve model performance. This includes creating more advanced datasets that capture a wider range of motion blur scenarios and lighting conditions.

Advanced Recoloring Techniques Future work can also focus on developing more advanced recoloring techniques that minimize artifacts and achieve better color consistency. This could involve creating machine learning models specifically trained for color correction tasks under varying lighting conditions.

Real-world Application and Testing Extending the testing of these methods to more real-world scenarios will provide further insights into their robustness and practical applicability. This includes testing on datasets from different domains such as medical imaging, autonomous driving, and surveillance systems to validate and refine the techniques.

Integration with Real-time Systems Another important direction is the integration of these deblurring and recoloring techniques into real-time systems, such as autonomous navigation and robotic vision systems. This requires optimising the computational efficiency of the algorithms to meet the demands of real-time processing.

7. Responsible Research

This section considers the reproducibility and integrity of the conducted research. The proposed method avoids using sensitive information, and all used data is completely synthetic.

7.1. Ethical Considerations

In conducting this research, all experiments were designed and executed with a commitment to ethical standards. The synthetic data generated for this study using Blender ensures that no real-world subjects were involved, thereby eliminating any ethical concerns related to privacy or consent.

7.2. Data Transparency

All data used in this research, including the synthetic images and the generated 3D models, are made publicly available. The datasets for reproducing the results can be accessed at the GS-preprocessor GitHub repository[2]. This transparency allows other researchers to validate and use the corresponding data.

7.3. Reproducibility

I have taken significant steps to ensure the reproducibility of my findings. Detailed descriptions of the methodologies, along with the scripts used for generating and processing the images, are provided. By sharing my code and data, I aim to enable others to replicate my experiments and verify the robustness of my results. The code for reproducing the results can be accessed at previously mentioned GitHub repository[3]. This transparency allows other researchers to validate and reproduce my work. The code base makes use of the Deblur-GAN algorithm which is available at the following repository [4]. The corresponding training data can also be found in this repository. The code used for generating the PyLops deblurred results can be found in this repository [5].

7.4. Integrity

Every aspect of the research, from data collection to analysis and interpretation, is conducted with utmost honesty and accuracy. Clear documentation of methodologies ensures that findings are based on reliable processes.

Engaging in peer review processes has further enhances integrity by subjecting the research to scrutiny from fellow experts in the field. Feedback received through peer review has helped in identifying potential weaknesses and strengthens the overall robustness of the study.

[2] https://github.com/surftijmen/GS-preprocessor

[3] https://github.com/surftijmen/GS-preprocessor

[4] https://github.com/KupynOrest/DeblurGAN

[5] https://pylops.readthedocs.io/en/stable/tutorials/deblurring.html

8. Conclusion

This research delves into the critical impact of noise on the effectiveness of 3D Gaussian Splats in multi-view image recognition. By systematically analyzing the effects of motion blur and varying lighting conditions, the study offers valuable insights into optimizing Gaussian Splatting parameters to enhance the robustness of 3D reconstructions. Through experiments with synthetic data, several key findings emerge:

Motion Blur The application of advanced deblurring techniques, particularly Deblur-GAN, significantly improves the clarity of motion-blurred images. Despite the inherent limitations in fully reconstructing motion-blurred scenes, these methods demonstrate substantial potential in approximating the original details, thereby enhancing the accuracy of 3D reconstructions.

Lighting Variations The study outlines the efficacy of color consistency techniques, with the Numpy Mean Method and Colormatcher demonstrating notable improvements in maintaining color fidelity across varied lighting conditions. These techniques are crucial for ensuring reliable feature matching and high-quality 3D models.

With these findings, significant advancements have been made in optimizing Gaussian Splatting for multi-view image recognition. The integration of advanced deblurring methods and color consistency techniques enhances the robustness and accuracy of 3D reconstructions, making Gaussian Splatting a more effective tool in diverse environmental and motion conditions.

References

[1] Prodip Biswas, Abu Sufian Sarkar, and Mohammed Mynuddin. Deblurring images using a wiener filter. *International Journal of Computer Applications*, 109(7):36–38, 2015. 2, 4

[2] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. 2

[3] Shereen El-Shekheby, Rehab F. Abdel-Kader, and Fayez W. Zaki. Restoration of spatially-varying motion-blurred images. In *2018 13th International Conference on Computer Engineering and Systems (ICCES)*, pages 595–600, 2018. 1

[4] Christopher Hahne and Amar Aggoun. Plenopticam v1.0: A light-field imaging framework. *IEEE Transactions on Image Processing*, 30:6757–6771, 2021. 1, 3

[5] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández

del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, 2020. 3

[6] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42 (4), 2023. 1, 2

[7] Orest Kupyn, Volodymyr Budzan, Mykola Mykhailych, Dmytro Mishkin, and Jiří Matas. Deblurgan: Blind motion deblurring using conditional adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2, 6

[8] Byeonghyeon Lee, Howoong Lee, Xiangyu Sun, Usman Ali, and Eunbyung Park. Deblurring 3d gaussian splatting, 2024. 1

[9] Yinxuan Li, Li Li, Jian Yao, Menghan Xia, and Hanyun Wang. Contrast-aware color consistency correction for multiple images. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 15:4941–4955, 2022. 1

[10] Yunmeng Li, Yinxuan Li, Jian Yao, Ye Gong, and Li Li. Global color consistency correction for large-scale images in 3-d reconstruction. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 15:3074–3088, 2022. 2

[11] Jeongtaek Oh, Jaeyoung Chung, Dongwoo Lee, and Kyoung Mu Lee. Deblurgs: Gaussian splatting for camera motion blur, 2024. 1, 2

[12] Jaesik Park, Yu-Wing Tai, Sudipta N. Sinha, and In So Kweon. Efficient and robust color consistency for community photo collections. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1, 6

[13] Matteo Ravasi and Ivan Vasconcelos. Pylops—a linear-operator python library for scalable algebra and optimization. *SoftwareX*, 11:100361, 2020. 2, 10

[14] Menghan Xia, Jian Yao, Renping Xie, Mi Zhang, and Jinsheng Xiao. Color consistency correction based on remapping optimization for image stitching. In *IEEE International Conference on Computer Vision Workshops (ICCVW)*, 2017. 1, 3, 6

[15] Lingzhe Zhao, Peng Wang, and Peidong Liu. Bad-gaussians: Bundle adjusted deblur gaussian splatting, 2024. 1

A. Recoloring real world data

The effect of applying the color matcher algorithm is seen in Fig. 8. [6][7]

Figure 8. Examples of generated images: (a) original image, (b) sample image, (c) recolored with colormatcher.

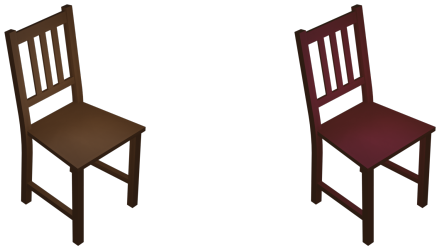

Figure 9. Recolored and lighted chair with red light.

Figure 10. Blurred and deblurred chair with Deblur-GAN

[6] https://www.universetoday.com/60072/what-is-a-moon/

[7] https://education.nationalgeographic.org/resource/moon/

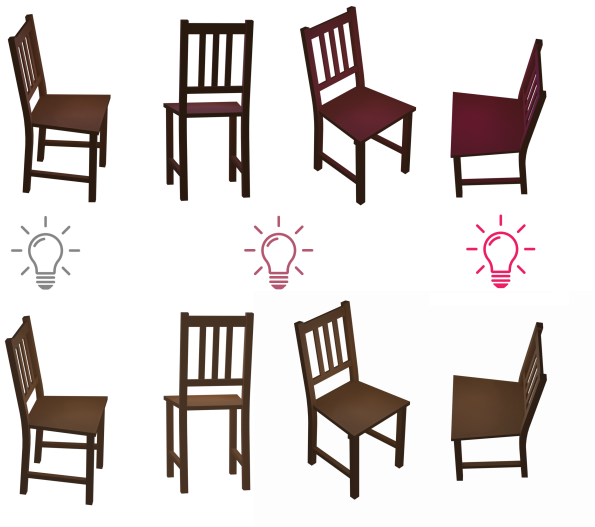

Figure 11. The effect of applying the Color Matcher algorithm on a 3D image scan with white to red changing light.

B. PyLops Deblur

In Fig. 12 , the results of applying various PyLops deblurring algorithms to an image of a cube are presented. The figure is organized into six main images and two line plots:

- **Original Image**: The top-left image shows the original, unblurred cube.
- **Blurred Image**: The top-center image displays the cube after applying a blurring operator.
- **Deblurred Image**: The top-right image represents the cube after applying a basic deblurring algorithm.
- **FISTA Deblurred**: The bottom-left image shows the result of using the Fast Iterative Shrinkage-Thresholding Algorithm (FISTA) for deblurring.
- **TV Deblurred**: The bottom-center image presents the cube after using Total Variation (TV) regularization for deblurring.
- **TV+Haar Deblurred**: The bottom-right image shows the result of combining TV regularization with Haar wavelet transform for deblurring.

Additionally, two line plots compare horizontal and vertical sections of the images:

- **Horizontal Section**: The top-right plot compares intensity values along a horizontal line across the images, showing how each deblurring method restores the original image compared to the blurred one.
- **Vertical Section**: The bottom-right plot compares intensity values along a vertical line across the images, illustrating the effectiveness of each deblurring method.

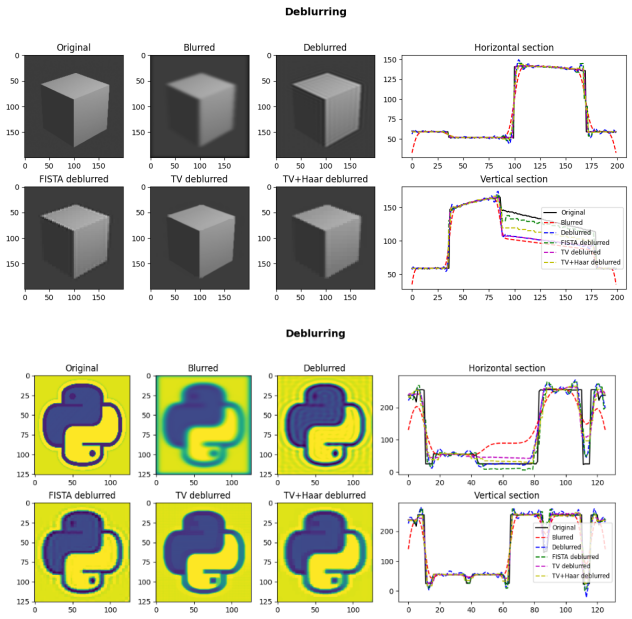

Figure 12. Deblurring methods using Python PyLops [13]

C. Zoomed In Examples

In this section the pictures used in the report are visible in a bigger format to have a better view on the details.

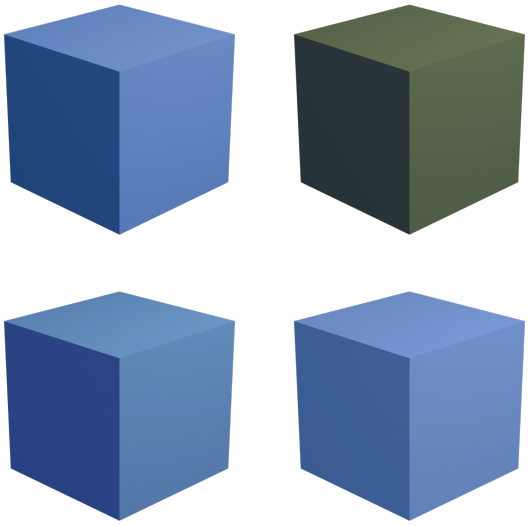

Figure 13. Examples of generated images under varying light conditions: (a) Original image, (b) Input image with altered lighting, (c) Image recolored using NumPy method, (d) Image recolored using Color Matcher method.

Figure 14. Examples of generated images: (a) original image, (b) input image, (c) Numpy, (d) recolored with colormatcher.