TU WIEN Informatics

Detaillierungsgradkonstruktion für die Synthese neuartiger Ansichten

Anti-Aliasing für Gaussian Splatting durch Skalenraumoptimierung

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Visual Computing

eingereicht von

Ole Siemers, BSc
Matrikelnummer 01529727

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Privatdoz. Mag. Dr.techn. Peter Kán
Mitwirkung: Dr.techn. Diana Marin, BSc MEng.

Wien, 19. März 2026

Ole Siemers	Peter Kán

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

TU WIEN Informatics

Image Based Level-of-Detail Construction for Novel View Synthesis

Anti-Aliased Gaussian Splatting via Scale-Space optimisation

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Visual Computing

by

Ole Siemers, BSc
Registration Number 01529727

to the Faculty of Informatics

at the TU Wien

Advisor: Privatdoz. Mag. Dr.techn. Peter Kán
Assistance: Dr.techn. Diana Marin, BSc MEng.

Vienna, March 19, 2026

Ole Siemers | Peter Kán

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Erklärung zur Verfassung der Arbeit

Ole Siemers, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.
Ich erkläre weiters, dass ich mich generativer KI-Tools lediglich als Hilfsmittel bedient habe und in der vorliegenden Arbeit mein gestalterischer Einfluss überwiegt. Im Anhang „Übersicht verwendeter Hilfsmittel“ habe ich alle generativen KI-Tools gelistet, die verwendet wurden, und angegeben, wo und wie sie verwendet wurden. Für Textpassagen, die ohne substantielle Änderungen übernommen wurden, habe ich jeweils die von mir formulierten Eingaben (Prompts) und die verwendete IT-Anwendung mit ihrem Produktnamen und Versionsnummer/Datum angegeben.

Wien, 19. März 2026

Ole Siemers

Danksagung

Ich danke Peter dafür, dass er das Projekt organisiert und den zentralen Impuls gegeben hat, diese Richtung einzuschlagen. Ohne seine Initiative wäre diese Arbeit nicht in dieser Form möglich gewesen.

Mein besonderer Dank gilt meiner Betreuerin Diana für ihren Rat und ihre Bereitschaft, sich Zeit für unsere Diskussionen zu nehmen. Unsere Diskussionen waren immer inspirierend und motivierend. Mit ihrer Expertise und ihrer motivierenden Art hat sie maßgeblich zu dieser Arbeit beigetragen.

Danke an Annalena für ihre Hilfsbereitschaft und dafür, dass sie sich mehr als einmal für mich eingesetzt hat.

Ich danke Lukas und Johannes dafür, das Realtime-Rendering-Meeting organisiert zu haben.

Ich danke Elias für den Krapfen.

Meiner Familie und meinen Freunden danke ich von Herzen für ihre bedingungslose Unterstützung, ihr Vertrauen in mich, ihre ständige Motivation und ihr Korrekturlesen. Ohne euren Rückhalt wäre diese Arbeit nicht möglich gewesen.

Ich danke den Leuten der Computergrafik-Gruppe für den Austausch und die Gespräche. Ich danke den Leuten der Computational Geometry and Design Gruppe dafür, dass ich ihre Hardware verwenden durfte.

Ich hoffe, dass alle, die mich auf diesem Weg begleitet haben – ob namentlich genannt oder nicht – wissen, wie sehr ich ihre Unterstützung schätze.

Acknowledgements

I thank Peter for organising the project and providing the key impetus to take this direction. Without his initiative, this work would not have been possible in this form.

I am grateful to my supervisor, Diana, for her advice and her willingness to devote time to our discussions. Our discussions were always inspiring and motivating. With her expertise and her motivating manner, she has made a significant contribution to this work.

Thanks to Annalena for her helpfulness and for standing up for me on more than one occasion.

I thank Lukas and Johannes for organising the real-time rendering meeting.

I thank Elias for the Krapfen.

I thank my family and friends from the heart for their unconditional support, belief in me, constant encouragement, and proofreading. Without your support, this work would not have been possible.

I thank the members of the Computer Graphics Group for the discussions and exchanges. I thank the members of the Computational Geometry and Design Group for allowing me to use their hardware.

I hope that everyone who has accompanied me on this journey – whether mentioned by name or not – knows how much I appreciate their support.

Kurzfassung

Diese Arbeit stellt ein Grob-zu-Fein-Optimierungsverfahren für 3D Gaussian Splatting (3DGS) vor, das während der Optimierung eine Level of Detail (LoD)-Hierarchie aufbaut, die selektiv dargestellt werden kann. Durch die schrittweise Anpassung der Auflösung reduziert das Verfahren den Rechenaufwand, beschleunigt die Optimierung und erzeugt dabei eine LoD-Hierarchie.

Basierend auf der Abtastrate, definiert als das Verhältnis zwischen der Auflösung, bei der das Modell optimiert wurde, und der Auflösung, bei der es betrachtet wird, wird ein selektives Darstellungsverfahren vorgestellt. Die selektive Darstellung reduziert die Anzahl der verarbeiteten Primitiven und mindert Aliasing-Fehler, auf Kosten einer erhöhten Speichernutzung auf der Graphics Processing Unit (GPU) durch die Verwendung mehrerer unabhängiger LoD-Ebenen. Das Verfahren wird anhand von 3DGS- und Elliptical Weighted Average (EWA)-Filtering als Vergleichsbasis auf geläufigen 360° und Luftbild-Datensätzen bewertet, wobei der Schwerpunkt auf Darstellungen mit niedriger Auflösung sowie auf entfernten Betrachtungspunkten liegt.

Die Ergebnisse zeigen, dass die Methode die Optimierung beschleunigt und die Anzahl der dargestellten Primitive reduziert. Insbesondere bei entfernten oder niedrigauflösenden Ansichten werden Bilder im Vergleich schneller generiert und Aliasing-Fehler nehmen ab. Bei voller Auflösung bleibt die visuelle Qualität annähernd gleich der Vergleichsbasis. Obwohl das Verfahren während der Darstellung zusätzlichen GPU-Speicher benötigt, bietet es einen praktischen Ansatz zur schnellen Optimierung kompakterer Modelle, die mit geringem Aliasing dargestellt werden.

Abstract

This thesis presents a coarse-to-fine optimisation method for 3D Gaussian Splatting (3DGS) that constructs a Level of Detail (LoD) hierarchy during optimisation, which can be rendered selectively. By gradually adjusting the resolution, the method reduces computational effort, speeds up optimisation and generates a LoD hierarchy in the process.

Based on the sampling rate, defined as the ratio of the resolution at which the model was optimised to that at which it is viewed, a selective rendering method is presented. Selective rendering reduces the number of primitives processed and mitigates aliasing errors, at the cost of increased memory usage on the Graphics Processing Unit (GPU) due to multiple independent LoD levels. The method is evaluated using 3DGS and Elliptical Weighted Average (EWA)-filtering as a basis for comparison on common 360° and aerial image datasets, with a focus on low-resolution renderings and distant viewpoints.

The results show that the method speeds up optimisation and reduces the number of processed primitives. Particularly for distant or low-resolution views, images are generated more quickly, and aliasing errors are reduced. At full resolution, the visual quality remains approximately the same as the baseline. Although the method requires additional GPU memory during rendering, it offers a practical approach to faster optimisation of more compact models that are rendered with reduced aliasing.

Contents

CHAPTER 1

Introduction

Selective Rendering Full Resolution Model

Figure 1.1: Demonstration of the method presented. A far view rendering of the *ArtSci* scene. Right: Image rendered with 3DGS uses 7.5 million primitives and exhibits aliasing. Thin structures appear too thick, and the overall appearance is too bright. Left: The selective rendering as presented in this thesis uses only 5.5 million primitives. Aliasing effects are reduced.

Reconstructing the geometric shape or visual appearance of the physical world, whether from active measurements (e.g., laser scans or sonar) or passive image-based observations, is known as 3D reconstruction. Aerial imagery, captured via drones or aircraft, is a primary data source for such reconstructions. Numerous applications exist for aerial 3D reconstruction, such as Building, Bridge, or Railway inspection [1]. In disaster management, structural damage to buildings can be remotely evaluated after disasters like earthquakes or floods [2, 3]. In environmental biology, 3D reconstructions from drone imagery are used for wildlife assessment or monitoring deforestation [4].

Methods for scene reconstruction and Novel View Synthesis (NVS) have made significant progress in the last few years towards faster optimisation and real-time view synthesis [5,

6, 7]. 3D Gaussian Splatting (3DGS) [7], a method for NVS, uses a set of images with their internal and external camera parameters as input and produces a set of Gaussian distributions that reconstruct the scene. For NVS, these Gaussians are projected to 2D splats that are efficiently rendered on the rasterisation hardware of a Graphics Processing Unit (GPU). The rasterisation process is fully differentiable, enabling gradient-based optimisation of the scene appearance from posed images.

1.1 Problem Statement

For increasingly large-scale reconstructions, several challenges become apparent. Novel views can deviate significantly from the views used to reconstruct the scene. When the frequency of the reconstructed signal does not match the frequency at which it was recorded, aliasing effects can negatively affect the reconstruction [8, 9]. Anti-aliasing techniques proposed in Mip-Splatting [9] effectively reduce aliasing but do not reduce the number of primitives in far views, making them infeasible for larger reconstructions.

Additionally, the computational requirements for reconstructing and rendering large scenes can be very high, prohibiting interactive reconstruction even on powerful hardware with ample video memory. To reduce computational demands and aliasing, reconstructions should dynamically adjust the Level of Detail (LoD) based on the required reconstruction detail [10, 11, 8].

Existing work treats creating an LoD hierarchy as an additional step that merges primitives from high-resolution reconstructions, adding complexity to the process [10, 11, 8]. Other contributions have shown that starting with lower input resolutions and gradually increasing them in a coarse-to-fine manner improves reconstruction quality while reducing the number of primitives needed to represent the scene [12]. These findings motivate the research on novel optimisation schedules and LoD mechanisms to optimise and display reconstructions of large-scale scenes.

1.2 Aim of the Work

This work aims to explore a method for on-the-fly LoD creation based on coarse-to-fine optimisation scheduling for 3DGS [7]. Special focus is placed on reducing aliasing errors in the reconstruction by optimising a Level of Detail (LoD) hierarchy on a scale-space representation of the image, from which the scene can be rendered selectively. The aim is to examine the effects of coarse-to-fine optimisation scheduling on the possibility of using this information to construct a LoD scheme. This thesis evaluates how a LoD hierarchy can be constructed during optimisation, and how this affects optimisation speed and reconstruction quality. Further, it is examined how selectively rendering the hierarchy compares to state-of-the-art anti-aliasing methods for 3DGS, in terms of rendering speed and quality.

1.3 Methodological Approach

This thesis proposes a coarse-to-fine optimisation process that reduces the number of primitives while maintaining visual quality. During optimisation, intermediate models optimised on low resolution are saved and eventually merged into an LoD hierarchy. Optimising for different resolutions offers a natural way to determine the ideal reconstruction frequency. The frequency at which a viewpoint is reconstructed depends on the camera's focal length and resolution, as well as the distance to the scene. The sampling frequency is used to determine an appropriate reconstruction frequency for LoD selection and anti-aliasing in large-scale reconstructions where the reconstructed viewpoints differ significantly from the views used to optimise the scene. Understanding efficient optimisation schedules and how to integrate them into the construction of an LoD scheme for a scene is a step towards photorealistic reconstruction of larger scenes with less computing power. A visual comparison of the proposed method with the 3DGS baseline in an aerial view is shown in Figure 1.1, where fewer primitives are required to render the scene and the aliasing visible in the baseline is effectively reduced with selective rendering.

This thesis aims to answer the following research questions:

Q1) How does coarse-to-fine scheduling impact the trade-off between convergence speed, reconstruction quality, and model size compared to baseline 3DGS?

Q2) How can coarse-to-fine optimisation enable on-the-fly LoD hierarchy construction, and what are the implications for memory requirements?

Q3) How can sampling-distance-based selective rendering adapt reconstruction quality and model size to different rendering resolutions and viewpoint distances, and what are the trade-offs in terms of aliasing, rendering speed, and primitive count?

Experiments are conducted on widely used Datasets for NVS [13, 14, 15, 16, 17]. The results are compared with respect to model size, rendering speed, and photometric and perceptual errors. The reconstruction error is measured on images withheld from the optimisation at different resolutions. Visualisations of the errors, the sampling distances used, and the LoD are provided.

1.4 Contributions

The main contributions of this thesis are:

- a coarse-to-fine optimisation schedule for 3DGS that saves intermediate models and turns them into an LoD hierarchy within a single training run;
- a sampling-distance-based selective rendering algorithm that chooses the appropriate LoD at render time using per-Gaussian metadata;

- an evaluation on standard NVS benchmarks and larger aerial scenes, showing reduced optimisation cost, improved low-resolution behaviour, and fewer rendered primitives for distant views.

1.5 Outline

The thesis is structured as follows. Chapter 2 provides a background to the topics, discusses related literature, and gives an overview of the state-of-the-art methods using 3DGS related to the problem tackled in this thesis. Chapter 3 presents details of the proposed methods for coarse-to-fine scheduling and selective rendering. Chapter 4 presents the implementation of the existing work on which this thesis builds and the extensions made to implement the proposed method and evaluate it. Chapter 5 presents an evaluation of the proposed method against the 3DGS baseline and an additional anti-aliasing method [9], addressing the research questions. In Chapter 6, the research questions are reviewed, with respect to the results and limitations. Finally, possible areas for future work are discussed.

CHAPTER 2

Related Work

3D reconstruction and NVS are central topics in computer vision and computer graphics. Extensive research has been conducted in these fields over several decades. This chapter presents the theoretical background for 3D reconstruction, defines key concepts, and surveys the current state of the art.

2.1 Theoretical Background & Definitions

The image formation process maps the 3D world onto a 2D plane, inherently losing depth information. Recovering this lost 3D structure requires either prior knowledge of the scene or multiple views of the same environment. In this section, we discuss the principles of 3D reconstruction from multiple views, relevant theorems for sampling and reconstructing discrete signals, multi-scale representations, and the synthesis of novel views from a set of images.

2.1.1 The Ideal Pinhole-Camera Model

The ideal pinhole camera model assumes no lens distortions and projects lines onto lines [18]. It is similar to the camera obscura as illustrated in Figure 2.1. Assuming rectangular photosensitive elements, parameters of the ideal pinhole camera are the focal length f and the principal point $\vec{c} \in \mathbb{R}^2$. The focal length is the distance from the focal point to the image plane. The principal point $\vec{c}$ is the position on the image plane that is connected orthogonally to the centre of projection. Written in homogeneous coordinates, the pinhole camera projects a point in homogeneous world coordinates $\vec{X} = (X, Y, Z, 1)^T$ to a point on the image plane $\vec{x} = (x, y, 1)^t$.

$$\lambda \vec{x} = P\vec{X} \tag{2.1}$$

Where λ is a scalar depth $\frac{f}{Z}$, and the projection matrix P is assembled from the intrinsic and extrinsic camera parameters, K and $[R|\vec{t}]$. R describes the orientation, and $\vec{t}$ the position of the camera in world-space.

$$K = \begin{bmatrix} f & 0 & \vec{c}_x \\ 0 & f & \vec{c}_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

$$P = K[R|\vec{t}] \tag{2.3}$$

Note that in some implementations, a vertical and a horizontal focal length are used, each equal to the focal length f multiplied by a scaling factor to account for non-uniform pixel spacing or non-rectangular photosensitive elements. This scaling is left out for clarity. The parameters of the ideal pinhole camera model are illustrated in Figure 2.2.

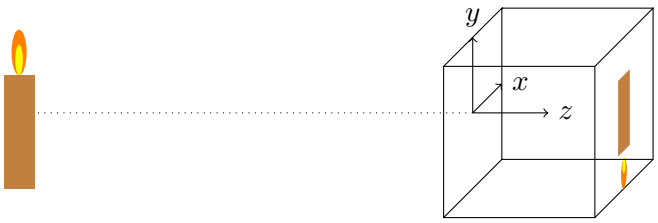


Figure 2.1: The camera obscura projects to an inverted image of the scene

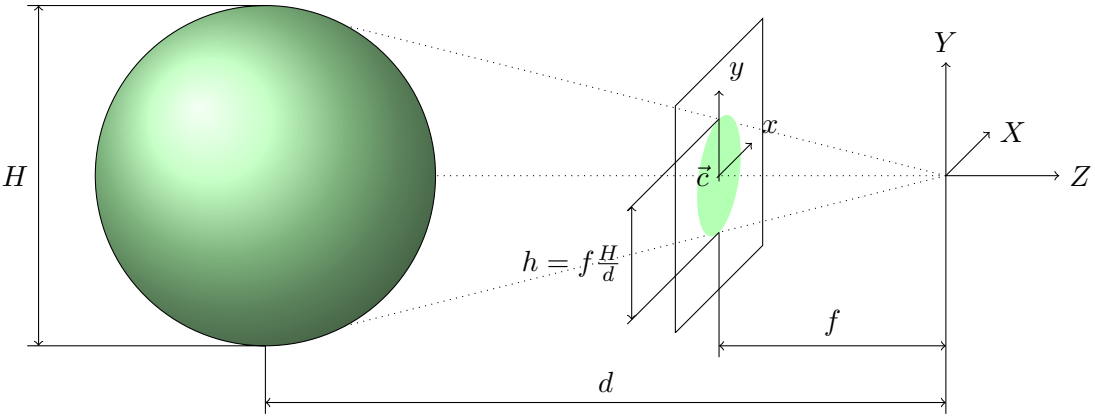
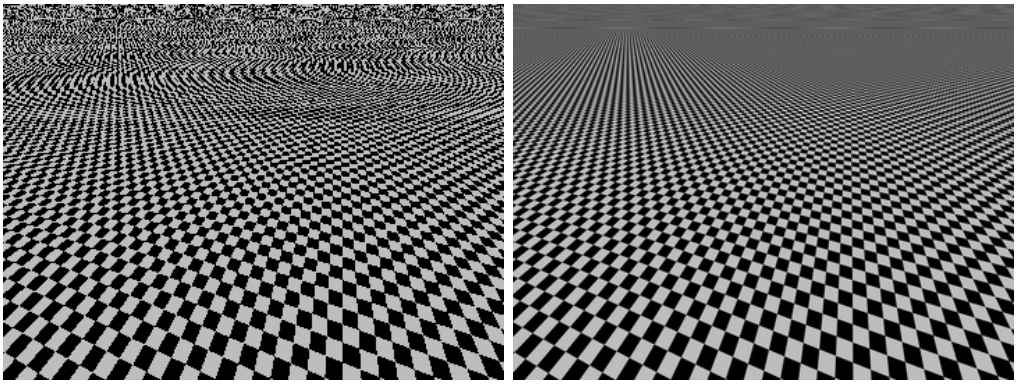


Figure 2.2: Illustration of an ideal pinhole camera model. The camera is looking along the negative z-axis. Projecting an object of height H to the image plane results in a projection on the image plane with height h depending on the focal length f and the distance d from the camera centre to the object.

2.1.2 Sampling

A continuous signal can be faithfully reconstructed from a discrete signal if the Nyquist-Shannon Sampling Theorem is respected [19, 20, 21]. The Nyquist-Shannon Sampling



(a) Aliased (b) Super-Sampled

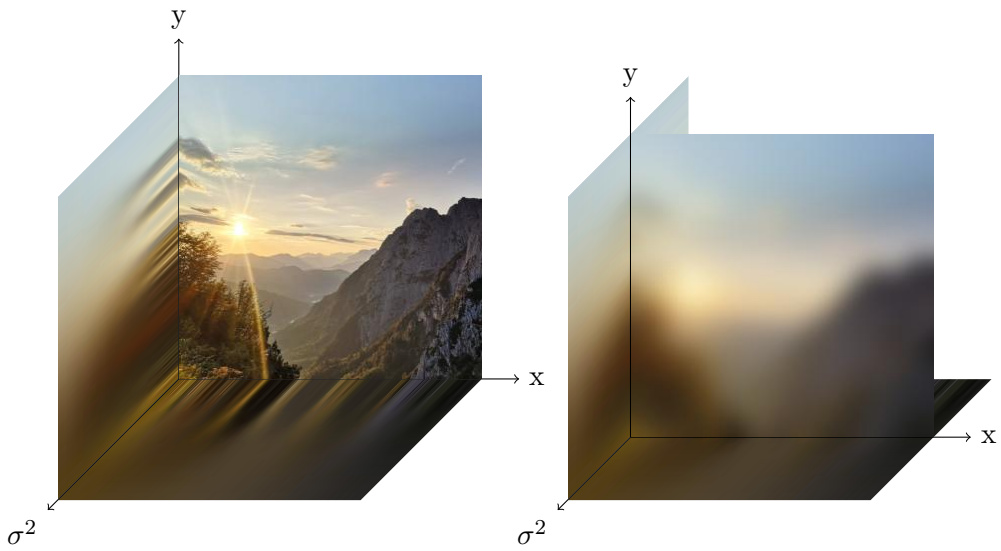
Figure 2.3: Two checkerboard patterns under perspective transformation. The image on the left uses one sample per pixel and exhibits strong aliasing artefacts as soon as the texture frequency falls below half the sampling frequency. The image on the right was generated at 16 times the resolution of the image on the left, then filtered and downsampled to the original resolution. Aliasing is still visible in the right image, though it is largely suppressed, as black-and-white pixels average out at a distance.

Theorem states that a bandlimited signal must be sampled at twice its bandwidth to avoid aliasing. Therefore, the information that can be sampled, processed, and reconstructed is, in any case, finite. When a signal is downsampled, higher frequencies must be removed, i.e., the signal must be low-pass filtered before resampling. Failing to do so can result in various aliasing errors, as illustrated in Figure 2.3. Arguing about sampling frequencies is difficult for 3D point-cloud-based renderings because of irregular sampling distances and perspective distortion. Therefore, an approximation of the sampling distance, based on the image's sampling distance from the viewpoints used to optimise the point cloud, is used in this thesis.

2.1.3 Scale-Space

Scale-space theory [22, 23] describes the Gaussian distribution and its derivatives as the only functions that can be used to rescale a signal and analyse it at different scales without prior knowledge of its content. Analysing a signal at several scales is necessary if a pattern of interest may occur at an unknown scale. Particularly in images, which resemble perspective projections of the world onto a plane, features may appear at any scale.

The scale-space augments the spatial dimension with a continuous scale dimension as illustrated with the scale-space cube in Figure 2.4. The top of the cube shows an image with full details. A cut through the cube shows a blurred, low-pass-filtered version of the image. When reducing the scale of a signal, high frequencies vanish, while low frequencies remain. An important property of the scale-space is that whenever frequencies



(a) Cut at highest frequency. (b) Cut at medium frequency.

Figure 2.4: (a) The outer layers of a scale-space cube show the highest frequencies. (b) A cut through a scale-space cube shows a low-frequency version of the original signal.

are reduced by Gaussian smoothing, no extrema can appear; the number of extrema is usually lower than for high-frequency signals. In this work, this property is instrumental in priming the optimisation process to focus on low frequencies first, and to optimise higher frequencies later.

2.1.4 Image Pyramids

When a low-frequency version of a signal is present, it follows from the sampling theorem that it can be represented with fewer samples if a small error is acceptable. The Gaussian distribution in the spatial domain is also Gaussian in the frequency domain, with the standard deviation in the frequency domain being the inverse of that in the spatial domain [1]. A spatial convolution is equivalent to a multiplication in the frequency domain. Convolving a signal spatially with a discrete Gaussian, therefore, attenuates the high-frequency components of the signal. However, since the continuous distribution has an infinite support, choosing the kernel size and standard deviation is a trade-off between introducing aliasing errors and overblurring the signal. In practice, a kernel size of 3 times the standard deviation provides a reasonable trade-off, with less than 1% of the distribution lying outside the Nyquist band [24]. A thorough examination of the errors introduced by approximating discrete Gaussian smoothing kernels is provided by Lindeberg [25].

[1] See also `https://mathworld.wolfram.com/FourierTransformGaussian.html`, last accessed at 13.3.2026

When an image is repeatedly filtered and downsampled, a pyramidal structure is obtained, enabling image analysis across different scales and frequency bands [26]. The construction of an image pyramid is illustrated in Figure 2.5. Such pyramidal structures, in which low-frequency components are represented with fewer samples than high-frequency components, have a wide variety of applications, such as when low-pass-filtered versions of a signal must be precomputed [27] or when analysis is performed across different bandwidths [28]. For rendering textured surface models, Mip-Mapping [27] is used to select the appropriately filtered version of a texture for reconstructing a perspectively distorted surface. Scale Invariant Feature Transform (SIFT) [28] involves finding interest points in a Difference of Gaussians (DoG) pyramid, which approximates a Laplacian Pyramid [26] - a representation that splits the original signal into different frequency bands.

2.1.5 Structure from Motion

One way to recover depth information is to use the scene's structure and the camera's motion, a process known as Structure from Motion (SfM) [29]. From a set of corresponding image points and known intrinsic camera parameters, the depth of the points can be reconstructed. Meaningful features and descriptors from an image can be extracted with the SIFT [28]. The SIFT algorithm splits an image into several frequency bands with DoG. The locations of extrema in each bandwidth are selected as features after removing low-contrast points. Neighbourhood gradients are used to calculate the descriptor for each feature. The feature point descriptors are matched, and outliers are removed using RANdom SAmple Consensus (RANSAC) [30]. For an initial pair of images, the fundamental Matrix describing the relative pose $[R|t]$ is estimated. Once the relative pose is known, the 3D positions of the matched 2D points are triangulated. The procedure is repeated for new images, and the reconstruction error is distributed over all estimations with Bundle Adjustment (BA) [31]. The result of the process is a sparse point cloud, where the points are relatively scaled to each other - the absolute scale is unknown without having a reference scale in the scene.

2.1.6 Level of Detail

Level of Detail (LoD) is used to hierarchically manage the geometric complexity of 3D models, balancing computational resources with visual fidelity. The idea is to represent a model at multiple resolutions. Whereas a lower LoD provides an overview and is computationally efficient, a higher LoD offers full detail but requires more resources for storage, transmission, and processing. The selection of an LoD aims to balance the computational cost versus the desired quality and detail. Heuristics used to select the proper level include, e.g., the distance of an object to the camera [32] or its projected size on the image plane [33, 34].

Automatic construction of LoD hierarchies can be approached either by subdividing a coarse model to a finer resolution [35] or by simplifying a high-resolution model [36, 37].

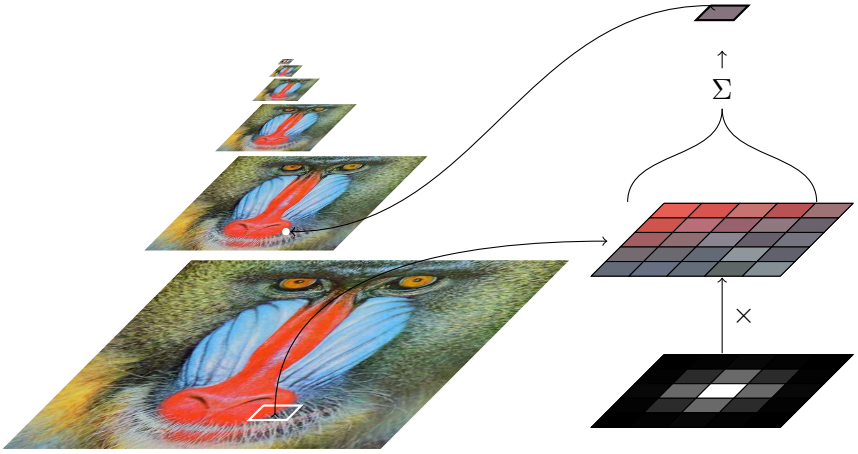


Figure 2.5: Construction of an image pyramid by convolving it with a Gaussian kernel and subsampling it. Every pixel in the next layer is the sum of pixels within a window weighted by a discrete approximation of a Gaussian kernel. The element-wise multiplication is denoted as $\times$, the sum is denoted as Σ. The *Mandrill* image is taken from the *USC-SIPI Image Database*[2].

For very large models, the LoD has to be preprocessed out-of-core to avoid overstraining GPU memory, i.e., loading it into host memory and moving it to GPU memory as needed [38]. A classical work concerning the selection of a LoD proposes the use of a hierarchical scene graph for efficient querying of the proper LoD for distinct objects within a scene [39]. The scene is represented as a tree, with the camera at the root and the nodes as high-resolution models. This hierarchical data structure allows efficient visibility computation requiring only a logarithmic number of comparisons relative to the number of objects in the scene [39].

While many methods focus on creating LoD hierarchies of surface models [40], there are also hybrid representations for surfaces that fall back to point clouds [41], volumetric representations [42], or prerendered images [43] when viewed from a distance. Crassin et al. [44] use a voxel representation of a scene at several resolutions to precalculate the radiance within each cell of an octree that can be efficiently queried for indirect lighting calculation.

When rendering large point clouds, different hierarchical data structures lend themselves to efficient in-memory representation. Rusinkiewicz and Levoy [33] propose using a hierarchy of bounding spheres whose projected radius is used as a criterion to decide how deep the tree has to be recursed. Dachsbacher et al. [34] propose a sequential point tree that uses an octree, which stores the radius of the bounding sphere of their children. Nodes that satisfy the LoD criterion are added to a sequential list for processing on the GPU [34]. The authors note that, as a limitation, the proposed data structure does not support visibility culling within an object, so the full model must be present in GPU

[2] `https://sipi.usc.edu/database/database.php`, last accessed at 13.3.26

memory. Gobbetti and Marton [45] propose layered point clouds, in which the set of points is separated into disjoint subsets with different spacings, which can be additively rendered. Another method presented by Schütz et al. [32] follows an additive scheme, where higher levels are added on top of the lower ones, and dithering reduces popping artefacts at level seams [32]. An illustration of the method proposed by Schütz et al. [32] is shown in Figure 2.6.

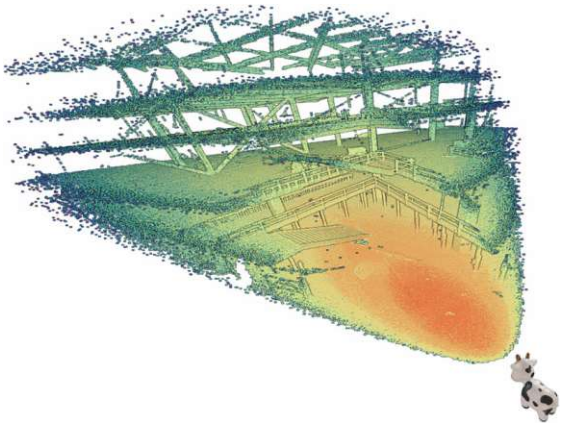


Figure 2.6: A continuous LoD for point clouds shown from a third-person view. The red dots represent higher detail accuracy, while the blue dots represent lower detail accuracy. In this method, the LoD decreases with distance from the viewpoint and distance from the view centre. The image is courtesy of the authors [32].

2.1.7 Novel View Synthesis

When geometric accuracy is less important than the appearance of the reconstruction, novel views can be generated from reference images. To produce images from previously unseen viewpoints or new camera intrinsics, the scene can be modelled as a function of space and direction that maps to light intensity [46]. The intensity of light flowing from a point in space in a given direction can be physically modelled by its radiance. Radiance is a radiometric quantity that measures the amount of radiation that is emitted, reflected, or transmitted per unit area per unit solid angle [47, 48]. It depends on the position and the direction and is measured in watts per square meter per steradian ($\frac{W}{m^{-2}sr}$). A radiance field maps the position $\vec{x} \in \mathbb{R}^3$ and a view direction $\theta, \phi \in \mathbb{R}$ to a volume density $\rho \in \mathbb{R}$ and color $\vec{c} \in \mathbb{R}^3$ [5].

An early approach to NVS is the Lumigraph - a system that models the radiance as a 4D function on the entry and exit points of a ray through a cube [49]. When a dense reconstruction is already given, weights for blending the input image onto the geometry can be estimated by minimising the photometric error [15]. NVS enables direct downstream applications such as video frame interpolation [50] and rendering a view with novel camera intrinsics [51], and even allows the extraction of dense geometric surface models from the volumetric representation [52, 53, 54, 55, 56].

2.1.8 Image-Based Error Metrics

When comparing an image y to a reconstructed version of it $\hat{y}$, the error between them can be measured in different ways. In the following paragraphs, N denotes the number of pixels in an image.

Mean Absolute Error The Mean Absolute Error (MAE or $\mathcal{L}_1$) measures the linear pixel-wise distances.

$$\mathcal{L}_1(y, \hat{y}) = \frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{y}_i| \tag{2.4}$$

Mean Squared Error The Mean Squared Error (MSE) is used to estimate the amount of Gaussian noise, but cannot measure perceived quality, since different corrupted images with the same MSE can have very different perceived quality [57].

$$\text{MSE}(y, \hat{y}) = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \tag{2.5}$$

Peak Signal to Noise Ratio Peak Signal to Noise Ratio (PSNR) is a logarithmic transform of the MSE. It is a measure for energy expressed in decibels [57]. As for $\mathcal{L}_1$ and MSE, PSNR does not correspond to perceptual quality.

$$\text{PSNR}(y, \hat{y}) = 10 \log_{10} \frac{\text{Peak}^2}{\text{MSE}(y, \hat{y})} \tag{2.6}$$

"Peak" denotes the dynamic range, e.g. $\text{Peak} = 2^8 - 1 = 255$ for 8-bit integers.

Structural Similarity Index Measure The Structural Similarity Index Measure (SSIM) is a perceptual metric [58] that accounts for local luminance, contrast, and structure between image patches. In this paragraph, the mean and standard deviation of the pixel values within an image patch are denoted as μ and σ. The constants c_1 and c_2 are small constants to avoid numerical instabilities.

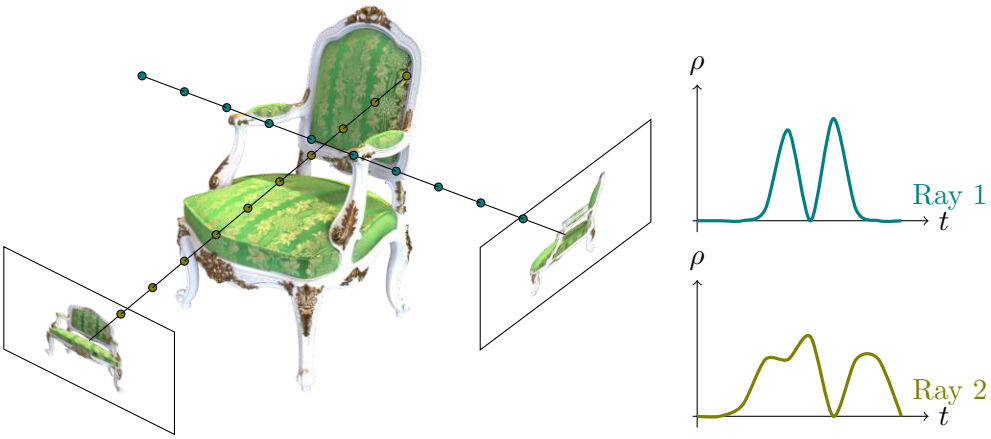
$$l(y, \hat{y}) = \frac{2\mu_y \mu_{\hat{y}} + C_1}{\mu_y^2 + \mu_{\hat{y}}^2 + C_1} \tag{2.7}$$

$$c(y, \hat{y}) = \frac{2\sigma_y \sigma_{\hat{y}} + C_2}{\sigma_y^2 + \sigma_{\hat{y}}^2 + C_2} \tag{2.8}$$

$$s(y, \hat{y}) = \frac{\sigma_{y\hat{y}} + C_3}{\sigma_y \sigma_{\hat{y}} + C_3} \tag{2.9}$$

$$\text{SSIM}(y, \hat{y}) = l(y, \hat{y}) \cdot c(y, \hat{y}) \cdot s(y, \hat{y}) \tag{2.10}$$

The function $l(y, \hat{y})$ quantifies how closely the average brightness of y matches that of $\hat{y}$. For contrast comparison, $c(y, \hat{y})$ evaluates the similarity in variance between the two images. Finally, $s(y, \hat{y})$ assesses the degree of correlation between y and $\hat{y}$.



(a) Sampling the NeRF (b) Density along the rays

Figure 2.7: Illustration of a NeRF's functionality. The radiances are sampled from the NeRF at positions along the viewing rays and integrated to the final pixel colour. Figure (a) exemplarily shows two rays cast from the camera centres of two ground truth images through the volume. Figure (b) shows the density along the two rays. Illustration adapted from [5].

Learned Perceptual Image Patch Similarity A similarity measure can also be optimised from a large set of image data. The Learned Perceptual Image Patch Similarity (LPIPS) is computed as the $\mathcal{L}_2$ distance between encoded images, e.g., using a VGG encoder [59] to obtain a similarity measure that correlates with how humans rate image similarities [60].

2.1.9 Neural Radiance Fields

The radiance field has been modelled as a Neural Radiance Field (NeRF) [5] by optimising the parameters of a Multilayer Perceptron (MLP) [61] to minimise the reconstruction error of an image sampled from the NeRF. Camera parameters of the supervision images are required and can be extracted with SfM. As illustrated in Figure 2.7, an image is composited by casting rays through the volume for each pixel, evaluating the MLP at several positions along each ray, and integrating them to the final pixel colour. The MLP parameters are optimised by propagating image errors back through the internal weights and minimising the error using gradient descent.

2.1.10 Point-Based Radiance Fields

Xu et al. [62] extend the NeRF idea to point-based rendering by starting from a sparse reconstruction obtained with SfM, estimating the features and appearance of each point with a PointNet [63], and rendering by integrating radiance along viewing rays. Points are added by estimating empty regions in space and the distance to the surface of the reconstructed object. The authors observe a significantly reduced optimisation time compared to volumetric NeRF methods.

A seminal work on point-based representations for surface rendering is the Elliptical Weighted Average (EWA) Surface Splatting Algorithm [64]. EWA Surface Splatting uses 3D anisotropic elliptical reconstruction kernels parametrised by mean and variance that absorb or emit light in a volume. This 'reconstruction kernel' is further referred to as a 'Gaussian primitive' or 'Gaussian'. To evaluate the volume projection, the Gaussian primitives can be either integrated along a viewing ray or projected to a 2D 'splat' and alpha-blended using a splatting algorithm that utilises rasterisation hardware.

Extending that idea of EWA Splatting, the parameters of the Gaussian primitives that represent a radiance field can be estimated in an image-based process by minimising the reconstruction error in a differentiable rendering pipeline, back-propagating the reconstruction error from a pixel to the Gaussians [65, 66, 7]. Kopanas et al. [65] optimise a set of anisotropic 2D Gaussians with a neural renderer. Zhang et al. [66] optimise a set of isotropic 3D Gaussians with a view-dependent colour representation in spherical harmonics, which can be efficiently rendered by rasterisation rather than raymarching, achieving interactive reconstruction framerates and fast optimisation.

2.1.11 3D Gaussian Splatting

Extending these ideas, the scene with 3D Gaussian Splatting (3DGS) [7] is represented as a set of anisotropic 3D Gaussian primitives parametrised by their position μ, covariance Σ, and opacity α. An example of a NVS of a synthetic dataset is shown in Figure 2.8.

To ensure the positive semi-definiteness of the covariance matrix Σ, it is represented with a scaling and rotation matrix.

$$\Sigma = RQQ^T R^T \tag{2.11}$$

The matrices Q and R are constructed from a 3D scaling vector and a unit quaternion, respectively.

To render a scene from a given viewpoint, the means and covariances of the visible 3D Gaussians are projected to 2D. The colour of a pixel C is calculated by front-to-back alpha-blending all gaussians at the same projected screen position, sorted by their distance to the camera.

$$C = \sum_{i=1}^{n} c_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j) \tag{2.12}$$

Figure 2.8: 3DGS reconstruction of the *Drum* scene from the *NeRF Synthetic* dataset, supervision images, and camera poses overlaid. Screenshot taken from NerfBaselines[3].

The density of a 3D Gaussian at a position $\vec{X}$ is noted as $G(\vec{X})$.

$$G(\vec{X}) = e^{-\frac{1}{2}(\vec{X}-\vec{\mu})^T \Sigma^{-1}(\vec{X}-\vec{\mu})} \tag{2.13}$$

The Jacobian J of the covariance Σ is used to find an affine approximation of the 2D projection of the 3D Gaussian given a view projection W.

$$\Sigma' = JW\Sigma W^T J^T \tag{2.14}$$

The optimisation process of 3DGS [7] is initialised from a sparse point cloud obtained with SfM. To optimise the model's parameters, the gradient of the parameters θ is computed by backpropagating the reconstruction error $\mathcal{L}$ to the primitives' parameters. The error is obtained by computing a loss $\mathcal{L}$ of the rendered image to a ground truth image used as a supervision signal. As a loss, a linear combination of the L1 loss and the mean structural dissimilarity (DSSIM) is used.

$$\mathcal{L}(y, \hat{y}) = 0.2\mathcal{L}_1(y, \hat{y}) + 0.8\mathcal{L}_{\text{DSSIM}}(y, \hat{y}) \tag{2.15}$$

$$\mathcal{L}_{\text{DSSIM}}(y, \hat{y}) = \frac{1 - \text{SSIM}(y, \hat{y})}{2} \tag{2.16}$$

[3] `https://nerfbaselines.github.io/viewer/?p=gsplat-blender-drums.json`, last accessed 13.3.26

In order to update the parameters of the model, we are interested in the gradient of the measured error to the supervision signal $\partial\mathcal{L}$ with respect to the rendered pixel colour ∂C_i. The loss is calculated by rendering an image in a forward pass and comparing it to the supervision signal. The way in which the gradients of each parameter are derived from the loss is illustrated in Figure 2.9.

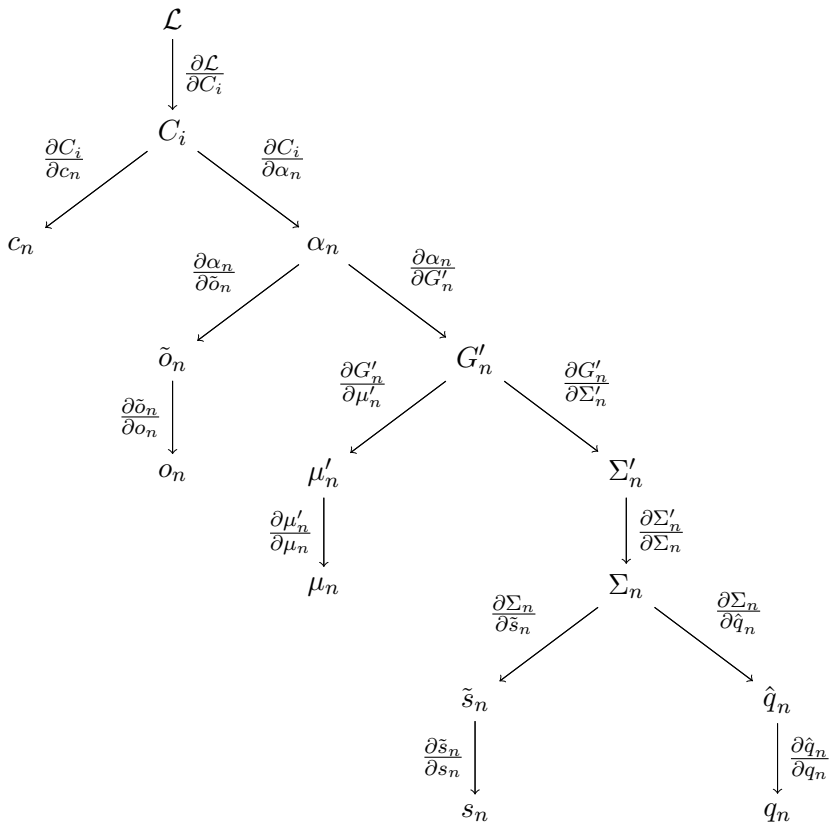


Figure 2.9: Derivation tree to compute the gradient of each parameter (leafs) of the Gaussian primitives from the loss (root). Illustration adapted from [67].

The model is densified by cloning or splitting primitives based on a heuristic that decides to split or clone them when the magnitude of their positional gradient exceeds a specified threshold. The opacity of every Gaussian is reset to zero at fixed intervals. Gaussians that do not recover to opacity and remain fully transparent are removed. This process is called Adaptive Density Control (ADC). Densification occurs during the first half of the optimisation over 15 000 epochs.

2.2 State of the Art in 3D Gaussian Splatting

While 3DGS achieves high-quality reconstructions in a reasonable time, much subsequent work has focused on addressing open questions. The following subsections describe recent

(a) Double Resolution (b) Original Resolution (c) Half Resolution

Figure 2.10: A 3DGS reconstruction at different resolutions showing aliasing. The same model of the *Garden* scene [13] was used to reconstruct the three images. The density of the Gaussians in the left image (a) rendered at twice the original resolution is too low, resulting in an overly dark appearance. The opposite effect is visible in the right image (c rendered at half resolution, where the Gaussians are too wide, and the image appears too bright.

work advancing the state of the art in 3DGS, focusing on improvements related to the methodology followed in this thesis. The focus of the following subsections is recent work concerning Anti-Aliased reconstruction, the model densification heuristics, the effects of coarse-to-fine scheduling, and large-scale reconstructions.

2.2.1 Anti Aliasing

When the camera parameters differ from the ones used to optimise a scene (i.e., the camera moves out of optimisation distribution or the focal length changes), the pixel size of the reconstructed image can be too large (zoom in) or too small (zoom out) compared to the projected size of the Gaussians of the optimised model. This results in a mismatch between the sampling frequencies, leading to aliasing artefacts in the reconstructed image, as illustrated in Figure 2.10.

To counteract aliasing, Yu et al. [9] propose Mip-Splatting, which ensures the reconstruction accounts for the sampling frequency through two measures. First, to avoid aliasing errors when zooming in, a world-space filter is applied during optimisation by scaling the covariance matrix based on the highest sampling rate available, the resolution of the supervision images, the focal length, and the distance to a Gaussian. This procedure is similar to the method for selecting the appropriate LoD used in this thesis, detailed in Chapter 3. Second, since the Gaussians are evaluated at the pixel centre, the projected Gaussians are low-pass filtered in screen space to ensure they cover at least a single pixel, thereby avoiding aliasing errors when zooming out. The formulation of this screen-space filter was originally proposed in EWA Surface Splatting [64] and is further referred to as EWA-Filtering in this thesis.

Another approach, proposed by Liang et al. [68], uses an analytic approximation of the sensor area to integrate Gaussian contributions over the pixel area, rather than using point samples. This approach is computationally more demanding and reports a 10%

lower framerate than Mip-Splatting, but it can reconstruct finer details [68].

The approach followed by Yan et al. [8] uses a multi-scale model in which Gaussians are aggregated within a voxel to form larger Gaussians that represent many small ones at a lower LoD. During rendering, Gaussians are selected from an LoD based on their expected pixel coverage. An advantage of their proposed method is that when the model is viewed from a distance, a lower LoD with fewer Gaussians can be used, thereby saving resources. Similar to the method presented in this thesis, the optimisation is supervised using both full and downsampled image versions.

2.2.2 Adaptive Density Control

The method for Adaptive Density Control (ADC) used in 3DGS [7] can create and remove Gaussians from the model in a non-monotonic way, leading to unpredictable memory peaks and producing more Gaussians than necessary to reconstruct a scene [69]. A lot of work has been devoted to the development of better heuristics for ADC [69, 70, 71]. The ADC mechanism should strike a balance between having enough primitives to represent the scene and avoiding overdensification to keep the model size manageable. Overdensification can significantly increase the time required for optimisation without improving the final model's quality [70]. Fang and Wang [70] examine the effects of different sampling techniques that reorganise the spatial distribution of Gaussians, and find that more regular samplings allow for reducing the number of Gaussians to represent the scene at the same quality level.

Bulo et al. [71] revise the ADC process and find that the positional gradient is particularly high on Gaussians that have just been cloned or split. They propose using a densification threshold based on image-based error, weighted by a Gaussian's contribution to a pixel's colour, and a smooth opacity decay rather than a hard reset. Additionally, they fix an opacity bias when cloning a Gaussian. They find that these improvements increase perceived quality, particularly in underdeveloped regions, even when the number of primitives remains the same.

2.2.3 Coarse-To-Fine Scheduling

When rendering and back-propagating, the model is evaluated per-pixel per-Gaussian. The runtime complexity of the 3DGS optimisation, therefore, depends mainly on the input resolution and the number of primitives to optimise [72]. One line of research has been exploring how to optimise the model on low frequencies first, then progressively increase image resolution to supervise the optimisation. This procedure leads to faster convergence speed, reduces the number of primitives, and lowers reconstruction error [73, 72]. Additionally, Coarse-to-Fine Scheduling has been shown to produce LoD hierarchies that can further be used for selective rendering [74, 73]

Jung et al. [75] observe that after initialisation, the 3DGS optimisation process fits low frequencies first. The authors find that a high number of initialisation points hinders the model from optimising low frequencies early in the process. Low-pass filtering the

Gaussians in 3D space, so they receive gradients from a larger neighbourhood early in optimisation, makes the optimisation more robust. The progressive Gaussian low-pass filter is implemented by modulating the anti-aliasing low-pass filter from a high value back to its original value. Their proposed improvements are applied during a warm-up phase of 10 000 epochs before densification begins. Together with their proposed ADC, which moves Gaussians farther apart, they can achieve lower reconstruction errors than the baseline, even when the model is initialised with random noise.

To reduce overreconstruction, Zhang et al. [76] propose regularising densification via a spectral discrepancy between the reconstructed and ground-truth images, prioritising the reconstruction of low frequencies early in the optimisation procedure.

Farooq et al. [74] examine the effect of blurring the input image with different low-pass filters and the scheduling of its kernel width during the densification phase. Their schedule can be applied to several backends, and they find that their novel schedule maintains high reconstruction quality compared to their baselines with a significantly lower number of primitives, which also reflects in less time required for the optimisation. A LoD hierarchy is produced by taking the model optimised on lower frequencies as a particular level in the hierarchy. Instead of blurring the supervision image, Seo et al. [73] constrain the Gaussians to have minimal size, which is increased at each LoD during optimisation. The reconstruction quality is improved over the proposed method of Farooq et al. [74], and the resulting LoD hierarchy is shown to be effective for rendering on low-end devices.

Chen et al. [72] reduce the number of Gaussians and increase the optimisation time by a densification schedule that is applied during the full optimisation procedure. They propose a ratio between the resolution of the supervision images and the number of Gaussians necessary to reconstruct the image. With their schedule, the optimisation resolution and the number of primitives are kept low early in the optimisation and progressively increased, with the full resolution used only in the last few epochs.

2.2.4 Large Scale Reconstruction

When reconstructing very large scenes, the memory requirements of the 3DGS optimisation quickly become a prohibitive factor for the method. Several ways have been proposed in the literature to keep models at manageable sizes. For optimisation, the model can be split into smaller spatial chunks that fit into the memory of a single machine, or optimised in parallel across several machines [10, 11, 77, 78]. When a model is rendered, a LoD mechanism is applied to have only the visible parts at a necessary LoD of a model in GPU memory [10, 11, 77].

Liu et al. [11] optimise the partitions of a scene separately at different LoD and switch between them during rendering based on the distance to a chunk's centre. When chunks are optimised independently, visible seams can emerge between chunk boundaries [79]. Kerbl et al. [10] optimise the partitions separately and merge them into a hierarchical tree structure in a post-processing step. Ren et al. [77] propose to represent the scene in an octree, start the optimisation with a low number of primitives, and put them in the next

higher level of the tree whenever densifying one level. This progressive approach creates a LoD during optimisation; it is not optimised at specific resolutions, but aims to minimise reconstruction error at full resolution with a low number of primitives. Windisch et al. [79] note that the tree structure used by Kerbl et al. [10] can not be modified after creation and propose to use a Sequential Point Tree [34] to structure the scene. Their approach eliminates the need for spatial partitioning and enables large-scale optimisation of a hierarchical structure without post-processing [79].

sss

CHAPTER 3

Methodology

The main contribution of this work consists of two parts.

1. First, a coarse-to-fine optimisation that starts with optimising on low-resolution images of the scene and gradually increases resolution throughout the optimisation. A LoD is then constructed from the intermediate steps in the reconstruction in a single run.

2. Second, a selective rendering approach that renders scene details at the appropriate LoD based on the ratio of the sampling frequency at which the model was optimised, and from which it is viewed.

3.1 Coarse-to-Fine Optimisation

The coarse-to-fine optimisation schedule starts from a low resolution, e.g., one-fifth of the original resolution, serving as a supervision signal for the 3DGS optimisation. The images used as supervision signals are bilinearly downsampled from the full resolution to create an image pyramid. The number and size of the individual levels of the image pyramid, as well as the point at which the supervision signal is upsampled, are adjustable. As these parameters influence the densification heuristic, suitable combinations must be found empirically. For the experiments conducted in this thesis (see Chapter 5) a 5-layer setup with resolution $(\frac{1}{5}, \frac{1}{4}, \cdots, \frac{1}{1})$ and upsampling at regular intervals leads to a good compromise between convergence speed, reconstruction quality, and model size.

Starting from the lowest, the resolution is gradually increased until it matches the original resolution. After each upsampling step, the intermediate model is saved. After the model has been optimised for a specific number of iterations, all resulting models are merged together. Since fewer primitives are required to fit low-resolution images, fewer gradients

need to be propagated during optimisation. The coarse-to-fine optimisation process is illustrated in Figure 3.1.

To select the appropriate primitives during rendering, each primitive must store the sampling rate at which it was optimised. The sampling distance at which a Gaussian is optimised is determined by the focal length, the distance to the camera, and the resolution. Additionally, each primitive needs to store the sampling distance relative to the full resolution at which it was optimised. The distance at which a Gaussian is optimised is approximated as the distance to the closest camera from which a Gaussian is visible. In practice, the resolution, focal length, and minimal distance can be stored in a single attribute, leaving two additional attributes to store. The resulting LoD hierarchy can then be rendered with a selective rendering approach described in the next section.

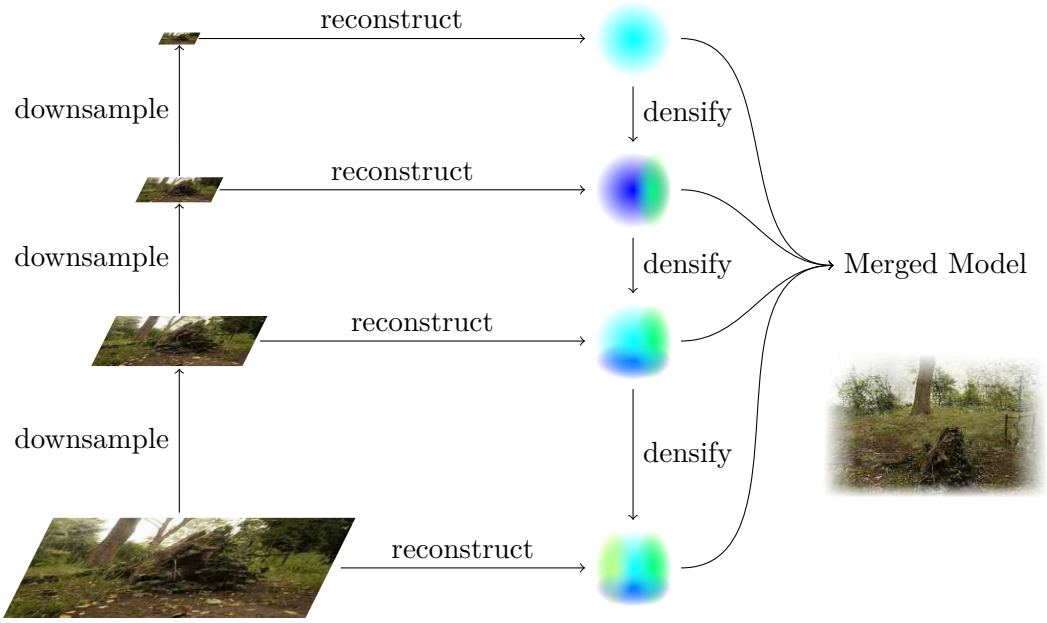


Figure 3.1: The reconstruction is progressively constructed using low- to high-frequency images as a supervision signal for the optimisation. Starting from low resolution, the supervision signal is refined as the optimisation progresses. Before upsampling, the model that fits the lower resolution is saved. Finally, the optimisation is supervised at full resolution, and the models are merged.

3.2 Selective Rendering

3DGS assumes an ideal pinhole camera model, as discussed in Section 2.1.1. In this model, the ratio between the projected size h and the focal length f, i.e. the distance from the camera centre to the image plane, is equal to the ratio of the size of an object

H at a distance d.

$$\frac{h}{f} = \frac{H}{d} \tag{3.1}$$

The projected size on a sensor, measured in pixels, therefore depends on the sensor size, the focal length f and the distance s at which the image is sampled. The sampling distance s is reciprocal to the resolution r, or the number of pixels per horizontal and vertical extents (r_x, r_y). If we assume the sensor to have a unit size, the sampling distance s is defined as follows.

$$s = \frac{1}{r} \tag{3.2}$$

The distance S at which we can sample the surface of an object with a pinhole camera at distance d from the object is therefore given by the sampling distance s of the sensor scaled by the ratio between the distance d and f.

$$S = s\frac{d}{f} \tag{3.3}$$

This relationship is illustrated in Figure 3.2

The frequency at which a surface reconstruction can be rendered depends on the image resolution, the focal length, and the distance from the camera at which it was optimised to the reconstructed surface. If any of these parameters change during rendering, the model is either over- or undersampled. When, for example, the camera moves away from the images the model was optimised on, the reconstruction frequency is lower than the frequency at which the model was optimised. In this case, more primitives than necessary are rendered, leading to aliasing errors and suboptimal performance. The selective rendering approach used in this work selects the appropriate LoD based on these parameters. A primitive is considered appropriate for rendering when the ratio of the optimisation parameters to the rendering parameters matches that of a LoD optimised for a sampling distance s; otherwise, it is discarded. The sampling distance at which the rendering occurs (S_r) is defined to be $S_r = 1$ if all parameters are the same as those on which the model was optimised. For example, if the rendering distance $\hat{d}$ to a primitive is twice as large as the distance d, at which it was optimised, the sampling distance doubles. So a model optimised at half the vertical and horizontal resolution would avoid sampling and aliasing errors in this case.

The variables of the intrinsic camera parameters used to optimise the model are depicted with a hat, i.e., the focal length f, vertical resolution r_y, and distance to a primitive d. The fraction of the sampling distances used for optimisation $\hat{S}$ and rendering S are used to decide which primitives are selected as the appropriate LoD.

$$S_r = \frac{S}{\hat{S}} = \frac{d \cdot \hat{f}_y \cdot \hat{r_y}}{\hat{d} \cdot f_y \cdot r_y} \tag{3.4}$$

A visualisation of the sampling distances $\hat{S}$, S, and S_r is shown in Figure 3.3. Given a set of models optimised at several resolutions, the primitives are selected from the model whose world-space sampling distance S_r matches the sampling distance at which it was optimised S_o, which is the reciprocal resolution of the supervision image. Because the levels are discrete within a minimal sampling distance s_{min} and maximal sampling distance s_{max}, the reconstruction sampling distance is clamped to the interval $[s_{min}, s_{max}]$ and its difference to the sampling distance of the optimisation S_o is compared to be in the interval $[-0.5, 0.5]$. The primitive is not culled if inequality 3.6 holds.

$$\text{clamp}(x, a, b) = \min(\max(x, a), b) \tag{3.5}$$

$$|S_o - \text{clamp}(S_r, s_{min}, s_{max})| \leq 0.5 \tag{3.6}$$

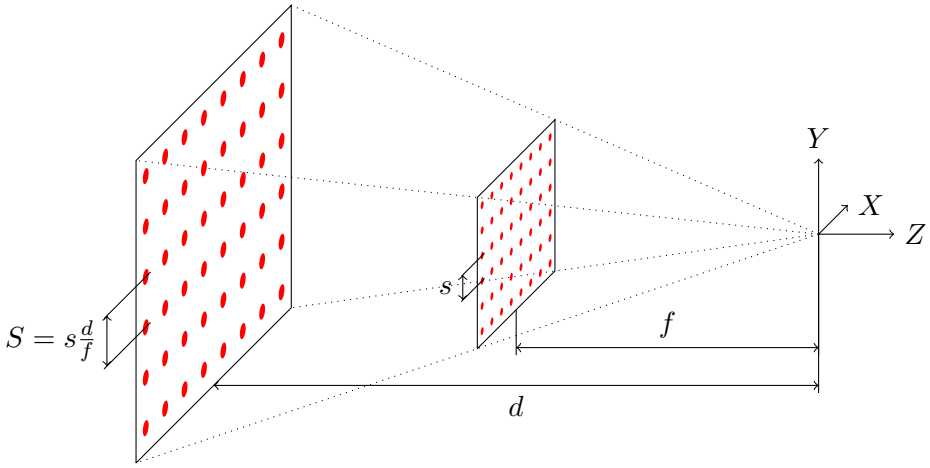


Figure 3.2: A pinhole camera with its centre at the origin, and its image plane at a distance f. Sample points are shown in red. The sampling distance s is given by the number of sample points along the height or width of the image sensor. When a surface is sampled from a distance d to the camera centre, the projected sampling distance S relates to s scaled by a factor of $\frac{d}{f}$. In this example, S is twice as large as s since d is twice as large as f.

(a) Optimisation ($\frac{1}{\tilde{S}}$) (b) Rendering (S) (c) Reconstruction $S_r = \frac{S}{\tilde{S}}$

Figure 3.3: Visualisation of three different interpretations of the sampling distance (from left to right) for two different viewpoints in the scene (top, bottom).
(a) The sampling distance during optimisation is approximated as the distance to the closest optimisation view from which a Gaussian has been optimised. Points close to the optimisation cameras have a lower sampling distance than points farther away.
(b) The resolution at which a point in space is reconstructed depends on the distance to the point from a given view. Points closer to the view camera are visible at a higher resolution than points farther away.
(c) The sampling distance at which the reconstruction is displayed is the ratio between the optimisation sampling distance (a) and the rendering surface resolution (b). When a reconstructed view is outside of the view distribution the model was optimised on (top), parts of the scene can be reconstructed at lower resolution. When the reconstructed view is close to the view distribution the model was optimised on (bottom), the sampling distance is uniform across the view, and the scene is reconstructed at a single resolution.

CHAPTER 4

Implementation

This chapter presents the actual implementation of the proposed method. The implementation of this method is built upon the baseline method implemented by the GraphDeco-Inria Group[1]. This chapter gives an overview of the existing implementations and the extensions provided in this thesis. The existing implementation is described in Section 4.1. How the implementation of the method proposed in this thesis extends the work of others is described in Section 4.2.

4.1 Existing Implementations

The source code is modularised into three major components:

- An optimisation front-end written in Python using the PyTorch [80] Application Programming Interface (API).
- A rasterisation back-end written in C++ using the Compute Unified Device Architecture (CUDA)[2] API.
- The System for Image-Based Rendering (SIBR) [81]. A real-time viewer written in C++ using the same rasterisation back-end.

Optimisation Front-end. The optimisation front-end is modularised into a scene that stores references to all supervision images and their camera poses, and a Gaussian model initialised from points in the sparse reconstruction. Before optimisation, the optimisation front-end loads the sparse reconstruction and initialises the `Scene`-object. The `Scene`-object initialises the `GaussianModel` from the point cloud and an array of

[1] `https://github.com/graphdeco-inria/gaussian-splatting`, last accessed 13.3.2026
[2] `https://developer.nvidia.com/cuda`, last accessed 13.3.2026

`Camera`-objects. The `Camera` class stores the intrinsic and extrinsic camera parameters, along with a reference to the image at the specified resolution. The `GaussianModel` represents a Struct of Arrays (SoA) storing all attributes of the GaussianPrimitives and implements splitting, cloning and pruning methods for the ADC as discussed in section 2.1.11. An illustration of the classes used is shown as a Unified Modeling Language (UML) diagram in Figure 4.1.

Forward Rendering. A front-end optimisation routine calls a parallelisable, differentiable CUDA rasterisation implementation. The rasterisation code has a forward and a backwards path. The forward path takes the model and render parameters to produce the reconstruction image. In a preprocessing step, each Gaussian primitives are projected to screen space, and Gaussians that lie outside the viewing frustum are culled. Visible Gaussians are sorted by their Z-distance to the camera. Each rendered fragment is assigned to a 16×16 tile, and the projected splats are then alpha-blended to compute the pixel colour.

Backward Pass. The backwards pass takes the loss, computed as the image-based reconstruction error, as input and returns the gradients of all visible Gaussians' parameters as described in Section 2.1.11. The parameters are then updated by multiplying each parameter by the learning rate and the gradient.

EWA-Filtering. To evaluate the anti-aliasing effectiveness of the selective rendering proposed in this thesis, it is compared with EWA-filtering. EWA-filtering, as proposed by Zwicker et al. [64] and used in Mip-Splatting [9], is integrated into the GraphDeco-Inria implementation by the maintainers. The method proposed in this thesis builds on the GraphDeco-Inria implementation, which serves as a baseline for testing and comparison.

4.2 Contributed Extensions

Building upon the existing implementations, the extensions implemented and evaluated in this thesis are as follows.

Coarse-to-fine scheduling. To optimise across multiple resolutions, the images are downscaled before optimisation. Instead of storing each image at a single resolution, this implementation maintains a 2D array for each `Camera` at each specified resolution in the `Scene`, representing the image pyramid levels for progressively optimising the model. The image for each camera is downsampled with the `PIL.Image.resize()`[3] method from Pillow[4]. As additional input parameters, the optimisation procedure takes the pyramid resolution as an array of downsampled factors and the number of iterations

[3] `https://pillow.readthedocs.io/en/stable/reference/Image.html#PIL.Image.Image.resize`, last accessed 13.3.2026

[4] `https://python-pillow.github.io`, last accessed 13.3.2026

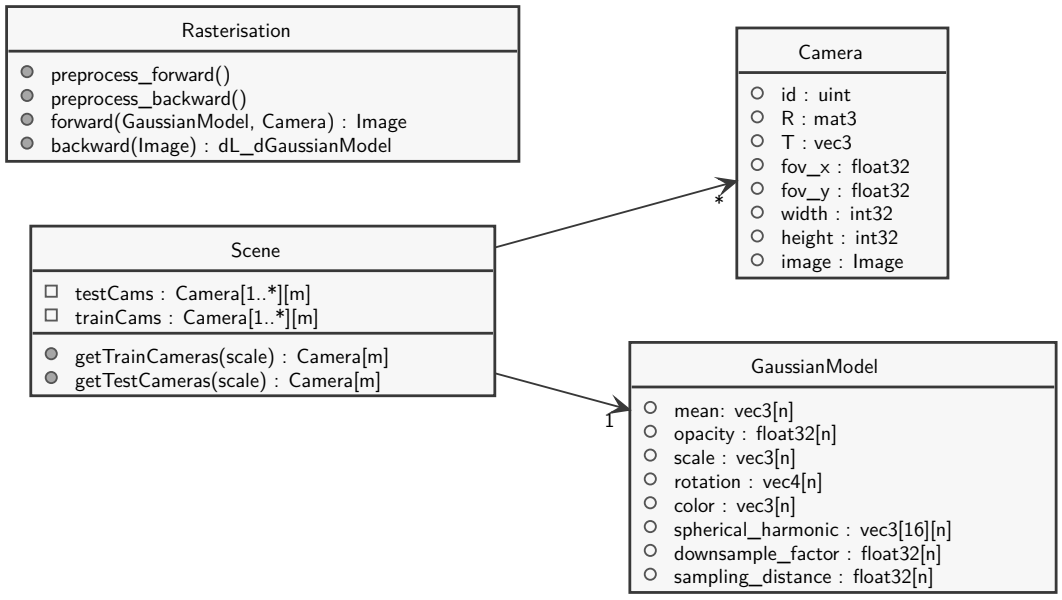


Figure 4.1: Conceptual overview of the classes used. The *Scene* holds references to the *Camera*s and the *GaussianModel*. The *Rasterisation* module provides methods for rendering an image and computing its gradient with respect to the loss.

at which to change the supervision resolution. The number of pyramid levels used and the iterations at which the supervision signal is upsampled are configured by two arrays specifying the downsample factors (e.g., $1, 2, 3, \dots$) and the iteration numbers (e.g., $3000, 6000, \dots$). The array specifying the upsample iterations must be one element shorter than the downsample factors. An overview of the optimisation process with coarse-to-fine scheduling is given in Algorithm 4.1,

Sampling Distance Computation. To selectively render the Gaussians based on their sampling distance, the `GaussianModel` stores an additional 32-bit floating-point number for each Gaussian's sampling distance $\hat{S}$, calculated as in Equation 3.3, which is updated every iteration. The view distance of a Gaussian $\hat{d}$ is approximated by the closest Euclidean distance to a camera. All distances are initialised with a large value. In each optimisation iteration, the distance $\hat{d}$ of the Gaussians that lie inside the viewing frustum is set to the minimum value of its current value and the computed distance for a particular viewpoint. This procedure avoids recomputing the set of visible Gaussians for each viewpoint and also avoids maintaining separate camera arrays for each Gaussian. For approximating the sampling distance, the vertical focal length f_y is used (see Section 2.1.1).

Algorithm 4.1: Optimisation of the Gaussian Model. In each iteration, the model is optimised by minimising loss computed from a single image. A supervision image is selected based on the current iteration and the current scale. When the current iteration equals an upsample iteration or the optimisation has reached the maximum number of iterations, the Gaussian model is saved with its current scale.

Data:

- $(I, V) \in D$: set of image/camera pairs
- θ: Gaussian model parameters
- $iterations \leftarrow 30000$
- $upsample\_iterations \leftarrow$[6k, 12k, 18k, 24k]
- $scales \leftarrow$[5, 4, 3, 2, 1]

Result: Optimized Gaussian point cloud parameters θ

$scale\_index \leftarrow 0$;
$i \leftarrow 0$;
$\hat{d} \leftarrow \infty$;
for $i \in [0, iterations)$ **do**
  $j \leftarrow i \mod |I|$;
  $\hat{I}, visibility\_indices \leftarrow \text{render}(V_j[scales[scale\_index]], \theta)$;
  $l \leftarrow \mathcal{L}(I_j[scales[scale\_index]], \hat{I})$;
  $\hat{d} \leftarrow ||V_j.position - \theta[visibility\_indices].position||$;
  $\theta[visibility\_indices].\hat{d} = \min(\theta[visibility\_indices].\hat{d}, \hat{d})$;
  backpropagate$(l, \theta[visibility_indices])$;
  if $i = upsample\_iterations[scale\_index] \vee i = iterations$ **then**
    $\theta[\dots].scale = scales[scale\_index]$;
    save(θ);
    $scale\_index \leftarrow scale\_index + 1$;
  end
  densify(l, θ);
  prune(l, θ);
end

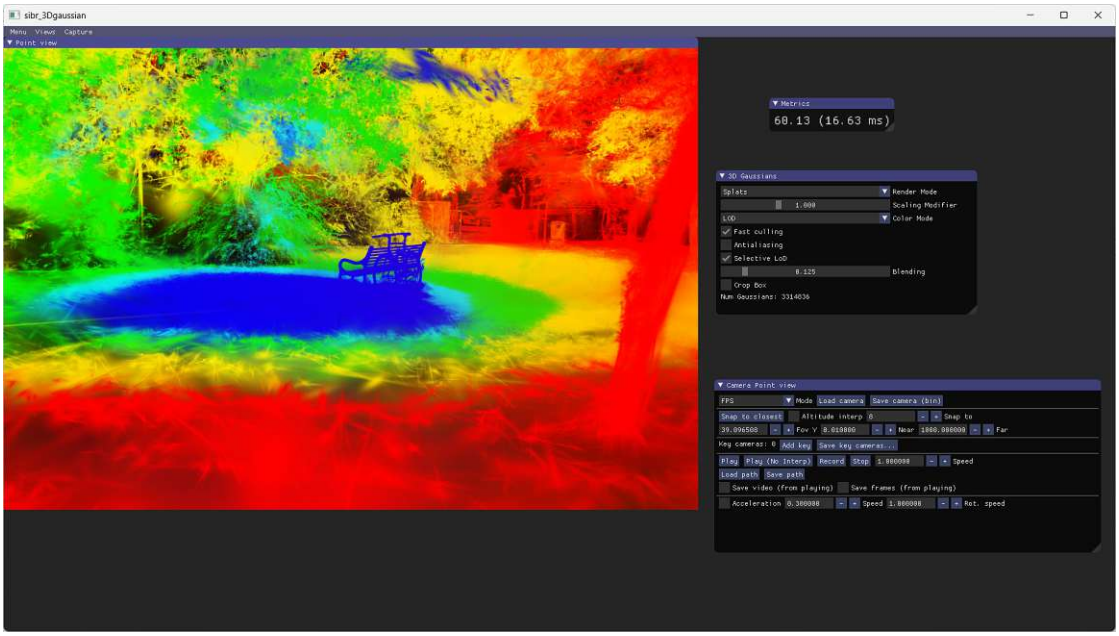


Figure 4.2: The graphical interface of the extended SIBR Viewer. The selected visualisation mode shows the Gaussians rendered with different colours for each selected LoD.

Selective Rendering. Before saving a model optimised on a particular image resolution, the Gaussians are augmented with the downsampling factor, i.e., the sampling distance of the image on which a Gaussian was optimised relative to the full resolution, represented as a 32-bit floating-point number. These two per-Gaussian attributes are saved in the resulting point cloud. After optimising the full resolution, the resulting models are merged into a single point cloud. For rendering, the additional parameter is a boolean that specifies whether to use selective rendering or not, and the resolution at which to render. After culling near primitives and before frustum-culling in the preprocessing step, the LoD is computed as in Equation 3.6 and primitives "outside" of the determined LoD are discarded.

Metric Calculation. The rendering front-end, which renders the reconstructed image for all ground-truth images in the test set, is extended to render the LoD at each resolution of the merged model. The interactive SIBR Viewer depends on the *Rasterisation* module and is used to render camera paths and record rendering performance. Each frame of a path is rendered 60 times, and the average frame time is saved together with each image. The graphical interface is extended to allow selective rendering to be toggled on and off. A particular model can be selected, and LoDs and sampling distances can be visualised as colour-coded images. A screenshot of the extended SIBR Viewer showing a visualisation of the LoD is depicted in Figure 4.2.

Evaluation Data. For evaluation, the scenes and optimisation parameters (e.g., upsampling intervals and resolutions) are manually specified and then automatically optimised for all parameter combinations. The rendered images are compared with the test sets, and qualitative metrics are computed from the results. Every 500 optimisation iterations, the views from the test set are reconstructed, the images are evaluated against the ground truth, and the results are saved. The results from each dataset are read from the output files, joined into a table, filtered for a particular parameter combination (e.g., resolution), and aggregated across all scenes.

CHAPTER 5

Evaluation

The proposed method is evaluated on widely used datasets, against the GraphDeco-Inria 3DGS implementation [7] and EWA-Filtering [9, 64], a state-of-the-art anti-aliasing method for 3DGS. The results of progressive resolution scheduling and selective resolution rendering are quantitatively and qualitatively compared. It is shown that progressive resolution reduces optimisation time and model size while maintaining reconstruction quality similar to that of the baseline. To show the effects of selective resolution rendering, experiments are conducted on low-resolution reconstructions and zoom-out trajectories, recording the number of visible primitives and the time to render a frame. While following a different anti-aliasing approach than EWA-Filtering [9], the selective rendering in this work can reduce aliasing when a model is reconstructed at lower resolutions. Additionally, the number of rendered primitives is reduced when the camera moves outside the optimisation distribution.

If not noted otherwise, all scenes from *Mip-NeRF 360* [13], two scenes from *Tanks & Temples* [14] (*Truck* and *Train*) and two scenes from *Deep Blending* [15] (*Dr. Johnson* and *Playroom*) are used for evaluation. Following the practice in Mip-NeRF 360, one-quarter-resolution versions of the images from the *Mip-NeRF 360* dataset are used. Experiments on render time and the number of rendered gaussians compared to the viewing distance are conducted on larger aerial-view datasets [16, 17]. Optimisation of the models and rendering of the reconstruction was performed on a 24-Core AMD Ryzen Threadripper 7960X CPU with 256GB RAM and a Nvidia GeForce RTX 4090 GPU with 24GB V-RAM.

5.1 Progressive Resolution Scheduling

When optimising at a lower resolution first and increasing the resolution during optimisation, the model generalises faster to higher resolution, while achieving the same reconstruction quality with fewer primitives. Since the error has to be backpropagated from fewer pixels to fewer primitives, a faster convergence is achieved.

The effects of different pyramid scaling and upsampling intervals are compared for a set of combinations. The optimised models are compared after 30 000 iterations, using the same hyperparameters. The Gaussian primitives are densified until iteration 25 000. The reported PSNR, SSIM, and LPIPS (see Section 2.1.8) are averaged across all scenes. The reported time is the total time for all scenes, expressed in minutes.

5.1.1 Hyperparameter Comparison

To evaluate the effects of different parameters on controlling the upsampling intervals and the pyramid level scales, several combinations are tested. For the resolutions, a linear $(1, 2, 3, 4, 5)$ and a quadratic $(1, 2, 4, 8, 16)$ sampling distance factor are compared. For the upsampling iterations, uniform, earlier, and later upsampling intervals are compared. Uniform intervals are upsampled every 3000 iterations (6000, 12000, 18000, 24000). Early and late upsampling use a quadratic spacing scheme for the upsampling intervals. Early upsampling has intervals closer to the beginning of the optimisation and increased spacing (2000, 5000, 11000, 19000), while later upsampling has intervals closer to the end with decreased spacing (11000, 19000, 25000, 28000).

Combinations with linearly spaced upsampling intervals using early upsampling take the longest to optimise, but yield higher reconstruction quality. Quadratic upsampling with interval spacing and late upsampling yield the smallest models, which optimise faster but yield the lowest reconstruction quality. Reconstructions with EWA filtering yield slightly larger, yet lower-quality models. The qualitative measures and the average number of primitives across all tested scenes are summarised in Table 5.1.

The combination of constant upsampling intervals and linear resolution provides a good trade-off between reconstruction quality, model size, and optimisation speed. The supervision signal for progressive resolution starts at one-fifth of the full resolution and is upsampled every 6000 iterations until it reaches full resolution at iteration 24 000. Further comparisons will use this combination.

5.1.2 Comparison to Baseline Models

Compared to the 3DGS baseline, decreases in reconstruction quality, the number of Gaussians, and reconstruction time are observed when using a progressive resolution schedule. Progressive resolution scheduling decreases the PSNR by 0.36 dB or 1%, SSIM by 0.017 and LPIPS by 0.034. The number of Gaussians produced in the progressive resolution schedule is 58.1% of the baseline. The progressive model is optimised in three-quarters of the time used for optimising the baseline. These findings are summarised in Table 5.3. The reconstruction errors, model sizes, and optimisation durations for the three datasets, with and without progressive resolution, are summarised in Table 5.2.

It is noticeable that reducing the model size by more than a quarter is rather significant. Fewer primitives have to be processed during optimisation and rendering, and smaller models are easier to store and transmit. The decrease in qualitative metrics can probably be compensated for by optimising for longer or by using just a few more primitives.

EWA-Filtering	Upsampling	Resolution	PSNR ↑	SSIM ↑	LPIPS ↓	Size ↓	Duration ↓
✗	constant	linear	27.04	0.82	0.22	$1.41 \cdot 10^6$	137.29
✗	constant	quadratic	26.73	0.81	0.24	$1.32 \cdot 10^6$	129.23
✗	early	linear	**27.25**	**0.83**	**0.20**	$1.70 \cdot 10^6$	149.45
✗	early	quadratic	27.13	0.83	0.21	$1.62 \cdot 10^6$	143.18
✗	late	linear	26.68	0.80	0.25	$1.02 \cdot 10^6$	128.14
✗	late	quadratic	25.91	0.77	0.29	$7.56 \cdot 10^5$	**117.83**
✓	constant	linear	26.84	0.81	0.23	$1.72 \cdot 10^6$	148.64
✓	constant	quadratic	26.62	0.80	0.25	$1.44 \cdot 10^6$	128.45
✓	early	linear	27.05	0.83	0.22	$2.17 \cdot 10^6$	161.47
✓	early	quadratic	26.92	0.82	0.22	$2.04 \cdot 10^6$	151.10
✓	late	linear	26.39	0.79	0.26	$2.25 \cdot 10^6$	154.57
✓	late	quadratic	25.74	0.76	0.29	$1.37 \cdot 10^6$	123.23

Table 5.1: Comparison of the reconstruction quality, model size in number of primitives, and optimisation duration in minutes for different 5-layer pyramid resolutions and upsampling intervals with and without EWA-Splatting. Upsampling intervals are either constant (6k, 12k, 18k, 24k), early (2k, 5k, 11k, 19k), or late (11k, 19k, 25k, 28k). Downsampling factors in the image pyramid are linear (1, 2, 3, 4, 5) or quadratic (1, 2, 4, 8, 16).

The speedup compared to the baseline can be explained by two factors: first, in a large fraction of the optimisation, the error has to be backpropagated from fewer pixels; and second, the number of primitives to which the error is backpropagated increases much later in the optimisation. The number of Gaussians, the measured test PSNR, and the losses during the optimisation of the *Stump* scene with 3DGS and EWA-Filtering, with and without a progressive resolution schedule, are shown in Figure 5.1.

The number of Gaussians for 3DGS converges after approximately 15 000 iterations. The model size in EWA-Filtering does not seem to converge during optimisation, as densification continues to add new primitives in the late stages. Using progressive optimisation results in a larger loss at the beginning of the optimisation, but fewer Gaussians are being densified. In the end, the model consists of fewer Gaussians, the optimisation loss is smaller, while the test PSNR is on par with 3DGS and EWA-Filtering without using progressive resolution scheduling.

Interestingly, with the combination of EWA-Filtering and progressive resolution scheduling, the number of primitives does not converge after 25 000 iterations. The loss increases in the later stages of optimisation, where densification is triggered much more strongly, and the test PSNR is slightly lower. Intuitively, EWA-Filtering and progressive resolution scheduling are orthogonal anti-aliasing approaches that might work well in conjunction, because the aliasing error should be reduced when upsampling the supervision signal to a higher resolution. However, these findings show that the two approaches together yield more primitives without significantly decreasing the reconstruction error.

Dataset	Prog.	PSNR ↑	SSIM ↑	LPIPS ↓	Model Size ↓	Duration [min] ↓
Deep Blending	✗	29.7	0.91	0.24	$1.96 \cdot 10^6$	29.93
Deep Blending	✓	29.77	0.91	0.24	$1.22 \cdot 10^6$	23.16
MipNeRF 360	✗	27.95	0.83	0.17	$2.53 \cdot 10^6$	138.43
MipNeRF 360	✓	27.48	0.81	0.21	$1.47 \cdot 10^6$	96.73
Tanks and Temples	✗	24.15	0.86	0.17	$1.36 \cdot 10^6$	23.26
Tanks and Temples	✓	23.88	0.84	0.2	$7.22 \cdot 10^5$	16.76

Table 5.2: Average error metrics, model size and total optimisation time for each dataset. Reconstruction quality is slightly lower when optimising with progressive resolution, while the model size is smaller and optimisation is faster.

	PSNR ↑	SSIM ↑	LPIPS ↓	Model Size ↓	Duration [min] ↓
3DGS	27.634	0.843	0.183	2 260 313	191.6
Prog. Schedule	27.275	0.826	0.217	1 314 356	136.7
EWA-Filtering	27.429	0.837	0.198	2 933 159	198.7
EWA-Filtering + Prog.	27.159	0.820	0.232	1 763 174	150.2

Table 5.3: Average error metrics, model size and total optimisation time aggregated over all datasets. Progressive resolution scheduling slightly decreases qualitative metrics while significantly reducing model size and optimisation time.

5.2 Selective Rendering

The approach of selectively rendering models optimised at different resolutions based on the sampling distance is evaluated by comparing reconstruction quality to a downsampled ground truth, and by measuring the number of rendered primitives and the time to render a frame while zooming out. When rendering selectively by comparing the rendering sampling distance to the optimisation, the aliasing error is reduced compared to the baseline. Since fewer primitives are required to represent a lower sampling frequency, rendering speeds up when a lower LoD is used.

5.2.1 Aliasing Errors

Selective rendering reduces errors and improves quality when rendering at a lower resolution than the model was optimised for. When rendering a full-resolution model at a lower resolution, characteristic oversampling errors occur: the image appears too bright, and thin structures are rendered too thick, as shown in the qualitative comparison in Figure 5.3. Selective rendering alleviates these errors by selecting a model that fits the target sampling frequencies. For 3DGS, the measured quality decreases with lower reconstruction resolutions, whereas EWA-Filtering shows more consistent reconstruction quality across resolutions.

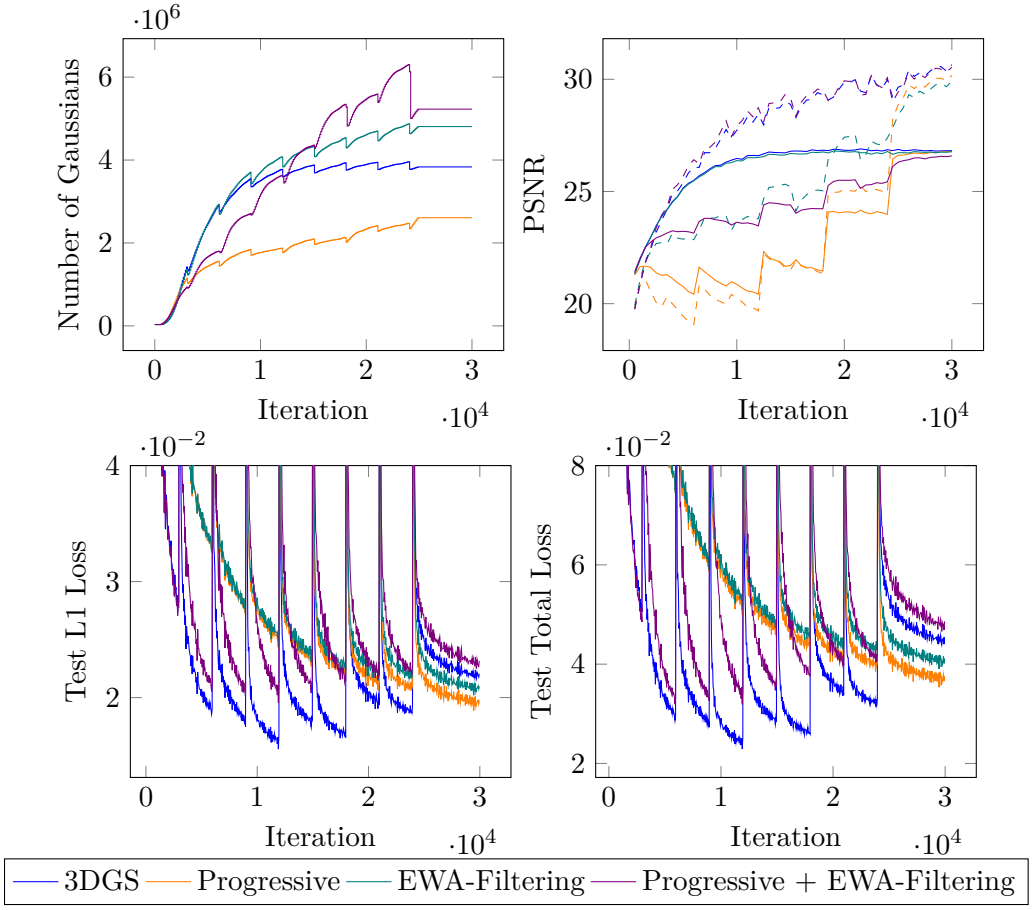


Figure 5.1: Progress of the number of Gaussians and PSNR measured against full resolution ground-truth during the course of optimisation of the *Stump* scene. The PSNR on the test set is drawn as a dotted line.

Selective rendering with and without EWA-Filtering produces slightly lower reconstruction quality, but higher quality at lower resolutions, surpassing plain EWA-Filtering at one quarter of the vertical and horizontal reconstruction resolution. EWA-Filtering processes the most Gaussians to achieve the same quality as 3DGS. The number of visible Gaussians with 3DGS and EWA-Filtering does not decrease at lower resolutions but, interestingly, slightly increases. In contrast, selective rendering processes only 40% of the Gaussians at full resolution, with a widening gap at lower resolutions. A side-by-side comparison of the PSNR and the number of rendered Gaussians is shown in Figure 5.2. Quantitative measurements are summarised in Table 5.4.

The fact that the reconstruction quality using selective rendering is almost as good as using EWA-Filtering when reconstructing a low image resolution shows that falling back to a low-resolution model is a valuable option, since it can not just be cheaper in terms of the number of processed primitives but can also be optimised much faster with fewer

optimisation iterations.

Selective	EWA-Filtering	Resolution	PSNR ↑	SSIM ↑	LPIPS ↓	Model Size ↓	Avg. Gaussians per View ↓
✗	✗	1	27.63	0.84	0.18	$2.26 \cdot 10^6$	$6.45 \cdot 10^5$
✗	✗	0.5	24.71	0.84	0.15	$2.26 \cdot 10^6$	$6.5 \cdot 10^5$
✗	✗	0.33	22.42	0.78	0.16	$2.26 \cdot 10^6$	$6.55 \cdot 10^5$
✗	✗	0.25	21.09	0.72	0.18	$2.26 \cdot 10^6$	$6.63 \cdot 10^5$
✗	✗	0.2	20.22	0.68	0.19	$2.26 \cdot 10^6$	$6.65 \cdot 10^5$
✗	✓	1	27.43	0.84	0.2	$2.93 \cdot 10^6$	$7.75 \cdot 10^5$
✗	✓	0.5	28.27	0.88	0.13	$2.93 \cdot 10^6$	$7.8 \cdot 10^5$
✗	✓	0.33	28.18	0.9	0.1	$2.93 \cdot 10^6$	$7.86 \cdot 10^5$
✗	✓	0.25	28	0.91	$9.15 \cdot 10^{-2}$	$2.93 \cdot 10^6$	$7.94 \cdot 10^5$
✗	✓	0.2	27.77	0.91	$8.45 \cdot 10^{-2}$	$2.93 \cdot 10^6$	$7.97 \cdot 10^5$
✓	✗	1	25.24	0.79	0.24	$5.38 \cdot 10^6$	$2.55 \cdot 10^5$
✓	✗	0.5	25.47	0.83	0.17	$5.38 \cdot 10^6$	$2.25 \cdot 10^5$
✓	✗	0.33	26.06	0.87	0.13	$5.38 \cdot 10^6$	$2.11 \cdot 10^5$
✓	✗	0.25	27.19	0.9	$9.98 \cdot 10^{-2}$	$5.38 \cdot 10^6$	$2.06 \cdot 10^5$
✓	✗	0.2	28.38	0.92	$7.84 \cdot 10^{-2}$	$5.38 \cdot 10^6$	$2.01 \cdot 10^5$
✓	✓	1	26.15	0.79	0.26	$7.95 \cdot 10^6$	$3.07 \cdot 10^5$
✓	✓	0.5	26.65	0.84	0.19	$7.95 \cdot 10^6$	$2.92 \cdot 10^5$
✓	✓	0.33	27.12	0.87	0.14	$7.95 \cdot 10^6$	$2.44 \cdot 10^5$
✓	✓	0.25	28.04	0.9	0.1	$7.95 \cdot 10^6$	$2.11 \cdot 10^5$
✓	✓	0.2	28.96	0.92	$7.91 \cdot 10^{-2}$	$7.95 \cdot 10^6$	$1.94 \cdot 10^5$

Table 5.4: Quantitative measures, total model size, and average number of rendered Gaussians per view at several resolutions with and without using selective rendering and anti-aliasing. Without selective rendering, the quality decreases at lower resolutions. Enabling selective rendering decreases quality at full resolution but reduces aliasing errors at lower resolutions.

5.3 Large Scale Datasets

When selecting a lower LoD, fewer primitives are rasterised; therefore, more frames can be processed per time interval. A zoom-out camera trajectory has been rendered while monitoring the number of visible Gaussians and the time per frame. Each viewpoint has been rendered 60 times, and the average of the measured timings has been recorded. As the camera moves farther from the object, the number of Gaussians increases as more of the scene becomes visible. For selective rendering, the number of Gaussians is generally lower and decreases after a point, at which parts of the scene are rendered using a low-resolution model.

To see the effects of selective rendering when reconstructing a scene at a viewpoint from where it has not been optimised, the reconstructed scenes are rendered along a zoom-out trajectory. The *Bicycle* scene from Mip-NeRF 360 [13], and three aerial photography scenes have been compared. These aerial view scenes are captured with a drone and contain more images than the Mip-NeRF 360 scenes [13]. The *Building* scene has been published in the Mill19 dataset as part of Mega-NeRF [16] and contains 1949 images. The *ArtSci* scene has been published in the UrbanScene3D dataset [17] and contains 3620 images. The *Wolf an der Au* scene has been recorded by Tobias Batik from the Virtual and Augmented Reality group and contains 416 images. The number of images per scene and the image resolution used to optimise the NVS is shown in Table 5.5.

Scene	Num. Images	Resolution ($w \times h$)
Bicycle (white background)	193	1237×822
Wolf an der Au	416	2016×1134
Building	1949	2048×1532
ArtSci	3620	1500×1000

Table 5.5: Number of images in the evaluated scenes and the downsized image resolution used for NVS.

Camera poses and the point clouds for initialisation are extracted with SfM [29]. Sizes of the image sets used, and timings taken for reconstructing the sparse model are summarised in Table 5.6. To match the input resolutions of the other scenes, the input images have been downscaled to approximately 2000 pixels horizontally. Following the practice used in Mip-NeRF 360 [13], every 8th image is used for the test set, and the remaining for optimisation. To reduce the number of floaters in the sky of the *Bicycle* scene, it has been optimised on a white background. The *Wolf an der Au* scene has been optimised for 40 000 iterations. For the larger scenes *Building* and *ArtSci*, the positional learning rate is reduced to 10%, and the scaling learning rate to 20%. The total number of optimisation iterations and the points at which upsampling and densification occur are doubled.

Scene	Feature Extraction	Feature Matching	Sparse Reconstruction
ArtSci	10.881	6.441	3354.293
Wolf an der Au	2.650	0.711	11.226

Table 5.6: Duration of the steps of the SfM process in minutes. The sparse reconstruction step, including iterative triangulation and bundle adjustment, takes the most time.

5.3.1 Test-set Evaluation

The model sizes and reconstruction errors, as measured by the test-set error metrics (PSNR, SSIM, LPIPS), are summarised in Table 5.7. Consistent with measurements on the smaller object-centric scenes, the reconstruction quality is slightly lower, while the number of primitives is significantly lower. Compared to the smaller object-centric scenes, the overall reconstruction quality in large aerial photography scenes is lower, with an approximate PSNR of 19 dB. This might be the case for several reasons.

The aerial scenes have fewer overlapping images since they cover a larger area than the object-centric scenes, where the object of interest is always in view. Therefore, the number of images from different viewing angles a single primitive receives gradients from is lower. For the large scenes, the number of iterations is doubled, while the number of images is almost 20 times that of the smaller scenes. That means the optimisation of a large-scene-model with 3000 images sees a single image only one-tenth as often as the smaller datasets with 150 images. The quality of the test set can be improved by selecting appropriate hyperparameters. Finding the best hyperparameters is out of scope for this evaluation.

Scene	Selective	SSIM	PSNR	LPIPS	Duration	Model Size	Visible
Bicycle	✗	0.75	24.56	0.21	44.20	$1.13 \cdot 10^7$	$1.73 \cdot 10^6$
Bicycle	✓	0.66	22.77	0.31	23.88	$2.25 \cdot 10^7$	$5.23 \cdot 10^5$
Building	✗	0.67	21.23	0.41	57.61	$4.25 \cdot 10^6$	$4.34 \cdot 10^5$
Building	✓	0.64	20.93	0.43	50.47	$1.62 \cdot 10^7$	$3.58 \cdot 10^5$
ArtSci	✗	0.56	20.95	0.51	44.52	$5.21 \cdot 10^6$	$1.24 \cdot 10^5$
ArtSci	✓	0.54	20.89	0.52	45.02	$2.01 \cdot 10^7$	$1.02 \cdot 10^5$
Wolf an der Au	✗	0.39	18.56	0.44	45.31	$8.02 \cdot 10^6$	$8.87 \cdot 10^5$
Wolf an der Au	✓	0.36	18.30	0.47	39.03	$2.29 \cdot 10^7$	$5.85 \cdot 10^5$

Table 5.7: Comparison of the reconstruction quality metrics, optimisation duration in minutes, and model size in number of primitives of the large-scale scenes with and without using selective rendering.

5.3.2 Zoom-out Trajectories

When rendering the scene along a zoom-out trajectory, selective rendering uses fewer Gaussians at greater distances. The reduction in the number of processed primitives is reflected in the time it takes to render a frame. While initially the number of Gaussians increases as larger parts of the scene become visible, the number drops after a threshold where Gaussians become smaller than a single pixel and are discarded from further processing. The frame time, in contrast, does not decrease at larger distances at the same rate, as the number of Gaussians increases. While selectively rendered scenes exhibit the same patterns, the effect is much smaller than without selective rendering.

The frametime increases even when the number of processed primitives decreases. This is likely due to the tile-based processing of the Gaussians. As many Gaussians fall within a few tiles, the process is not effectively parallelised and more, GPU capacity is idling. The *Bicycle* scene is an extreme case, where many small primitives are clustered in the centre of the scene. When zooming out, the frametime increases significantly while the number of primitives decreases slightly. The number of Gaussians and the time each frame took to render compared to the camera distance to the origin are shown in Figure 5.4. Images rendered at increasing distances are shown in Figure 5.5.

A side-by-side comparison of the selectively rendered images and the full resolution 3DGS baseline is shown in Figure 5.6. Again, aliasing effects are much attenuated with selective rendering but clearly visible in the images produced by the baseline.

The crane visible in the *ArtSci* scene is more transparent in comparison. This is likely the case because the model did not fully converge, and the primitives at this location received gradients only from a few supervision images, which is compensated for by the over-opaque primitives in the 3DGS baseline. The windows in the buildings are very blurry in the selective rendering, but the primitives are more isotropic in size. The same building, reconstructed at full resolution, exhibits high-frequency Gaussians on an overly under-reconstructed facade.

In the *Wolf an der Au* scene, the vegetation exhibits a very uneven brightness across the image, and thin, regular structures are blurred out. With selective rendering, vegetation has a more natural appearance, and thin structures are more easily distinguishable. These results show that selective rendering avoids aliasing errors in out-of-distribution viewpoints that can be rendered more efficiently than full-resolution models.

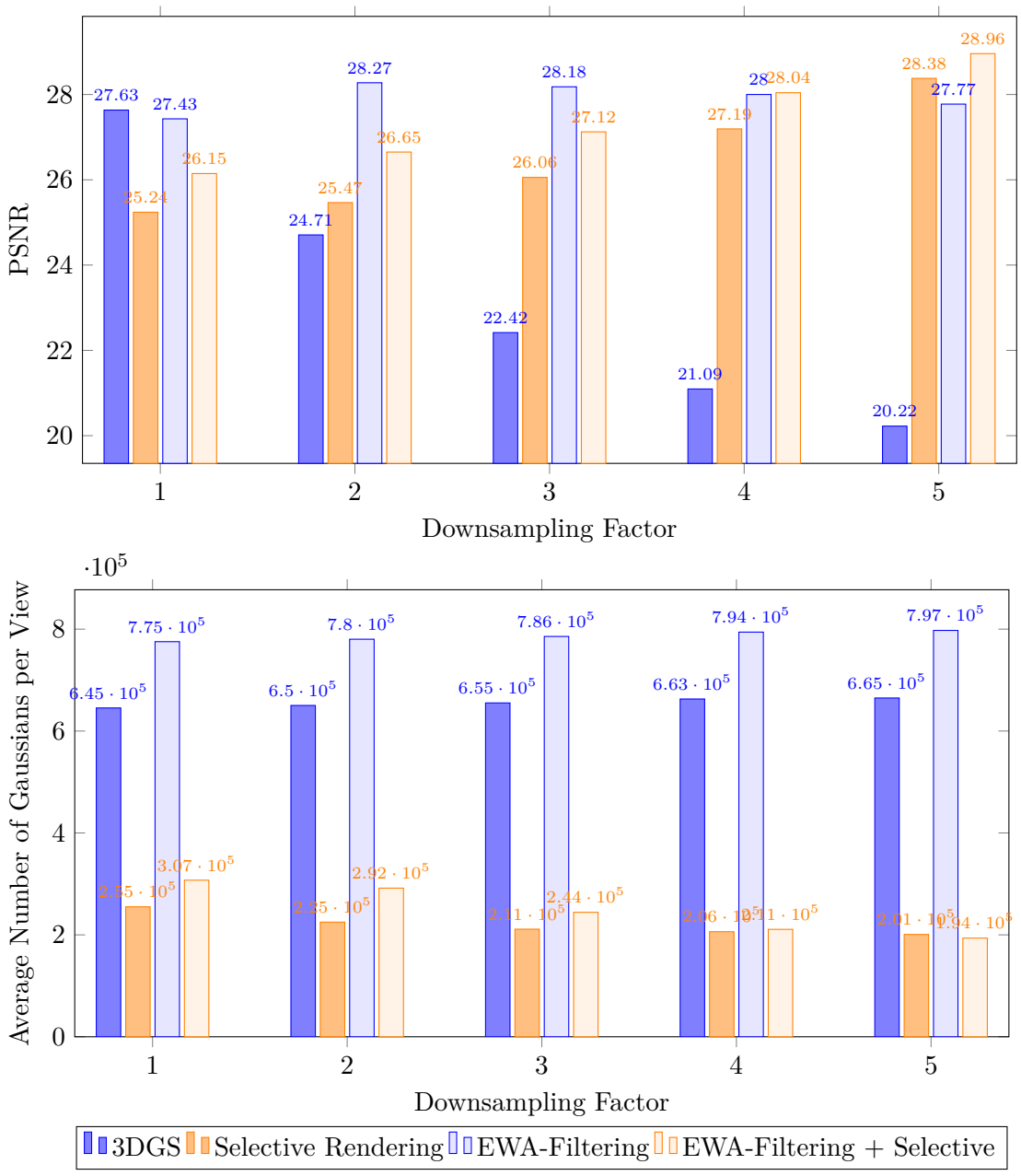


Figure 5.2: Comparison of the average PSNR and number of Gaussians per view at different fractions of the original resolution. The quality of the baseline decreases at lower resolutions, while it increases with selective rendering. For selective rendering, the number of primitives decreases with resolution and is generally lower than with 3DGS and EWA-Filtering, where the number of primitives remains constant across resolutions.

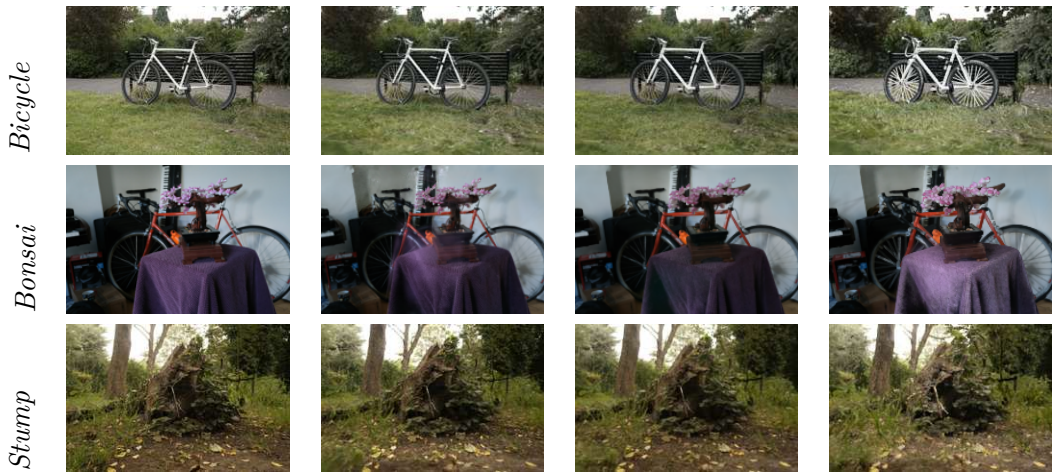
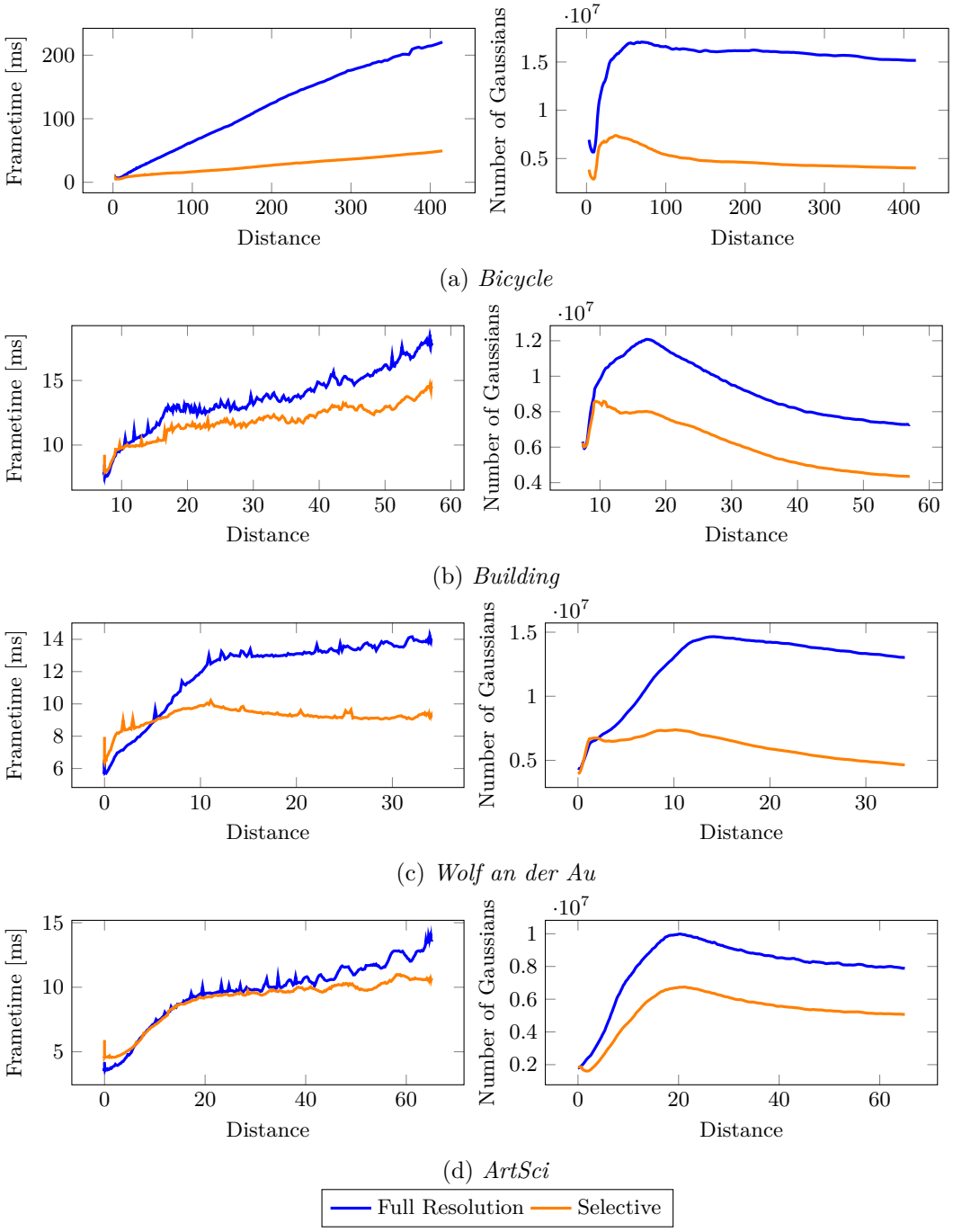


Figure 5.3: Qualitative comparison of the ground truth images and the renderings at one-quarter resolution from the test set of different scenes. Selective rendering (2nd column) effectively reduces aliasing errors, similar to EWA-Filtering (3rd column), which are present in renderings from 3DGS (4th column), where thin structures are rendered too thick and too bright.



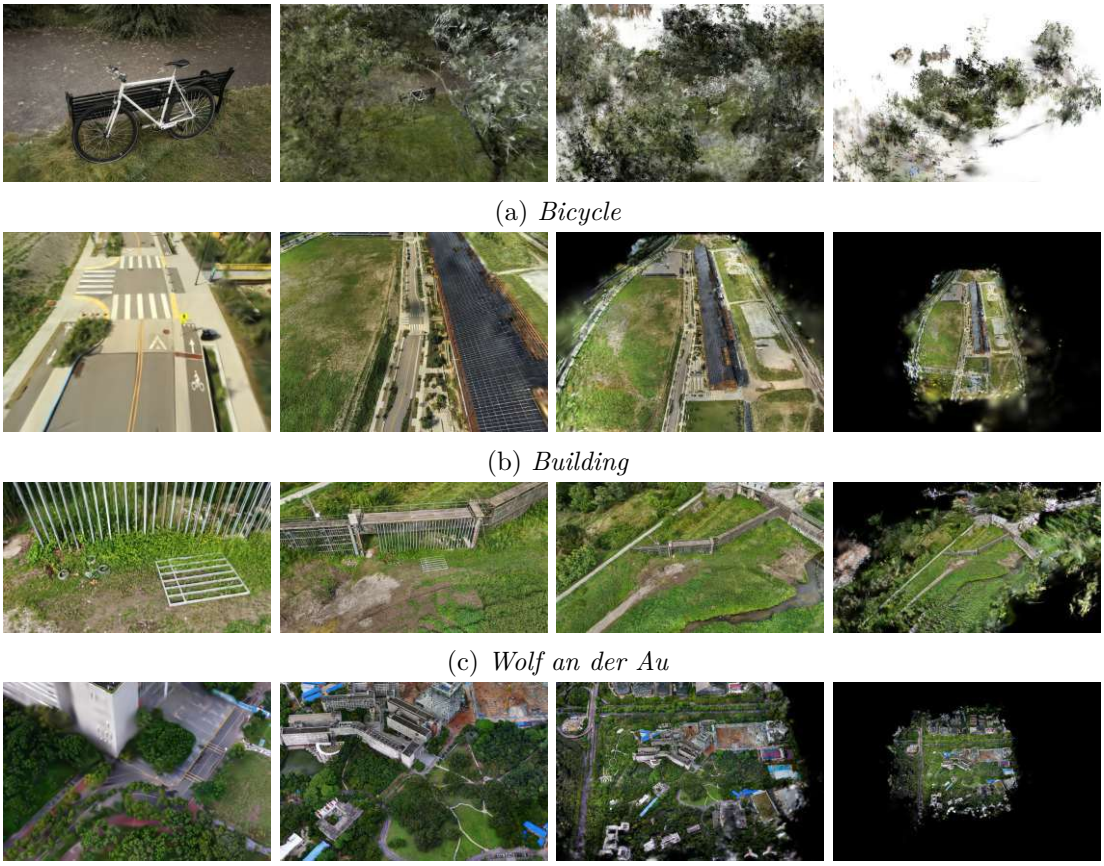
(a) *Bicycle*

(b) *Building*

(c) *Wolf an der Au*

(d) *ArtSci*

Figure 5.4: Number of rendered Gaussians and the time a frame takes to render in milliseconds versus the camera distance to the origin in a sequence of images when zooming out. Selective rendering reduces the number of rendered Gaussians and takes less time to render a frame when the scene is viewed from far away compared to the 3DGS implementation.



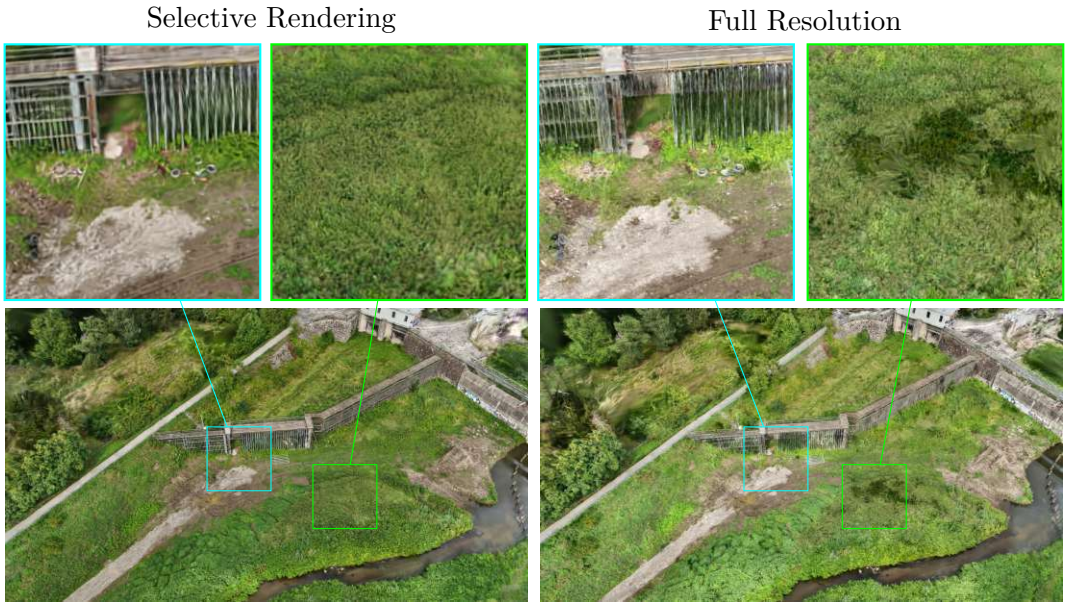
(a) *Bicycle*

(b) *Building*

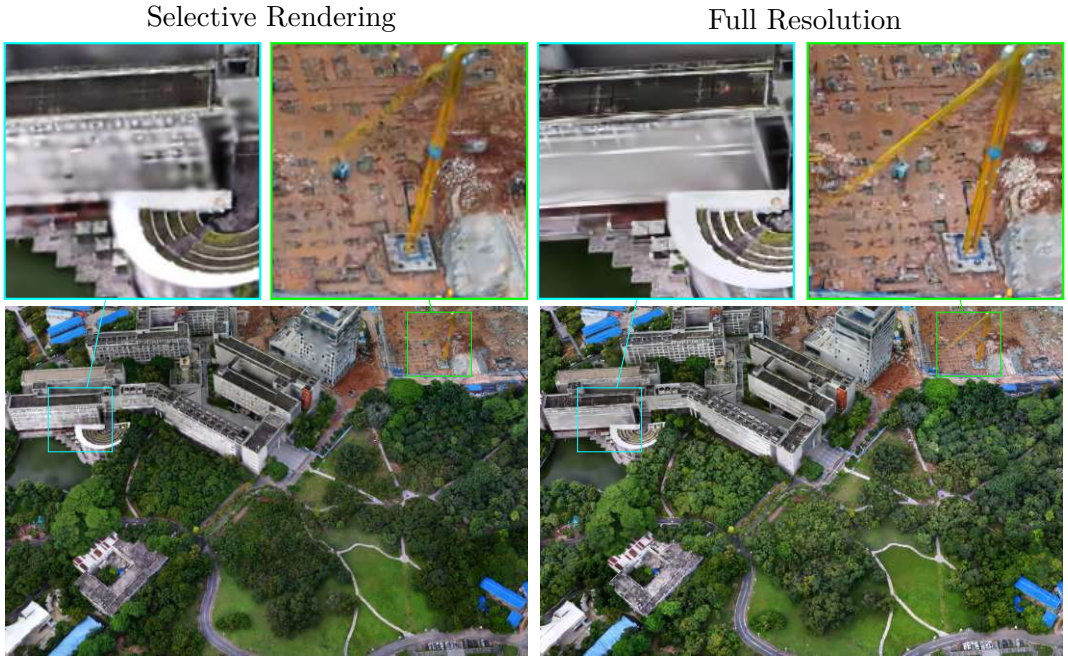
(c) *Wolf an der Au*

(d) *ArtSci*

Figure 5.5: Renderings using selective rendering of two scenes on a zoom-out trajectory at four distances. The four images from left to right are taken at 0%, 16%, 33% and 66% of the trajectory.



(a) *Wolf an der Au*



(b) *ArtSci*

Figure 5.6: Side-by-side comparison of selective rendering (left) against the full resolution 3DGS baseline (right) showing a far view of the *Wolf an der Au* (a) and *ArtSci* b scenes. Enlarged regions highlight areas of interest.

CHAPTER 6

Discussion

A method for fast optimisation of a compact 3D Gaussian model by progressively optimising at different sampling rates and using intermediate models as levels in a LoD hierarchy has been presented and evaluated. The method has been compared in terms of reconstruction error, rendering speed, and the number of processed primitives. A quantitative comparison of reconstruction quality at lower resolutions and the rendering speed on zoom-out trajectories has been conducted. This chapter discusses the results in the context of the original research questions, explores applications and limitations, and outlines avenues for future work.

Q1) How does coarse-to-fine scheduling impact the trade-off between convergence speed, reconstruction quality, and model size compared to baseline 3DGS? As shown in Section 5.1.2, progressive resolution scheduling significantly reduces the number of primitives in the full-resolution model from 2.26 million to 1.31 million by 41.9%, averaged over the datasets tested. The coarse-to-fine optimisation processes 30k iterations 28.6% faster than the baseline. The measured reconstruction error is slightly larger than the baseline's.

Although the differences are small (−0.36 dB PSNR, −0.017 SSIM, 0.034 LPIPS), the progressive schedule used in this work did not improve reconstruction quality relative to the baselines on full-resolution renderings. While, in principle, the optimisation of the baseline 3DGS method could be stopped earlier, the LoD hierarchy, as a 'byproduct' of the optimisation, can be used to be selectively rendered.

Q2) How can coarse-to-fine optimisation enable on-the-fly LoD hierarchy construction, and what are the implications for memory requirements? The method for selectively rendering the LoD does not rely on any post-processing beyond merging the models that have already been produced during the optimisation. Aggregating existing Gaussians from a full-resolution model to construct a LoD in post-processing

is avoided, saving time for longer optimisation or faster previews. The selection of the appropriate LoD is achieved merely by saving the sampling distance at which a Gaussian was optimised and comparing it with the sampling rate at which the novel view is synthesised. This formulation requires two additional attributes per model primitive but does not require approximating the projected size of each visible Gaussian, which would not necessarily match the sampling rate at which a Gaussian has been optimised.

As shown in Section 5.2, the error on low-resolution and zoomed-out renderings decreases and gets lower compared to the baseline. While the individual models optimised with a progressive resolution schedule are more compact than those without, the overall size of a merged model is larger than that of a single-resolution-optimised model. The full model must be present in GPU memory during rendering. The effect of faster rendering and reduced aliasing is traded for larger storage and GPU memory requirements.

The optimisation can balance between smaller models and higher quality by adjusting the resolution of the supervision image pyramid and the times at which to upsample the image during optimisation. When the pyramid is upsampled later during optimisation, fewer Gaussians are produced.

Q3) How can sampling-distance-based selective rendering adapt reconstruction quality and model size to different rendering resolutions and viewpoint distances, and what are the trade-offs in terms of aliasing, rendering speed, and primitive count? Selective rendering switches to a low-resolution model when necessary, further reducing the number of rendered primitives and reducing aliasing effects in the image. In particular, for far-views, the reconstruction exhibits less aliasing, as shown in Section 5.3.2. Selective rendering leads to better generalisation by the model to far viewpoints outside the optimisation distribution. Additionally, the reduced primitive count increases the rendering speed. Drawbacks are that the selective rendering approach is biased towards the lower resolutions and adds a visible seam along the borders of a LoD.

Selective rendering and EWA-filtering are two orthogonal anti-aliasing approaches. EWA-filtering limits the sampling frequency by applying a screen-space dilation to enforce that each Gaussian covers at least one pixel. The experiments conducted in Section 5.2.1 revealed that the loss is generally higher with than without EWA-Filtering, which, in turn, triggers densification more strongly as shown in Figure 5.1. EWA filtering produces smooth results, whereas slight Aliasing errors are visible near LoD boundaries in selective rendering and exhibit a visible seam when moving the camera. On the other hand, EWA-Filtering is not effective at reducing the number of primitives at lower resolutions or for viewpoints far from the optimisation distribution. Both methods, used in conjunction, lead to decent reconstruction quality while effectively reducing the number of rendered primitives at far viewpoints.

6.1 Limitations

High memory requirements to render the LoD hierarchy. The method requires all Gaussians from the entire LoD hierarchy to be in GPU memory during rendering, which limits its applicability on devices with limited memory. For example, as shown in Section 5.2, the total number of primitives averaged over the datasets tested for a merged model with 5.38 compared to a full-resolution 3DGS model with 2.26 million primitives is 2.3 times larger.

Narrow world-space LoD spacing. As shown in Section 5.1.1, the method presented yields better reconstruction quality when the sampling distance of the pyramid levels increases linearly. When projected to 3D, this results in a rapid decrease in the LoD with increasing distance. While the optimisation supports different scaling of the pyramid levels, the current implementation of the selective rendering only supports linear falloff. While this is not an issue for the datasets tested in this work, for even larger scenes and higher supervision image resolution, more levels with different scales will be required.

Occlusion-unaware distance heuristic. As discussed in Chapter 3, the heuristic for determining the lowest sampling distance at which a Gaussian is optimised is based on the distance to the closest camera in which the Gaussian lies within the camera frustum. This heuristic does not check whether a Gaussian is occluded. Occluded Gaussians would distort the estimation of the sampling distance. A Gaussian close to a Camera does not necessarily contribute to the image colour from this viewpoint. If a Gaussian is occluded in its closest camera, the spatial distance can be underestimated, and the sampling distance can be overestimated. This is hypothesised to bias the LoD selection towards lower resolution levels. Investigating this effect is left for future work.

Seams between LoD boundaries When the camera is moved, a seam between the boundaries of the single levels becomes visible. Initial experiments with blending between the LoD by allowing an overlap and adjusting the Gaussians' opacities to fade in and out showed that visible seams can be reduced. While blending between levels reduces the seam, it increases the number of rendered Gaussians, especially when the reconstruction is close to an optimisation viewpoint.

6.2 Future Work

Frequency Aware Upsampling It has been shown that using a coarse-to-fine schedule for optimisation improves the model's convergence. The LoD reduces aliasing by switching to a model that best fits the reconstruction frequency. Anti-aliasing is not explicitly incorporated into the optimisation process. Due to the point-sampling practice of 3DGS, the optimisation is not forced to produce Gaussians that fall within the sampling frequency of the supervision image.

In the original 3DGS implementation [7], when a viewpoint is reconstructed at a higher resolution than the model was optimised for, the Gaussians' opacities are too low, they do not cover the full image, and the image is overall too dark. This is the case because the Gaussians are evaluated only on a single point at the centre of a pixel [68]. The covariance of a Gaussian can, in this case, be smaller than covering a full pixel as long as it covers its centre points, representing a higher frequency as present in the supervising image. This behaviour shows that optimising on a downsampled image is very different from optimising on a low-pass-filtered image.

Using a coarse-to-fine resolution schedule is a key feature for accelerating convergence by supervising the optimisation with fewer pixels. While still providing a good initialisation for the higher-resolution models, the loss after an upsampling step could perhaps be smoother, leading to higher-quality reconstructions. Closing the gap in the discrepancy between subsampling and low-pass filtering is an interesting avenue for future work.

One method not tested in this work is 3D smoothing, as proposed in Mip-Splatting [9], for zoom-in views to broaden the Gaussians' covariance and limit their frequency in world-space. Using this world-space regularisation, together with coarse-to-fine scheduling, should lead to lower optimisation errors after upsampling. How far these two methods support each other when used in conjunction shall be examined in future work.

Selectively Optimising the Model The LoD is constructed from several models, where each level serves as an initialisation for the next, but they are otherwise optimised independently. When dealing with large scenes, it would make sense to optimise the low-resolution model for far views, which can, in turn, support the low-resolution optimisation of another close view.

In the current implementation, selective rendering is not used during optimisation, but is used for the reconstruction of a merged model. Having the whole model in memory during optimisation incurs a significant memory cost that can easily exhaust available resources. Keeping the full model hierarchy with all gradients in memory during optimisation exhausted the GPU memory during experimentation. This problem could be addressed from several directions.

1. Further reducing the size of individual LoD.
2. Reusing primitives across several LoD to have a smaller LoD hierarchy.
3. Using out-of-core rendering to only fetch required parts of the model into GPU memory.

Reducing the primitive count of individual LoD. A primary inspiration for this work was DashGaussians [72], which estimates the number of iterations for upsampling based on the spectral energy difference between the downsampled and full-resolution images. Other work has explored ways to reduce the number of Gaussians required to

reconstruct a scene, through advanced ADC methods. E.g., the method presented in RAIN-GS [75] regularises the primitives to have a more uniform spatial distribution, leading to better utilisation of the primitives and, hence, smaller models. The approach presented in Mini-Splatting [70] introduces a simplification step by keeping Gaussians with the maximum contribution to the pixel colour. Integrating such regularisations and simplification steps can further reduce the model size while keeping the reconstruction error low. How such approaches work in conjunction with a coarse-to-fine schedule is a subject for future investigation.

Out-of-core rendering. The LoD could be represented in a nested hierarchy, such as a tree, where higher-frequency primitives depend on lower-frequency primitives. The nesting could be either per-primitive, e.g., in a binary tree, or spatial, e.g., in an octree. That way, the required primitives could be selected out-of-core, allowing for more efficient utilisation of device memory. Exploring how the several models can be represented in a way that enables fine-grained LoD selection and primitive streaming is left for future work.

Reusing Gaussians over several LoD. It would be interesting to investigate ways to optimise a model that can reuse Gaussians additively rather than having independent models to switch between. The approach discussed leads to independent levels, each rendered exclusively. An additive approach would yield the lowest LoD, with all higher levels stacked on top to match the required detail. How an additive hierarchy could work for a coarse-to-fine optimisation shall be investigated in future work.

In a 2D image, the levels of a DoG pyramid contain positive and negative brightness, and can simply be added. In a 3D radiance field, this could lead to occlusion problems, and a negative radiance value would violate the energy-conservation property of the radiative transfer. A workaround would be to optimise and render each frequency band separately, then combine the resulting images in a post-processing step. Another possibility for reusing primitives would be to track, for each Gaussian, the sampling distance at which it can be used.

Dataset for evaluation of out-of-distribution generalisation. While lower-resolution rendering provides some insight into how the model behaves when the sampling frequency is changed or reduced, more care must be taken when comparing the model's reconstruction quality and generalisation capability across zoomed-out views. First, a ground-truth dataset containing views from multiple distances must be used for evaluation. This evaluation could be run as a 3-split cross-validation where the close and far views are split in three ways:

1. A split containing mostly close views in the optimisation set, and mostly far viewpoints in the test set.
2. A split containing far and close viewpoints, the other way around.

3. A balanced split containing approximately the same ratio of far and close sets.

Such a dataset could be recorded either by a drone flying at different heights or by using different recording carriers, e.g., ground level, drone, plane, satellite. New challenges arise in optimising scenes across images taken at significantly different scales, e.g., when image features must be matched despite differing appearances or when certain features only exist at some scales due to occlusion. Collecting such a dataset and comparing it with the discussed method would be an interesting challenge for future work.

Wider comparison to additional state-of-the-art methods. The method was compared to 3DGS [7] with and without EWA-Filtering [9] for anti-aliasing. Another method proposed by Yan et al. [8] is similar to the one presented, as it constructs a LoD during optimisation. The difference lies in how the LoD is constructed during optimisation and in the heuristic used to select the LoD during rendering. Based on the projected size in pixels, primitives are aggregated to construct a lower LoD and selected if a Gaussian covers a specific number of pixels. In contrast, the method presented selects the appropriate LoD by approximating the sampling distance at which each Gaussian was supervised. How these two and other related methods compare is subject to future investigations.

6.3 Conclusion

This thesis explored a way to build a LoD hierarchy during optimisation that can be rendered selectively based on comparisons between the optimisation and rendering sampling distances. This LoD hierarchy should represent a step towards more efficient optimisation and rendering of larger reconstructions. The underlying hypothesis was that coarse-to-fine scheduling can enable faster optimisation while producing a LoD hierarchy that can be used to efficiently render larger scenes while avoiding aliasing errors. Three research questions were formulated to evaluate the trade-offs of the coarse-to-fine optimisation schedule, assess the implications of the constructed LoD, and quantitatively analyse its impact on rendering efficiency and reconstruction error.

To answer the three research questions, the model size and convergence speed were compared to the baseline method. Renderings at several resolutions were quantitatively compared in terms of model size and error measures, and qualitatively compared for visible aliasing errors. Comparisons were done against another state-of-the-art anti-aliasing method on datasets widely used to test NVS methods. To show the method's effect on out-of-distribution views, novel views were rendered along a zoom-out trajectory, with rendering speed and the number of processed primitives recorded. Different-sized datasets were used, including one containing more than 3000 images.

Key findings are that coarse-to-fine scheduling significantly reduces the number of primitives necessary to represent the scene while only slightly increasing the reconstruction error. Selective rendering further reduces the number of required primitives as the

viewpoint moves farther from the scene, thereby accelerating rendering. Less optimisation time and fewer primitives are required for far-view optimisation than for the EWA-filtering anti-aliasing method that extrapolates from a high-resolution model. A current limitation is that the full hierarchy must be present in GPU-memory during rendering, motivating future work on out-of-core rendering.

Due to the model's generalisability across different rendering resolutions and distances, a high-quality overview of a large-scale reconstruction can be provided efficiently, even when the model has been optimised from closer viewpoints. This makes the discussed approach a step towards interactive large-scale reconstructions. While the actual requirements for remote-sensing applications are diverse, this work is a contribution towards pushing the scale barrier of novel view synthesis towards larger areas and applications that enable more interactive workflows from recording to reconstruction in the future.

Overview of Generative AI Tools Used

Grammarly was used to correct spelling. DeepL was used to translate and to improve the readability of some sentences by translating them back and forth between German and English. Mistral LeChat was used for proofreading and improving readability.

List of Figures

List of Tables

List of Algorithms

Acronyms

$\mathcal{L}_1$ Mean Absolute Error. 12

3DGS 3D Gaussian Splatting. xi, xiii, 2–4, 14–19, 21, 33, 42, 47, 49, 51, 54

ADC Adaptive Density Control. 16, 18, 19, 28, 53

API Application Programming Interface. 27

BA Bundle Adjustment. 9

CUDA Compute Unified Device Architecture. 27, 28

DoG Difference of Gaussians. 9, 53

EWA Elliptical Weighted Average. xi, xiii, 14, 17, 28, 33, 35, 37, 54, 55

GPU Graphics Processing Unit. xi, xiii, 2, 10, 19, 42, 51, 52, 55

LoD Level of Detail. xi, xiii, 2, 3, 9–11, 17–23, 31, 49–54, 59

LPIPS Learned Perceptual Image Patch Similarity. 13

MAE Mean Absolute Error. 12, 65

Mip multum in parvo. 9

MLP Multilayer Perceptron. 13

MSE Mean Squared Error. 12

NeRF Neural Radiance Field. 13, 14

NVS Novel View Synthesis. 1–5, 11, 14, 40, 54, 61

PSNR Peak Signal to Noise Ratio. 12

RANSAC RANdom SAmple Consensus. 9

SfM Structure from Motion. 9, 13–15

SIBR System for Image-Based Rendering. 27, 31, 59

SIFT Scale Invariant Feature Transform. 9

SoA Struct of Arrays. 28

SSIM Structural Similarity Index Measure. 12

UML Unified Modeling Language. 28

List of Symbols

K intrinsic camera parameters. 6

P projection matrix. 6

R rotation matrix representing the camera orientation in world-space. 6

S_o sampling distance of the supervision image during optimisation. 24

S_r sampling distance of the rendering. 23, 24

S sampling distance of a projected rendered surface. 23, 24

$\hat{S}$ sampling distance of a projected optimised surface. 23, 24

$\hat{d}$ distance of a single Gaussian to the closest optimisation camera centre. 23

$\vec{c}$ principal point on the image plane. 5

$\vec{t}$ translation vector representing the camera position in world-space. 6

d distance of a single Gaussian to the rendering camera centre. 23, 24

f focal length of a camera. 5, 6, 22–24

r_x horizontal resolution of the rendered image. 23

r_y vertical resolution of the rendered image. 23

s_{max} the largest sampling distance s used to optimise the model. 24

s_{min} the smallest sampling distance s used to optimise the model. 24

s sampling distance of the rendered image (reciprocal resolution). 23, 24, 67

Bibliography

[1] N. Ejaz and S. Choudhury, "Computer Vision in Drone Imagery for Infrastructure Management," *Automation in Construction*, vol. 163, p. 105418, July 2024.

[2] J. Fernandez Galarreta, N. Kerle, and M. Gerke, "UAV-based urban structural damage assessment using object-based image analysis and semantic reasoning," *Natural Hazards and Earth System Sciences*, vol. 15, pp. 1087–1101, June 2015.

[3] A. Vetrivel, M. Gerke, N. Kerle, and G. Vosselman, "Identification of damage in buildings based on gaps in 3d point clouds from very high resolution oblique airborne images," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 105, pp. 61–78, July 2015.

[4] M. M. Nowak, K. Dziób, and P. Bogawski, "Unmanned Aerial Vehicles (UAVs) in environmental biology: A Review," *European Journal of Ecology*, vol. 4, pp. 56–74, Jan. 2019.

[5] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, *NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis*, pp. 405–421. Springer International Publishing, 2020.

[6] Sara Fridovich-Keil and Alex Yu, M. Tancik, Q. Chen, B. Recht, and A. Kanazawa, "Plenoxels: Radiance Fields without Neural Networks," in *CVPR*, 2022.

[7] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3D Gaussian Splatting for Real-Time Radiance Field Rendering," *ACM Transactions on Graphics*, vol. 42, July 2023.

[8] Z. Yan, W. F. Low, Y. Chen, and G. H. Lee, "Multi-Scale 3D Gaussian Splatting for Anti-Aliased Rendering," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 20923–20931, IEEE, June 2024.

[9] Z. Yu, A. Chen, B. Huang, T. Sattler, and A. Geiger, "Mip-Splatting: Alias-free 3D Gaussian Splatting," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19447–19456, June 2024.

[10] B. Kerbl, A. Meuleman, G. Kopanas, M. Wimmer, A. Lanvin, and G. Drettakis, "A Hierarchical 3D Gaussian Representation for Real-Time Rendering of Very Large Datasets," *ACM Transactions on Graphics*, vol. 43, July 2024.

[11] Y. Liu, C. Luo, L. Fan, N. Wang, J. Peng, and Z. Zhang, *CityGaussian: Real-Time High-Quality Large-Scale Scene Rendering with Gaussians*, pp. 265–282. Springer Nature Switzerland, Oct. 2024.

[12] C. Peng, Y. Tang, Y. Zhou, N. Wang, X. Liu, D. Li, and R. Chellappa, *BAGS: Blur Agnostic Gaussian Splatting Through Multi-scale Kernel Modeling*, pp. 293–310. Springer Nature Switzerland, Oct. 2024.

[13] J. T. Barron, B. Mildenhall, D. Verbin, P. P. Srinivasan, and P. Hedman, "Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, June 2022.

[14] A. Knapitsch, J. Park, Q.-Y. Zhou, and V. Koltun, "Tanks and Temples: Benchmarking Large-Scale Scene Reconstruction," *ACM Transactions on Graphics*, vol. 36, pp. 1–13, July 2017.

[15] P. Hedman, J. Philip, T. Price, J.-M. Frahm, G. Drettakis, and G. Brostow, "Deep Blending for Free-Viewpoint Image-Based Rendering," *ACM Transactions on Graphics*, vol. 37, no. 6, pp. 1–15, 2018.

[16] H. Turki, D. Ramanan, and M. Satyanarayanan, "Mega-NeRF: Scalable Construction of Large-Scale NeRFs for Virtual Fly- Throughs," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12912–12921, IEEE, June 2022.

[17] L. Lin, Y. Liu, Y. Hu, X. Yan, K. Xie, and H. Huang, "Capturing, Reconstructing, and Simulating: The UrbanScene3D Dataset," in *Computer Vision – ECCV 2022*, pp. 93–109, Springer Nature Switzerland, 2022.

[18] A. Heyden and M. Pollefeys, "Multiple view geometry," in *Emerging Topics in Computer Vision* (G. Medioni, ed.), IMSC Press multimedia series, p. 47, Upper Saddle River, NJ: Prentice Hall, 2005.

[19] V. A. Kotel'nikov, "On the Transmission Capacity of the "Ether" and Wire in Electrocommunications," in *Modern Sampling Theory: Mathematics and Applications* (J. J. Benedetto and P. J. S. G. Ferreira, eds.), (Boston, MA), pp. 27–45, Birkhäuser Boston, 2001.

[20] H. Nyquist, "Certain Topics in Telegraph Transmission Theory," *Transactions of the American Institute of Electrical Engineers*, vol. 47, no. 2, pp. 617–644, 1928.

[21] C. Shannon, "Communication In The Presence Of Noise," *Proceedings of the IEEE*, vol. 86, no. 2, pp. 447–457, 1998.

[22] T. Lindeberg, "Scale-Space Theory: A Basic Tool for Analysing Structures at Different Scales," *Journal of Applied Statistics*, vol. 21, pp. 225–270, Jan. 1994.

[23] T. Lindeberg and B. M. ter Haar Romeny, *Linear Scale-Space I: Basic Theory*, pp. 1–38. Mathematical Imaging and Vision, Springer Netherlands, 1994.

[24] S. S. Arslan, L. Vogelsang, M. Fux, and P. Sinha, "Uniform Resampling vs. Image Blur: Aliasing Approximation via Isotropic Gaussian Filtering," 2025.

[25] T. Lindeberg, "Discrete Approximations of Gaussian Smoothing and Gaussian Derivatives," *Journal of Mathematical Imaging and Vision*, vol. 66, pp. 759–800, June 2024.

[26] P. Burt and E. Adelson, "The Laplacian Pyramid as a Compact Image Code," *IEEE Transactions on Communications*, vol. 31, no. 4, pp. 532–540, 1983.

[27] L. Williams, "Pyramidal parametrics," in *Proceedings of the 10th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '83, (New York, NY, USA), p. 111, Association for Computing Machinery, 1983.

[28] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, vol. 60, pp. 91–110, Nov. 2004.

[29] J. L. Schönberger and J.-M. Frahm, "Structure-from-Motion Revisited," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, June 2016.

[30] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, pp. 381–395, June 1981.

[31] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment a modern synthesis," in *Vision Algorithms: Theory and Practice*, pp. 298–372, Springer Berlin Heidelberg, 2000.

[32] M. Schutz, K. Krosl, and M. Wimmer, "Real-time continuous level of detail rendering of point clouds," in *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pp. 103–110, IEEE, Mar. 2019.

[33] S. Rusinkiewicz and M. Levoy, "QSplat: a Multiresolution Point Rendering System for Large Meshes," in *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, (USA), p. 343352, ACM Press/Addison-Wesley Publishing Co., 2000.

[34] C. Dachsbacher, C. Vogelgsang, and M. Stamminger, "Sequential Point Trees," *ACM Trans. Graph.*, vol. 22, p. 657662, July 2003.

[35] E. E. Catmull, *A subdivision algorithm for computer display of curved surfaces.* PhD thesis, The University of Utah, 1974. AAI7504786.

[36] H. Hoppe, "Progressive Meshes," in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, (New York, NY, USA), p. 99108, Association for Computing Machinery, 1996.

[37] J. Daniels, C. T. Silva, J. Shepherd, and E. Cohen, "Quadrilateral Mesh Simplification," *ACM Trans. Graph.*, vol. 27, Dec. 2008.

[38] D. Aliaga, J. Cohen, A. Wilson, E. Baker, H. Zhang, C. Erikson, K. Hoff, T. Hudson, W. Stuerzlinger, R. Bastos, M. Whitton, F. Brooks, and D. Manocha, "MMR: An Interactive Massive Model Rendering System Using Geometric and Image-Based Acceleration," in *Proceedings of the 1999 symposium on Interactive 3D graphics*, I3D99, pp. 199–206, ACM, Apr. 1999.

[39] J. H. Clark, "Hierarchical Geometric Models for Visible Surface Algorithms," *Communications of the ACM*, vol. 19, pp. 547–554, Oct. 1976.

[40] G. W. Brian Karis, Rune Stubbe, "A deep dive into nanite virtualized geometry," in *Advances in Real-Time Rendering in Games: Part I (proc. SIGGRAPH courses)*, 2021.

[41] J. Cohen, D. Aliaga, and W. Zhang, "Hybrid Simplification: Combining Multi-Resolution Polygon and Point Rendering," in *Proceedings Visualization, 2001. VIS '01.*, pp. 37–539, 2001.

[42] E. Gobbetti and F. Marton, "Far voxels: a multiresolution framework for interactive rendering of huge complex 3D models on commodity graphics platforms," *ACM Trans. Graph.*, vol. 24, p. 878885, July 2005.

[43] M. Wimmer, P. Wonka, and F. Sillion, "Point-Based Impostors for Real-Time Visualization," in *Proceedings of the 12th Eurographics Conference on Rendering*, EGWR'01, (Goslar, DEU), p. 163176, Eurographics Association, 2001.

[44] C. Crassin, F. Neyret, M. Sainz, S. Green, and E. Eisemann, "Interactive Indirect Illumination Using Voxel Cone Tracing," *Computer Graphics Forum (Proceedings of Pacific Graphics 2011)*, vol. 30, sep 2011.

[45] E. Gobbetti and F. Marton, "Layered point clouds: a simple and efficient multiresolution structure for distributing and rendering gigantic point-sampled models," *Computers & Graphics*, vol. 28, pp. 815–826, Dec. 2004.

[46] J. T. Kajiya, "The rendering equation," in *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, SIGGRAPH 86, pp. 143–150, ACM, Aug. 1986.

[47] W. R. McCluney, *Introduction to Radiometry and Photometry*. The @Artech House optoelectronics library, Boston: Artech House, 1994.

[48] M. Pharr, W. Jakob, and G. Humphreys, *Physically Based Rendering: From Theory to Implementation*. MIT Press, fourth ed., 2023. Online: `https://pbr-book.org/4ed/`, last accessed at 13.3.2026.

[49] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen, "The Lumigraph," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, SIGGRAPH96, pp. 43–54, ACM, Aug. 1996.

[50] S. Niklaus, L. Mai, and F. Liu, "Video Frame Interpolation via Adaptive Convolution," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2270–2279, IEEE, July 2017.

[51] S. Niklaus, L. Mai, J. Yang, and F. Liu, "3D Ken Burns Effect from a Single Image," *ACM Transactions on Graphics*, vol. 38, pp. 1–15, Nov. 2019.

[52] L. Yariv, J. Gu, Y. Kasten, and Y. Lipman, "Volume Rendering of Neural Implicit Surfaces," in *Proceedings of the 35th International Conference on Neural Information Processing Systems*, NIPS '21, (Red Hook, NY, USA), Curran Associates Inc., 2021.

[53] Z. Yu, S. Peng, M. Niemeyer, T. Sattler, and A. Geiger, "MonoSDF: exploring monocular geometric cues for neural implicit surface reconstruction," in *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, (Red Hook, NY, USA), Curran Associates Inc., 2022.

[54] A. Guédon and V. 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)*, pp. 5354–5363, IEEE, June 2024.

[55] B. Huang, Z. Yu, A. Chen, A. Geiger, and S. Gao, "2D Gaussian Splatting for Geometrically Accurate Radiance Fields," in *SIGGRAPH 2024 Conference Papers*, Association for Computing Machinery, 2024.

[56] Z. Yu, T. Sattler, and A. Geiger, "Gaussian Opacity Fields: Efficient Adaptive Surface Reconstruction in Unbounded Scenes," *ACM Trans. Graph.*, vol. 43, Nov. 2024.

[57] Z. Wang and A. C. Bovik, "Mean squared error: Love it or leave it? A new look at Signal Fidelity Measures," *IEEE Signal Processing Magazine*, vol. 26, no. 1, pp. 98–117, 2009.

[58] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[59] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv*, Sept. 2014.

[60] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The Unreasonable Effectiveness of Deep Features as a Perceptual Metric," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[61] Y. LeCun, Y. Bengio, and G. Hinton, "Deep Learning," *Nature (London)*, vol. 521, no. 7553, pp. 436–444, 2015.

[62] Q. Xu, Z. Xu, J. Philip, S. Bi, Z. Shu, K. Sunkavalli, and U. Neumann, "Point-NeRF: Point-based Neural Radiance Fields," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5428–5438, 2022.

[63] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 77–85, IEEE, July 2017.

[64] M. Zwicker, H. Pfister, J. Van Baar, and M. Gross, "EWA Splatting," *IEEE Transactions on Visualization and Computer Graphics*, vol. 8, pp. 223–238, July 2002.

[65] G. Kopanas, J. Philip, T. Leimkühler, and G. Drettakis, "Point-Based Neural Rendering with Per-View Optimization," *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering)*, vol. 40, June 2021.

[66] Q. Zhang, S.-H. Baek, S. Rusinkiewicz, and F. Heide, "Differentiable Point-Based Radiance Fields for Efficient View Synthesis," in *SIGGRAPH Asia 2022 Conference Papers*, SA '22, (New York, NY, USA), Association for Computing Machinery, 2022.

[67] V. Ye, R. Li, J. Kerr, M. Turkulainen, B. Yi, Z. Pan, O. Seiskari, J. Ye, J. Hu, M. Tancik, and A. Kanazawa, "GSplat: An Open-Source Library for Gaussian Splatting," *arXiv*, Sept. 2024.

[68] Z. Liang, Q. Zhang, W. Hu, L. Zhu, Y. Feng, and K. Jia, *Analytic-Splatting: Anti-Aliased 3D Gaussian Splatting via Analytic Integration*, pp. 281–297. No. 15075 in Lecture notes in computer science, Berlin, Heidelberg: Springer Nature Switzerland, Nov. 2024.

[69] S. S. Mallick, R. Goel, B. Kerbl, M. Steinberger, F. V. Carrasco, and F. D. L. Torre, "Taming 3DGS: High-Quality Radiance Fields with Limited Resources," in *SIGGRAPH Asia 2024 Conference Papers*, SA 24, pp. 1–11, ACM, Dec. 2024.

[70] G. Fang and B. Wang, *Mini-Splatting: Representing Scenes with a Constrained Number of Gaussians*, pp. 165–181. Springer Nature Switzerland, 2024.

[71] S. R. Bulò, L. Porzi, and P. Kontschieder, "Revising Densification in Gaussian Splatting," in *Computer Vision – ECCV 2024*, pp. 347–362, Springer Nature Switzerland, Nov. 2024.

[72] Y. Chen, J. Jiang, K. Jiang, X. Tang, Z. Li, X. Liu, and Y. Nie, "DashGaussian: Optimizing 3D Gaussian Splatting in 200 Seconds," in *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11146–11155, IEEE, June 2025.

[73] Y. Seo, Y. Choi, H. Son, and Y. Uh, "FLoD: Integrating Flexible Level of Detail into 3D Gaussian Splatting for Customizable Rendering," *ACM Transactions on Graphics*, vol. 44, pp. 1–16, 7 2025.

[74] U. Farooq, J.-Y. Guillemaut, A. Hilton, and M. Volino, "Optimized 3D Gaussian Splatting using Coarse-to-Fine Image Frequency Modulation," in *Proceedings of the 22nd ACM SIGGRAPH European Conference on Visual Media Production*, CVMP 25, pp. 1–10, ACM, Dec. 2025.

[75] J. Jung, J. Han, H. An, J. Kang, S. Park, and S. Kim, "Relaxing Accurate Initialization Constraint for 3D Gaussian Splatting," *arXiv*, Mar. 2024.

[76] J. Zhang, F. Zhan, M. Xu, S. Lu, and E. Xing, "FreGS: 3D Gaussian Splatting with Progressive Frequency Regularization," in *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21424–21433, IEEE, June 2024.

[77] K. Ren, L. Jiang, T. Lu, M. Yu, L. Xu, Z. Ni, and B. Dai, "Octree-GS: Towards Consistent Real-time Rendering with LOD-Structured 3D Gaussians," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PP, pp. 1–15, 2025.

[78] Y. Guan, Z. Wang, S. Zhang, J. Han, W. Wang, S. Wang, Y. Zhu, Y. Lv, W. Zhou, and J. She, "A Grid-Based Hierarchical Representation Method for Large-Scale Scenes Based on Three-Dimensional Gaussian Splatting," *Remote Sensing*, vol. 17, p. 1801, 5 2025.

[79] F. Windisch, L. Radl, T. Kohler, M. Steiner, D. Schmalstieg, and M. Steinberger, "A LoD of Gaussians: Unified Training and Rendering for Ultra-Large Scale Reconstruction with External Memory," *arXiv*, 2025.

[80] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: an imperative style, high-performance deep learning library," in *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, (Red Hook, NY, USA), Curran Associates Inc., 2019.

[81] S. Bonopera, J. Esnault, S. Prakash, S. Rodriguez, T. Thonat, M. Benadel, G. Chaurasia, J. Philip, and G. Drettakis, "SIBR: A System for Image-Based Rendering," 2020. Online: https://gitlab.inria.fr/sibr/sibr_core, last accessed 13.3.2026.