

SAMPLING AND SELECTION METHODS FOR APPLYING 2D NEURAL NETWORKS TO 3D GAUSSIAN SPLATS

By

Raphael DuSablón

May, 2025

Director of Thesis: David Hart, PhD

Major Department: Data Science

ABSTRACT

We propose a novel approach for applying interpolation methods to unstructured volumetric data that allows for the operation of 2D neural networks directly on 3D Gaussian splats. Gaussian splatting is at the cutting edge of volume rendering methods, 2D neural networks have achieved a dominant and lasting degree of success and real-life application. We propose leveraging the advantages of both, an approach which is the first of its kind. We extend the method for interpolated convolution on 3D surface meshes with 2D CNNs by Hart *et al* to the unstructured 3D volumetric data of Gaussian splats and present an end-to-end pipeline for our work. We showcase our results with style transfers on 3D Gaussian splats performed by a 2D convolution model with no retraining. Our results compare favorably with those of current approaches to performing style transfers on 3D Gaussians using purpose-built and purpose-trained 3D models.

SAMPLING AND SELECTION METHODS FOR APPLYING 2D NEURAL
NETWORKS TO 3D GAUSSIAN SPLATS

A Thesis

Presented to The Faculty of the Department of Data Science
East Carolina University

In Partial Fulfillment of the Requirements for the Degree
Master of Science in Computer Science

By

Raphael DuSablón

May, 2025

Director of Thesis: David Hart, PhD

Thesis Committee Members:

Nic Herdon, PhD

Moritz Dannhauer, PhD

©Raphael DuSablon, 2025

Table of Contents

LIST OF TABLES	v
LIST OF FIGURES	vi
CHAPTER 1: INTRODUCTION	1
CHAPTER 2: RELATED WORK	5
2.1 3D Rendering and Novel View Synthesis	5
2.2 Style Transfers	8
2.3 Concurrent Work	9
CHAPTER 3: METHODOLOGY	11
3.1 Preliminaries	11
3.2 Neighborhood-Approximated-Normal Based Filtering	12
3.3 Gaussian Sampling	15
3.4 Interpolated SelectionConv	18
3.5 Gaussian Splats to Interpolated SelectionConv	22
CHAPTER 4: RESULTS AND DISCUSSION	26
4.1 Ablation Study: Filtering	26
4.2 Ablation Study: Sampling	31
4.3 Ablation Study: Normals	35
4.4 Style Transfer Results	37

4.5 Comparison with StyleGaussian	40
CHAPTER 5: CONCLUSION AND FUTURE WORK	43
BIBLIOGRAPHY	44

LIST OF TABLES

4.1	Results of Neighborhood-Approximated-Normal-Based Filtering on a program- matically generated Gaussian splat	28
-----	---	----

LIST OF FIGURES

1.1	Style Transfer Results, Flower Bouquet	3
1.2	Style Transfer Results, Father’s Day Scene, with Detail	4
3.1	Point Cloud vs Gaussian Splatting	12
3.2	Noisy Gaussian Splat with Highlighted Surface and Noise	13
3.3	Neighborhood-Approximated Normal Vector	14
3.4	Point Cloud Sphere with Normal Vector Magnitudes	15
3.5	Five Gaussian splats, Center Points, and Gaussian Sampling	19
3.6	Selection Conv	21
3.7	Interpolated Selectionconv	22
3.8	Convolution on Gaussian splats	23
3.9	End-to-End Pipeline	25
4.1	3DGS Sphere, with Noise, Before and After Filtering	27
4.2	3DGS and Point Cloud showing Filtering	29
4.3	Gaussian Splat Before and After Filtering, with Detail	30
4.4	Gaussian Sampling on Table Representation	32
4.5	Effects of Gaussian Sampling on Style Transfer	34
4.6	Effects of Ball-Pivoting vs Random Normals on Style Transfer	36
4.7	Style Transfer Results on Gaussian splats of differing quality	38
4.8	Two Style Transfers on Bicycle Representation	39
4.9	Our Results and Style Gaussian, Bike and Bench Scene	40

4.10 Our Results and Style Gaussian, Truck Scene	41
4.11 Our Results and Style Gaussian, Truck Scene, High Res	42

Chapter 1

Introduction

Gaussian splatting, a volume rendering method that enables real time displays of radiance fields and even dynamic 4D scenes, represents the cutting-edge of current 3D rendering techniques. This method has proven highly effective for higher dimensional media, especially for rendering complex scenes and difficult lighting effects. Given this success and the increasing adoption of Gaussian splatting, there is growing interest in performing machine learning on Gaussian splats. A perennial challenge of performing machine learning on 3D and higher dimensional media, not just Gaussian splats, is the toilsome training process, due in part to the inherent difficulty of finding optimal solutions in unstructured higher dimensional data—the infamous curse of dimensionality. Additionally, there is a paucity of good datasets for training on.

In both cases, the opposite case holds for 2D images, in particular conventional pixel-based images. Most machine learning techniques and algorithms work most effectively on structured 2D data, the computational requirements for performing that work are low, and 2D machine learning models are a more mature and established technology than models operating on other spaces. The sheer volume of excellent 2D datasets and pre-trained off-the-shelf models are an incredibly rich resource. That resource is the result of decades of research on 2D models and enormous quantities of compute being devoted to the training of those models. As higher-dimensional volume rendering techniques such as Gaussian splats proliferate, tapping into that resource for performing machine learning tasks on higher dimensional volumes could shave years off of research and developmental timelines and would

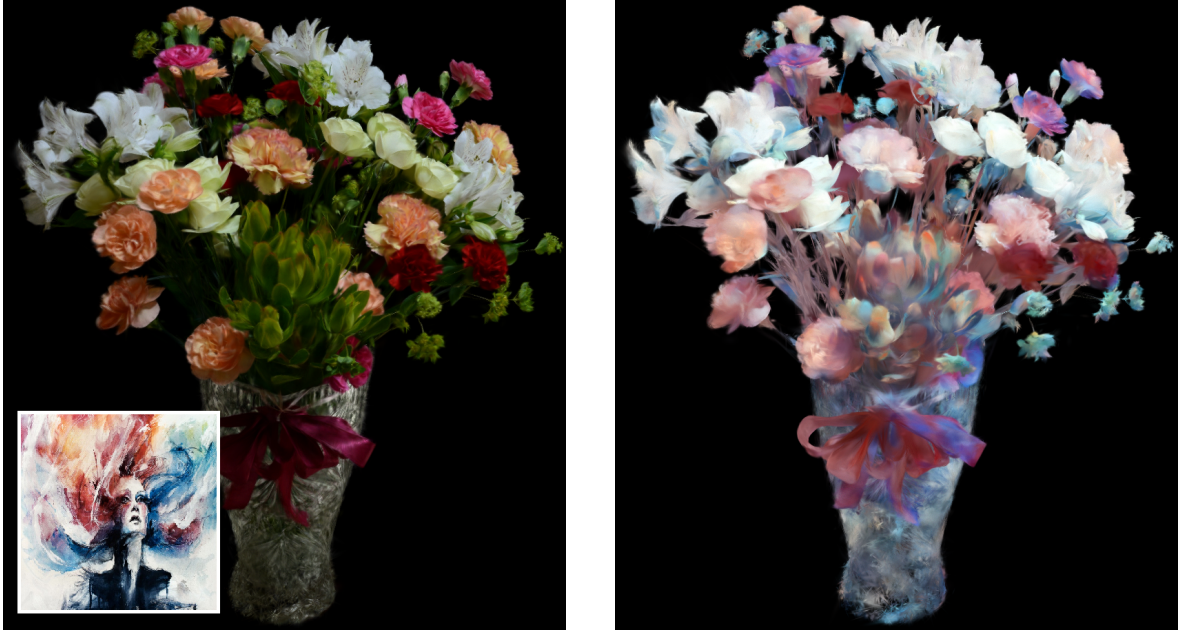
significantly reduce the amount of compute required for that research. Hence, our paper aims to tap into this rich resource by applying established, vetted, well-understood 2D CNNs to the unstructured volumetric data of 3D Gaussian splats.

Our project builds on the works of Hart *et al*, [12] and [13]. In those works, a method for performing machine learning on structured mesh-based surfaces using 2D image models was introduced. Here, we develop a pipeline that allows for the same convolution operators used by Hart *et al* on structured mesh-based surfaces to perform machine learning on volumetric 3D Gaussian splat data. Our work demonstrates that this pipeline is highly effective, specifically for our example test case of style transfer.

Our pipeline involves an I/O layer for both of the common file types used for storing 3D Gaussian splats (.ply and .splat), a filtering step to improve the image quality of the final result, a sampling step to improve content preservation and model performance, and a layer integrating the method of Hart *et al* into our pipeline. A final step transforms model outputs back to 3D Gaussian splats. Our pipeline is therefore an end-to-end solution whose input and output are both 3D Gaussian splats and which is capable of performing style transfer using 2D image models.

A distinguishing feature of our work is that no model training or retraining is performed or even required. To the best of our knowledge, all other techniques for performing machine learning on 3D Gaussian splats require the training of new boutique models from scratch—a requirement both expensive given the computational requirements of training in 3D and problematic given the rarity of large datasets of labeled 3D Gaussian splats. We instead leverage already existing and trained off-the-shelf 2D image models, many of which, such as VGG, have demonstrated excellent performance and seen widespread applicability and adoption.

This pipeline and our methods produce satisfactory results. Figure 1.1 and figure 1.2 demonstrate both the capabilities of 3D Gaussian splatting and the effectiveness of our method. We performed ablation studies on the efficacy of each step in our pipeline, and



View of a 3D Gaussian and a Style Image The 3D Gaussian splat after Style Transfer

Figure 1.1: A 3D Gaussian splat with geometrically complex content and an image containing a distinct artistic style (Left) and a 3D Gaussian splat of the same content after style transfer (Right).

we benchmarked our results against *StyleGaussian*, a method contemporary to ours which performs style transfer on 3D Gaussians using the more traditional approach of training a dedicated 3D model [37]. We find our results to be broadly similar to those produced by StyleGaussian. At times, superior. In particular, color alignment between the style image and our style transferred 3D Gaussian splats is as good or better than the benchmark.

To conclude our introduction, our contributions are the following:

- Presenting a first-in-kind graph-based implicit surface approach for style transfer on Gaussian splats
- Ablation studies on the proposed approach and its implementation
- Our results and comparisons with a contrasting approach

The remainder of this thesis outlines related and concurrent works, our methodology, our results, ablation studies, and our benchmark. Finally, we present our conclusions.

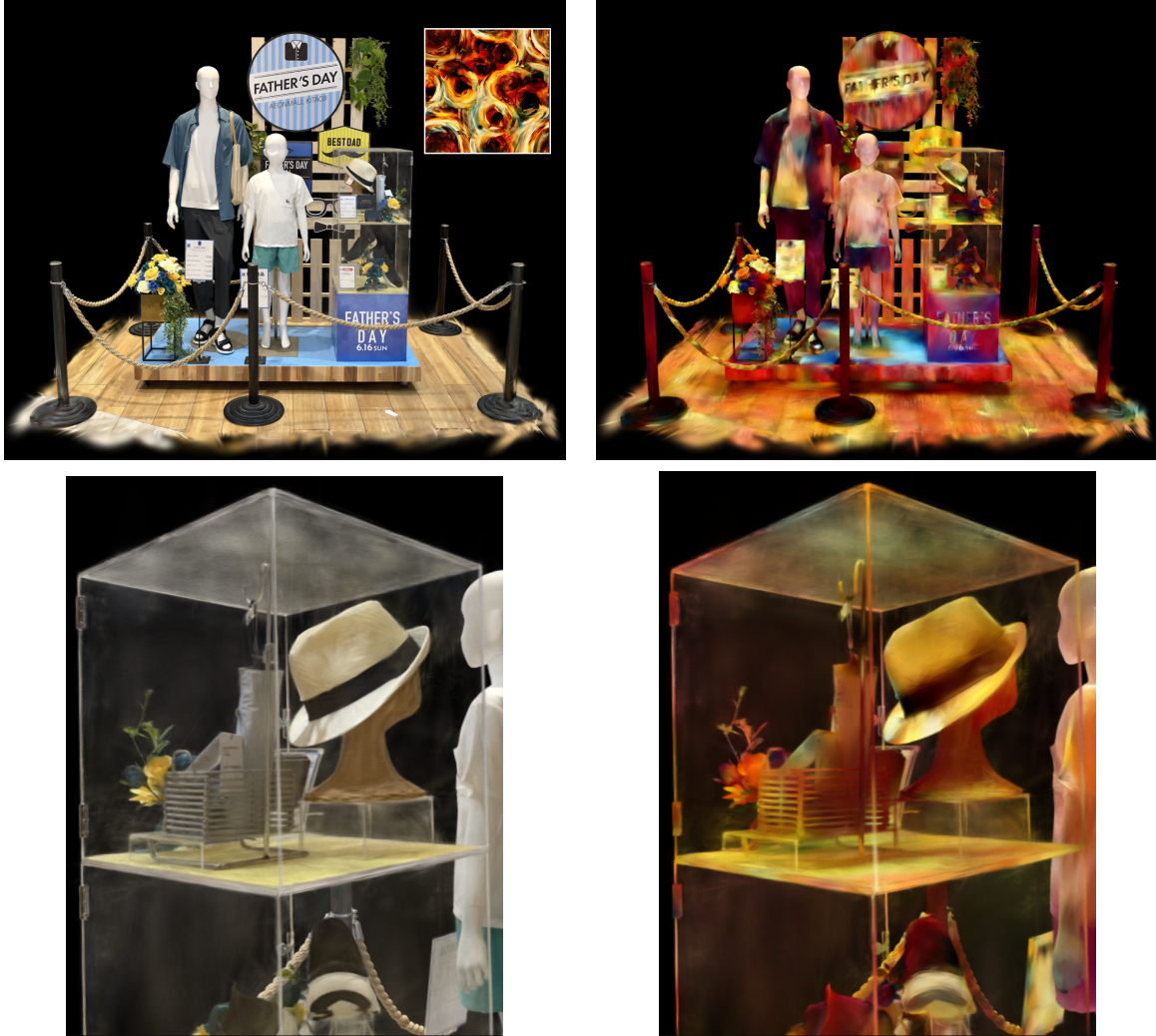


Figure 1.2: A 3D Gaussian splat representation of a complex and detailed scene (Top Left) and a 3D Gaussian splat of the same scene after style transfer (Top Right). Detail from the scene shows a glass case viewed from behind (Bottom).

Chapter 2

Related Work

In this section, we cover the general overviews of the foundations of relevant subjects, followed by discussions of more specialized areas involved in this paper.

2.1 3D Rendering and Novel View Synthesis

A cursory or even casual look at most publicly available and accessible 3D digital media makes it apparent that classical 3D rendering relies primarily on mesh-based methods. Academically and in industry, meshes have continued to dominate 3D rendering in recent decades and retain solid importance [39] with triangular mesh surfaces making up by far the most common approach. Such approaches in some respects remain the state-of-the-art and an active area of research [36, 38, 23]. But meshes are not without their drawbacks, such as poor handling of lighting and light phenomena, issues with scaling and performance, etc. A traditional competitor to mesh is the three-dimensional pixel, or voxel [7, 42], and there is a history of voxels proving to be superior on the margins to mesh techniques [25, 14, 56, 32]. A few typical drawbacks of voxels are that they suffer at higher resolutions due to requiring a uniform density and due to their poor memory performance [23, 32, 2]. Something of an outlier were point clouds, which though they were a less common tool for rendering, saw considerable use in visio-spatial applications such as robotics and mapping [44, 54]. However, more recently point clouds have become more prevalent as a rendering method [11].

In the various approaches to 3D rendering, the development of effective neural networks

rapidly led to new iterations of and improvements on classical methods [28, 50, 55]. In particular, the field saw a revolution with the introduction of NeRFs, Neural Radiance Fields, introduced by [40]. This work presented the first algorithm for representing a 3D scene using a deep neural network. The model takes in three spatial and two camera view variables and outputted volume density and radiance values which could then be used to render a view of a 3D scene using classical rendering methods carefully constrained to be differentiable. Additionally, there is a positional encoding that enables the representation of higher frequency functions from the 5D inputs. As a result, NeRF is able to represent geometrically complex scenes involving difficult optical phenomena as five-dimensional radiance fields, the differentiability of the rendering process allows for optimization to be performed and for a loss function to be defined on the final 2D view of the representation—a necessary condition for training on reference images. High frequency scene content is represented without undue costs in neural network performance. A particular feature of this algorithm is that it allows for rendering novel views of 3D scenes, a process termed novel view synthesis. More technically, novel view synthesis is the generation of unseen views of some object or scene from a set of 2D images, as described by [49], in effect generating a view or image as if the camera had been moved in 3D space to some vantage it had never been at. Typically, this is a difficult problem, as one has to not only faithfully reconstruct those sections captured in 2D images, but also unseen areas [38]. Decades-old novel view synthesis techniques do exist [6], but as with other areas, the field saw a marked improvement with the development of modern machine learning techniques [15, 52, 53, 55].

Those improvements employ neural networks, either a Multi-Layer Perceptron (MLP) as in the case of the original [40] or some other network. These networks typically share the same overall architecture and performance envelope: inputs are some representation of points in space and viewing angles and outputs are the views of a scene from that point and angle. Training these models is an effective albeit slow and computational costly process, as training on 3D scenes suffers from poor generalizability and has problems with details at distance

[28], but the resulting trained model is currently the best-in-class tool for performing novel view synthesis, far exceeding the capacities of classical novel-view synthesis techniques. Once trained, the model is employed at render-time to produce novel views. Certain drawbacks remain, such as the training requirements and the inability of most implementations to perform rendering at acceptable speeds. A variety of efforts have been directed at mitigating these limitations [57, 43, 41, 46], among the most fruitful has been to dispense with rendering with a neural network entirely.

3D Gaussian Splatting for Real-Time Radiance Field Rendering builds on the success of radiance fields while correcting for some of the drawbacks described above by eliminating the use of a neural network at render time. In doing so, Gaussian splatting, first proposed by [24], enabled real-time rendering of novel-view synthesis at high resolutions and qualities. This work introduced *3D Gaussian splats*, a point-cloud-like primitive that is differentiable and unstructured and which can easily be projected to 2D for rendering, an optimization algorithm for fine-tuning the properties of Gaussian splats, and a novel rasterizer that is likewise differentiable and allows for learning and for back-propagation while also being fast to render high quality scenes. Alpha-blending of the Gaussians as they are projected to 2D ensures good textural detail and smooth geometries, at least in regions of high splat density [17]. While a neural network is still employed, only the initial creation of the 3D Gaussian splat depends on a model. Like most novel-view synthesis tools, 3D Gaussian splats are constructed from a set of 2D photos sampling the object to be represented, and a neural network is trained on these 2D images to create the 3D scene. The advantages 3D Gaussian splatting holds over previous methods are 1. the novel primitives, rasterizer, and the optimization algorithm greatly improve training time, and 2. no model I/O is required at render-time, making 3D Gaussian splatting the first technology to achieve real-time rendering of high-quality outputs with novel view synthesis.

As noted above, the primitive used in 3D Gaussian splatting is in many ways similar to those in point clouds. An inherent benefit of point clouds are their high scalability, in part

due to simple encoding and in part due to their non-uniform sampling in representing real objects, both being features preserved in Gaussian splatting. As noted by [23], naive point clouds suffer in having a middling capacity for rendering lighting and textural phenomena, a weakness not shared by Gaussian splats [5]. By utilizing a point cloud not of discrete points but of Gaussian functions, the 3D Gaussian splatting solves a number of shortcomings relating to detail, lighting, and textural fidelity while generally keeping all the advantages, such as smaller file sizes, of point clouds [1], making for a powerful tool for 3D rendering with novel view synthesis.

2.2 Style Transfers

While examples go back to the 90s [33], if not earlier [8], convolutional neural networks gained a high profile in the world of computer science in 2012 with the success of AlexNet, which was a major improvement on previous attempts at computer vision and was the first winner of the ImageNet challenge [30, 31]. As convolutional neural networks came into their own in the mid 2010s, a variety of different models tackling various problems in computer vision, including object detection, debuted [47, 45]. With the ability to represent objects or semantic content of images came the ability to represent the content of images other than objects. This is the insight which lead to the first work on performing Style Transfers with neural networks [10]. This work pioneered using CNNs to separate out the content of an image and combine it with the style of another image, typically a painting or other famous work of art. In detail, a textural informational feature space is used to find a transferable representation of an image style, and new images are synthesized by combining the content of the target image with the style of the source image using a loss function that is a linear combination of the losses from content and style. A model architecture for image recognition called VGG, introduced by [51] and which involves using small convolutional filters but very deep networks, has become the one most commonly relied on for style transfer [9]. Among the most advanced implementations based on this architecture is Linear Style Transfer [35],

which we utilize in our work.

With this foundation, a number of advances were made, including those which improved performance, those offering some degree of flexibility [34, 19, 22], as well as extensions to other domains such as by [59] which presented the first method for applying style transfer techniques to 3D scenes using radiance fields. Additionally, a novel loss function was introduced based on nearest-neighbor algorithms, with renderings in 2D of the 3D scene being used for evaluating loss. Given the variety of different 3D rendering techniques, separate attempts have been made to replicate the success of 2D style transfer in the 3D realm, such as with radiance fields [59] and mesh surfaces [29, 13]. Such is the case also with Gaussian splatting.

2.3 Concurrent Work

Naturally, given the success of 3D Gaussian splatting, there has been considerable attention paid to attempting to translate the success of 2D Neural Style Transfers to the 3D realm with Gaussian splatting, one such attempt being the current work. Some concurrent works have built on [18] which augments VGG with a special layer called AdaIN that aids in aligning the features of the style image and the content image, including the work of [49], which was among the very first attempts to perform style transfers on 3D Gaussian splats and which processes the Gaussian splats at a variety of resolutions with their model and uses that to obtain stylized 3D Gaussian splats. Similarly, [37] introduced using a KNN 3D CNN model to reduce operations that break multi-view consistency and it employs a unique approach of encoding 2D VGG features to 3D Gaussians, then features are transformed using AdaIN and then decoded to produce novel RGB values for the 3D scene.

Combining style transfer with a few other computer vision tasks such as editing, [20] uses image segmentation of the edit or style image to separate out distinct regions. Segmented regions are matched using DINO features [4] to propagate across views. Likewise, by leveraging segmentation models [21] and our benchmark paper [37] allow for style transfers to

select portions of the 3D Gaussian by learning with a 3D segmentation model the objects represented by the Gaussian splat. They also use KNN feature matching to improve results, with [21] taking an additional alignment step to ensure better visual consistency. VGG features are also used by [58], which constructs the loss function in a way that evolves the style spatially along the objects represented in the 3D Gaussian. [28] likewise uses a refinement of the loss function to improve results. A different niche is filled by [9], which applied a new method of addressing scale to tackle style transfers to ultra-high resolution 3D Gaussians. In short, a plethora of efforts are underway.

All of the described works, including our benchmark, share a commonality: they involve one or multiple training steps or passes on custom models to be able to perform style transfers on 3D Gaussians. The present work, uniquely, takes inspiration from what made Gaussian splatting attractive and revolutionary in the first place, you skip the costly neural network step. In the case of Gaussian splatting, model I/O at render time is skipped; in our case, training and retraining of custom models is skipped.

Chapter 3

Methodology

3.1 Preliminaries

3D Gaussian splats are an unstructured and explicit representation of 3D objects comprised of a large number of ellipsoidal 3D Gaussian functions, commonly referred to as “splats” owing to their appearance, arranged in 3-space [24]. The collection of all these individual splats or primitives together comprise the whole of representation, the 3D Gaussian splat, often abbreviated as a 3DGS. It is helpful to note that a “splat” may refer to a single primitive i.e. an individual Gaussian function, or to the entire collection of Gaussian functions that make up the whole representation of the scene or object being captured. Here, the overall scene or representation is referred to as a 3D Gaussian splat, 3DGS, or Gaussian splat, while the component Gaussian functions are referred to as “individual Gaussian splats” unless context is sufficiently clear.

The standard encoding for 3DGS files consists of a set of vectors very similar to the typical encoding of a point cloud but with several additional parameters. The files contain vectors encoding a set of 3D coordinates for position, scaling values along each dimension, color values and opacity (typically in RGBA format), quaternions which are used for determining orientation, and optionally a vector describing normal vectors and vectors for encoding spherical harmonics. Figure 3.1 illustrates the graphical difference in this construction versus a standard point cloud.

From a set-theoretic perspective, we can describe a 3D Gaussian splat as a set of positions

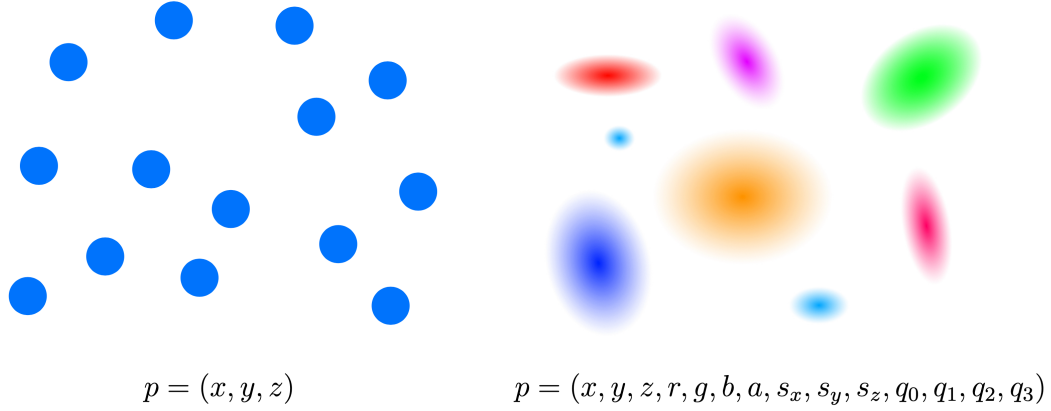
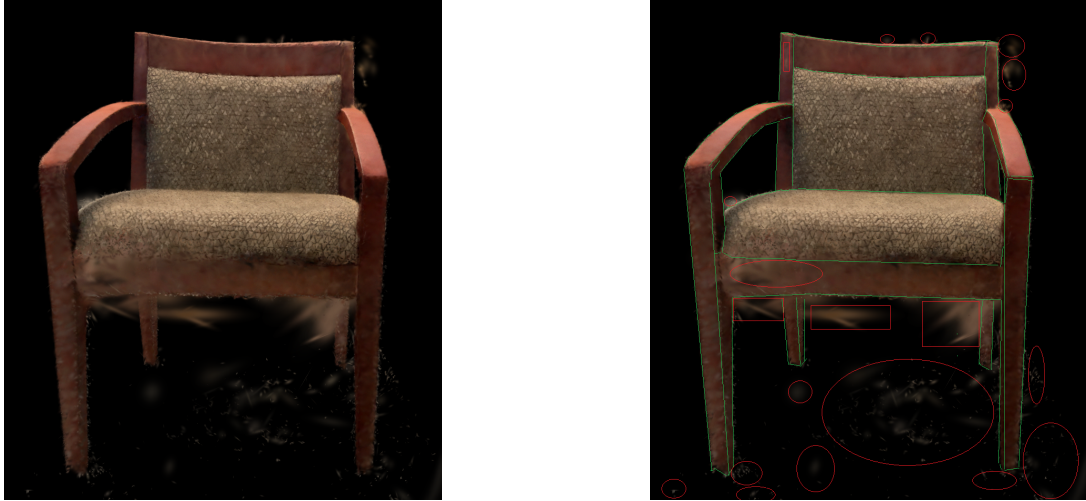


Figure 3.1: A standard point cloud representation (Left) with individual points described by spatial coordinates, and a Gaussian splat representation (Right) described by a much richer set of parameters.

$P = \{p \in \mathbb{R}^3\}$, a set of scaling values $S = \{s_0, s_1, s_2 \in \mathbb{R}\}$, a set of normalized values for color and opacity $C = \{r, g, b, opacity \in \mathbb{R}\}$, and a set of quaternions $Q = \{q \in \mathbb{H}\}$ i.e. $Q = \{q = q_0 + q_1 + q_2 + q_3 = a + b\mathbf{i} + c\mathbf{j} + d\mathbf{k} \text{ where } 0 \leq a, b, c, d \leq 1\}$ where each quaternion is equivalent to a matrix of rotation R of the form $M_{3,3}(\mathbb{R})$.

3.2 Neighborhood-Approximated-Normal Based Filtering

In a practical sense, individual Gaussian splats tend to lie along implicit geometric surfaces, or at the very least tend to cluster rather than occupying isolated points in space. In most situations, any Gaussian splats that are highly isolated are likely to be “noisy” data points or artifacts left over from the training step when the representation was first generated from image data. To reduce this noise, a filtering mechanism is implemented. Most user-generated Gaussian splats, and even many publicly available splats, have some degree of noisy data points. In addition to being visually unappealing, these noisy data potentially distort or even obscure content and so reduces the accuracy of style transfers and other content-related machine learning tasks. Correctly identifying these noisy points would typically require identifying the geometry of the rendered object and then removing any splats falling



A 3D Gaussian Splat with a high level of noise Highlighted Surface Edges and Noisy Data

Figure 3.2: A 3D Gaussian splat with a high level of noise (Top) and the same with the edges of the implicit surface (Green) and sections of noise highlighted (Red). Noisy regions demonstrate a strong tenancy towards falling outside the surface.

outside of that geometry, else leveraging machine learning to try and segment objects within the representation and filter out individual Gaussian splats not lying on or in those objects. While not beyond the capacity of modern neural networks [16, 48] such approaches can be time consuming. Here, a lightweight heuristic for segregating noisy data points is introduced and used instead.

Armed with the assumption that most geometric objects a user might be interested in will tend to be contiguous and will tend to be represented by splats that roughly follow along an implicit surface, we can then conclude that the more isolated a point is, and the more distant it is from any implicit surface, the more likely it is to be a noisy data point and a candidate for filtering out. This assumption is borne out in figure 3.2. Handily, both isolation from neighboring points and distance from an implicit surface can be calculated at the same time in our method. That method is as follows: the Gaussian splat is converted to a sparse point cloud comprised of the central points of each individual Gaussian splat, equivalent to the set P defined above. As an aside for later use, we will define as a *one-to-one point cloud* any point cloud which is constructed from exactly those points which are in set P , a name

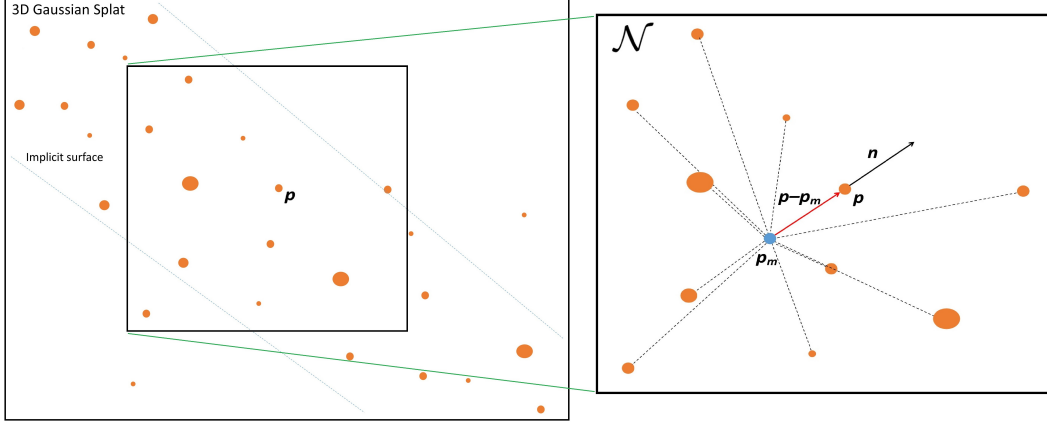


Figure 3.3: Within the point cloud derived from a 3D Gaussian splat, points in a neighborhood about some selected point p are used to determine the mean point p_m from which the Neighborhood-Approximated normal vector n is derived. Insofar as p is on or within the implicit surface, n will be normal to that surface. Additionally, the magnitude of n increases when p is at a greater distance from other points in its neighborhood, i.e when p does not lie in or on the implicit surface.

chosen since the mapping between the point cloud and the collection of individual Gaussian splats is a bijective function. For each $p \in P$, the KNN algorithm is used to determine the neighborhood $\mathcal{N}$ about that point. In that neighborhood, the midpoint p_m is calculated as the mean of all points in the neighborhood. Then, a vector n' which traces the line from the mean point p_m to the selected point p is determined and then translated from p_m to p , giving the vector n which traces the line from p to $p + (p - p_m)$. More formally, that vector is given by:

$$n = p + (p - p_m) \text{ where } p_m = \frac{1}{K} \sum_{j \in \mathcal{N}} p_j$$

This resulting vector, originating at the selected point and pointing in the opposite direction away from the mean point, is an approximate normal vector, here called the Neighborhood-Approximated normal vector. Figure 3.3 provides a visual demonstration of this mathematical operation.

What is more, the vector's magnitude $|n|$ is exactly equal to the distance of the selected point p from the neighborhood mean p_m . As such, the $|n|$ of isolated points is much greater

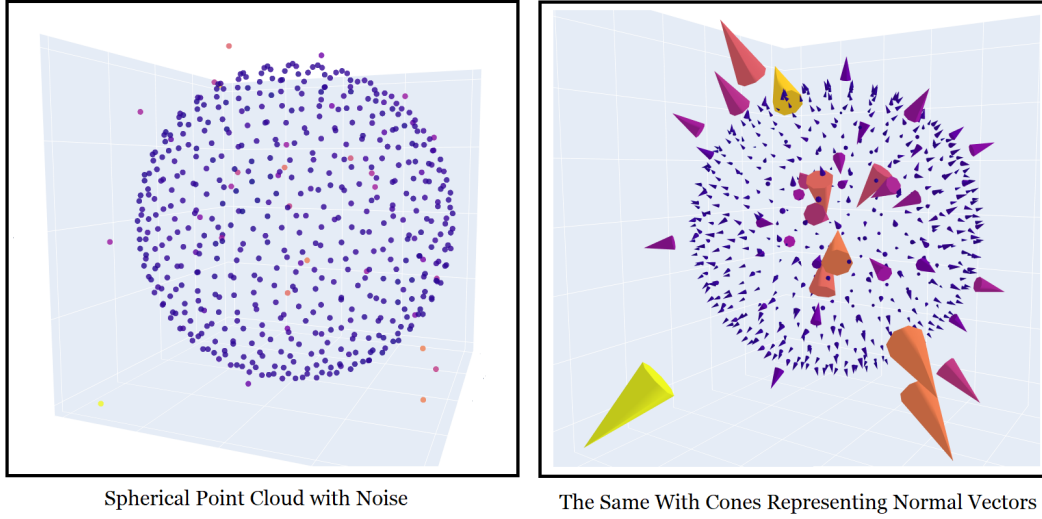


Figure 3.4: A spherical point cloud with a large number of points added at random distances from the surface to simulate noise (Left) and the same point cloud with cones representing the Neighborhood-Approximated normal vectors (Right) demonstrating how the magnitude of the Neighborhood-Approximated normal vector strongly segregates points close to the surface from those far from the surface.

than the $|n|$ of points in or on the implicit surface or for points that are in tightly clustered groups. So, the magnitude of the Neighborhood-Approximated normal vector serves as an easily-calculated heuristic for identifying which points are highly isolated or which are far from the implicit surface of the geometric object being represented. Figure 3.4 demonstrates this effect. Having been identified, those points may be filtered, reducing noisiness in the 3D Gaussian splat. This filtering process we term *Neighborhood-Approximated-Normal-Based Filtering*.

3.3 Gaussian Sampling

As previously indicated, of the classical rendering methods, Gaussian splats are conceptually most similar to point clouds, and programmatically they share a number of features. A 3D point cloud file with colors and a 3D Gaussian splat file will both be made up of a vector of spatial coordinates and a vector for colors using RGB or similar schema. For the Gaussian splat, the coordinates designate the center of the splat, and the colors vector also

includes a column for opacity values. In addition to those vectors, Gaussian splat files have vectors describing scale and orientation. In keeping with the conceptual parallels, this means Gaussian splats are like point clouds where each point p is the center of an oriented ellipsoidal 3D Gaussian function, the configuration of this ellipsoid being determined by scaling values s_0, s_1, s_2 and the rotation matrix R [24].

For optimization reasons, the rotation matrix is stored as a quaternion, being converted back to a matrix during rendering. That conversion is as follows:

$$R = \begin{bmatrix} 1 - 2q_2^2 - 2q_3^2 & 2q_1q_2 - 2q_0q_3 & 2q_1q_3 + 2q_0q_2 \\ 2q_1q_2 + 2q_0q_3 & 1 - q_1^2 - q_3^2 & 2q_2q_3 - 2q_0q_1 \\ 2q_1q_3 - 2q_0q_2 & 2q_2q_3 + 2q_0q_1 & 1 - q_1^2 - q_2^2 \end{bmatrix}$$

Or as the equivalent

$$R = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2q_1q_2 - 2q_0q_3 & 2q_1q_3 + 2q_0q_2 \\ 2q_1q_2 + 2q_0q_3 & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2q_2q_3 - 2q_0q_1 \\ 2q_1q_3 - 2q_0q_2 & 2q_2q_3 + 2q_0q_1 & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix}$$

For our purposes, the Gaussian splat must first be converted to a point cloud before machine learning is performed. Naïvely selecting just the center of each Gaussian splat, i.e. constructing a One-To-One Point Cloud from the positions vector P and the colors vector C , yields a point cloud that is far too sparse for useful machine learning tasks to be performed on it. We provide a technique for preserving more of the information encoded in the Gaussian splat file by generating sets of points from sampling the oriented ellipsoidal Gaussian functions formed by the individual Gaussian splats.

Owing to the considerable variation in size between the individual Gaussian splats (in practice, generally between two and three orders of magnitude), uniformly sampling the Gaussian splats is both inefficient and leads to point cloud densities that do not correspond well to the geometric objects represented by the Gaussian splats. Similar issues present

themselves for Gaussian splats with low and high opacity values, as less opaque Gaussian splats generally do not play as important a role in the representation and are less impactful at the rendering stage. To address this, splats with high opacity values and splats that have higher scaling values are sampled from at a higher rate. In other words, more data is collected from the larger, more visible splats.

To this end, a column vector of scalar values, also with a one-to-one correspondence to the set of individual Gaussian splats, designates how strongly to weight each individual Gaussian splat is calculated. That vector, here called an “importance” vector, is the set of weights $W = \{w | 0 \leq w \leq 2\}$ given via

$$w = \frac{|s_0| + |s_1| + |s_2|}{\kappa_s} + \frac{opacity}{\kappa_o}$$

where κ are normalization constants, with κ_s being the largest scaling value in the set S and κ_o being the largest opacity value in set C . Normalization of the terms is preferred as opacity is theoretically bound by 0 and 1 but, in practice, the greatest opacity value may be lower than 1. Moreover the scaling values are not bound and vary considerably from file to file, at times being substantially smaller than the opacity values and at times being substantially larger than the opacity values. Hence normalization of both terms to the range $[0, 1]$ is used.

With the importance vector, we can calculate how many times to sample from each Gaussian splat. Given a desired total number of extra points to sample $T_{samples}$, the vector $K = \{k \in \mathbb{Z} | k \geq 0\}$ containing the number of times each Gaussian should be sampled is given by

$$k = w \frac{T_{samples}}{\sum_{j \in W} w_j}$$

As an implementation detail, $T_{samples}$ is passed by the user as a parameter, the size of K being up to the discretion of the user given preferences for details and performance. Also, each element in K is rounded down as k must be greater than or equal to 0 and an integer.

To produce a novel point p_{GS} using *Gaussian Sampling* for some Gaussian splat centered on point p , several steps are required. Along a given axis in 3D-space, a Gaussian function centered about the origin is given by

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}}$$

With σ in this case being the element $s \in S$ corresponding to the given axis. Using random values of x , the coordinate values of p_{GS} for each axis are calculated by the equation above. p_{GS} is then rotated about the origin with the rotation matrix R . Finally, p_{GS} is translated from its position relative to the origin to the same position relative to point p . This process is repeated k times for each individual Gaussian splat, and the color and opacity values of that individual Gaussian splat are assigned to the novel points. All novel points and the initial One-to-One Point Cloud are concatenated together to give the final point cloud formed by Gaussian Sampling. See Figure 3.5 for a visual explanation and justification. The resulting point cloud provides a sufficiently dense and information-rich representation to support style transfer using an extension of *Interpolated Selectionconv*.

3.4 Interpolated SelectionConv

Interpolated Selectionconv is a framework for convolutional neural network operations on meshes. It allows for neural networks originally configured for and trained on 2D images to operate on complex 3D meshes by representing the mesh as a graph network and using an interpolation technique and a special operator to perform convolution on the mesh in a manner mathematical equivalent to convolution on 2D pixels [13].

Interpolated Selectionconv is an extension of *SelectionConv*, which introduced the graph convolution operator that transfers 2D convolution weights from pre-trained and well-established 2D convolutional models (such as the ubiquitous and highly performant VGG models) to graph networks [12]. Convolution on graph networks is not equivalent to convolution on

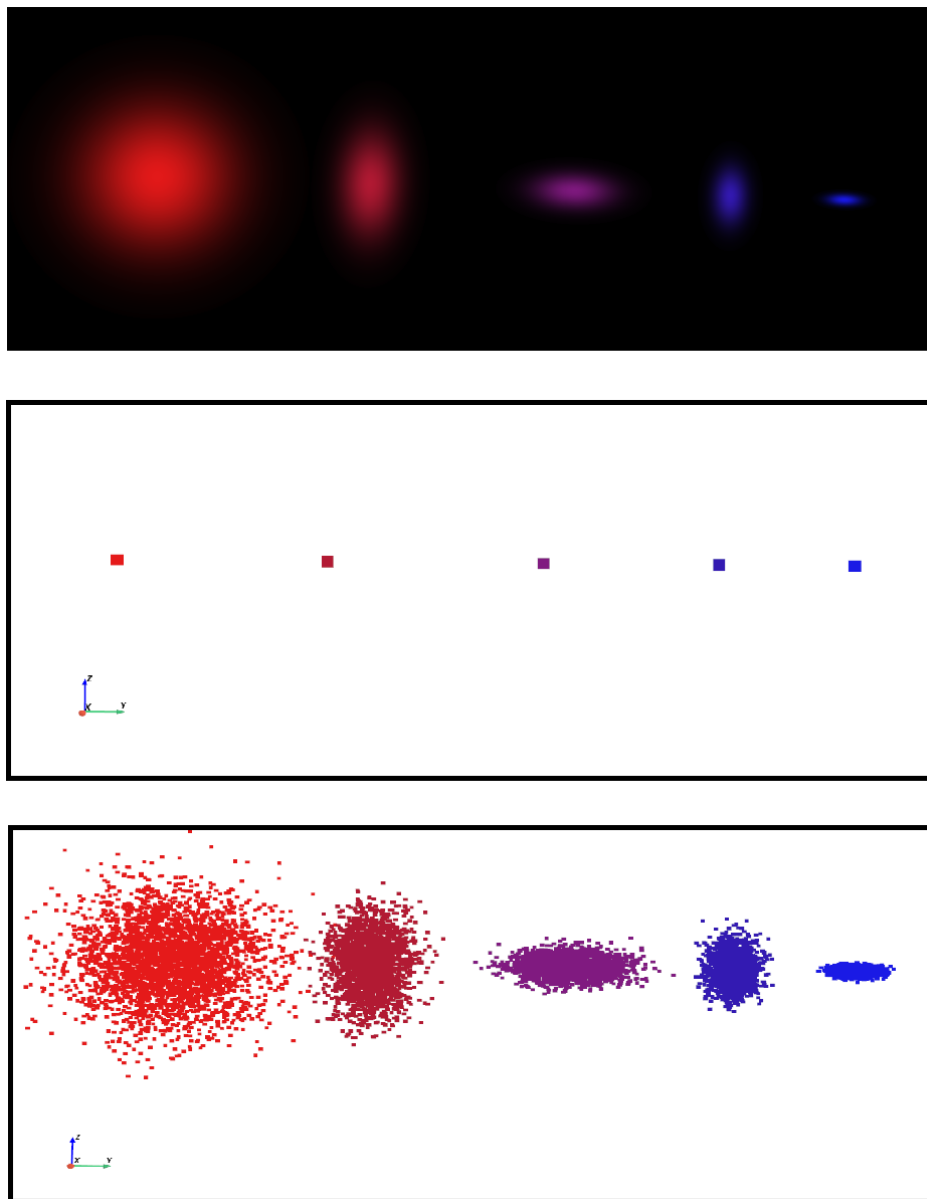


Figure 3.5: A very simple 3D Gaussian splat scene made from five individual Gaussian splats of varying colors and scalings (Top), the point cloud produced by selecting only the points provided in the 3D Gaussian splat file encoding, i.e. the centers of each individual Gaussian (Middle) and a point cloud produced by performing extensive Gaussian Sampling to collect more points within each Gaussian. This graphically demonstrates how Gaussian Sampling preserves information that is otherwise lost such as size, shape, and orientation. Note that opacity is not shown in this example.

pixel grids since convolution on pixels is generally not permutation invariant, for example with the Edge-Detection kernel $\omega = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$ which assigns different weights on different pixels based on their order and so will give different outputs given a reordering of the pixel inputs which is to say is permutation variant, so SelectionConv also presented a novel graph network that can distinguish between different edges. This novel graph network involves multiple adjacency and weight matrices for each node, with a *selection function* that associates vertices with 2D spatial positions. Each selection m has associated adjacency and weight matrices, the former given by

$$\mathbf{S}_{m_{i,j}} \begin{cases} 1 & \text{if } s(v_i, v_j) = m \\ 0 & \text{if } s(v_i, v_j) \neq m \end{cases}$$

And so the graph convolution operator, typically given as

$$\mathbf{X}^{k+1} = \tilde{\mathbf{A}}\mathbf{X}^{(k)}\mathbf{W}$$

with activated node $\mathbf{X}^k$, normalized adjacency matrix $\tilde{\mathbf{A}}$, and weights $\mathbf{W}$ [26], becomes

$$\mathbf{X}^{k+1} = \sum_m \tilde{\mathbf{S}}_m \mathbf{X}^{(k)} \mathbf{W}_m$$

after normalizing $\mathbf{S}_m$ to $\tilde{\mathbf{S}}_m$, i.e. combining any occurrences of multiple nodes within the same selection. Each selection then can support unique behavior in some particular direction. Figure 3.6 illustrates how, with proper application, this operator allows convolution on a graph network to mimic convolution on a pixel grid.

One effect not captured in this method is the relationships a node can have with multiple weights. To account for that, Interpolated Selectionconv introduces two different methods for performing interpolation, a Barycentric Interpolation and an Angular Interpolation.

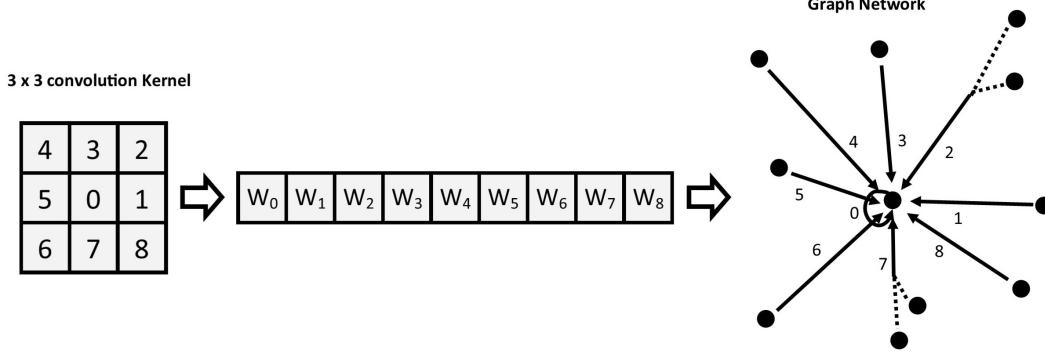


Figure 3.6: Transfer of weights from a 3 x 3 Kernel to a graph network. Each selection may combine multiple nodes. The distances between selections are not necessarily equal, unlike in a pixel grid. Interpolated Selectionconv improves on this aspect.

Figure 3.7 characterizes the difference in this interpolated approach compared to the original. Additionally, the selection normalization method from SelectionConv is expanded on in Interpolated Selectionconv, becoming a two-step process. First, the interpolation weights across all selections are normalized to total to one

$$\hat{S}_m(i, j) = \frac{S_m(i, j)}{\sum_k S_k(i, k)}$$

Then adjacency matrices are normalized such that all edge weights for a target node are equal to one,

$$\tilde{S}_m(i, j) = \frac{\hat{S}_m(i, j)}{\sum_k \hat{S}_k(i, k)}$$

Thus equipped, Interpolated Selectionconv is capable of mapping from complex 3D meshes to a modified graph network and then performing the same kinds of convolution operations on that network as are performed on the structured grids of 2D images, and so weights learned on 2D images are valid for performing machine learning on modified graph networks derived from complex 3D meshes.

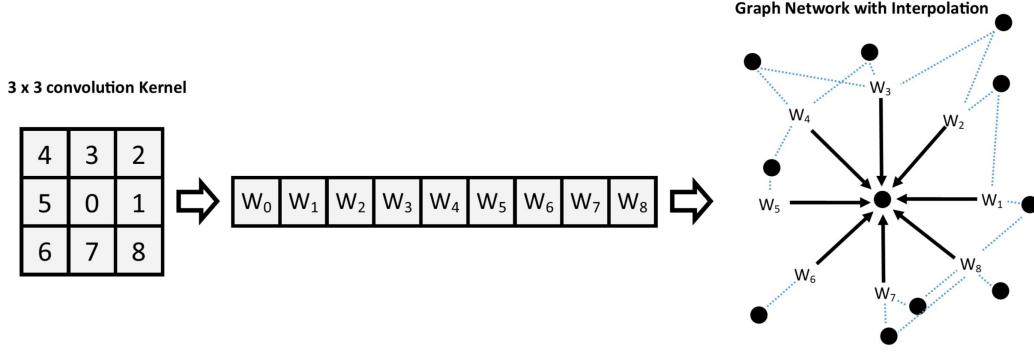


Figure 3.7: Interpolated Selectionconv employs equally-spaced selections in each Cardinal and Ordinal direction whose values are interpolated from any graph nodes with a component in that direction.

3.5 Gaussian Splats to Interpolated SelectionConv

Interpolated Selectionconv employs graph networks and requires information such as up vectors and accurate normal vectors in order to perform the interpolation and weight assignment. This allows the orientation of the cardinal and ordinal directions to remain consistent for the convolution operation on both the 3D surface and on an ordered 2D pixel grid. Satisfying those requirements with the unstructured data in 3D Gaussian splats takes a combination of techniques, as a variety of differences between 3D Gaussian splats and mesh surfaces are apparent. In addition to having unstructured rather than structured data like a mesh, 3D Gaussian splatting is ultimately a volume rendering technique, not a surface rendering technique, and so strictly speaking operations defined on a surface, or the projection of a surface onto a graph network or the 2D plain, are undefined when operating on a volume. However, as with the filtering step, here we allow for the assumption that while technically a volume, the typical 3D Gaussian splat that is of interest to a user will in fact be a volumetric representation of some solid object or objects with geometrically well-defined surfaces. And so while the 3D Gaussian splat contains no explicit surfaces, and really little explicit geometric information at all, they will regardless tend to represent objects that do have explicit geometries, surfaces included. And so, while a bit of mathematical rigor is lost, we proceed to take advantage of the optimization and performance strengths of 3D Gaussian

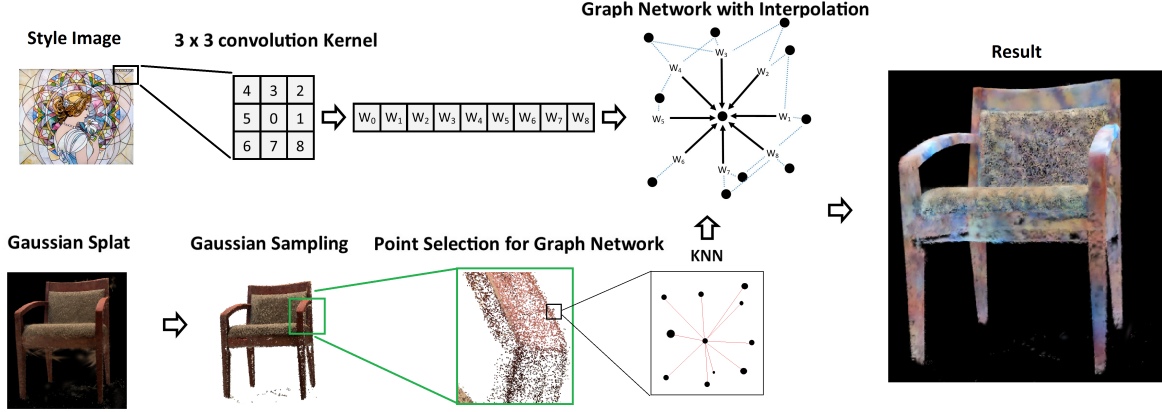


Figure 3.8: Performing Convolution on Gaussian Splats using weights from 2D Image models, showing the general evolution from Gaussian splat to point cloud to graph network.

splatting that come with its being unstructured, while still treating it with the assumption of structure, and performing the analysis and machine learning such assumptions allow for; 3D Gaussian splats themselves, after all, being the outputs of a model trained with a loss function defined on a dataset of 2D structured pixel data, with the edges of objects within those images being the 2D equivalent of surfaces.

Up vectors are the easiest requirement, as an arbitrary up vector can be selected or potentially one can be derived from geometric information implied by the Gaussian splat if desired. As noted previously, in order to map to a graph network, the 3D Gaussian splat is first converted to a One-to-One Point Cloud and filtering, if opted for, is performed. Then a denser point cloud that preserves content information is produced using Gaussian Sampling. At this stage, Neighborhood-Approximated normal vectors could potentially be transferred to the new point cloud in the same way that colors and opacity are, or potentially could be calculated again using the same Neighborhood Approximated Normal calculation. However, while Neighborhood-Approximated normals are expedient for filtering purposes, superior methods for calculating unit normal vectors exist. Here, the *Ball-Pivoting* sampling method of [3] is used via Open3D’s implementation.

Each point in the point cloud is assigned to a node in a graph network. The KNN algorithm can then be used for each point to collect the nearest points, which then becomes

a set of connected nodes in the graph network, as seen in figure 3.8. That network is operated on by the novel convolution operator of [13], thereby allowing machine learning tasks to be performed on 3D Gaussian splats by already trained, highly effective 2D models. In this case, the task being style transfer. Model outputs on the graph network are transferred back to the dense point cloud. At this step, we produce a new One-to-One Point Cloud and assign it color values from the dense point cloud containing the model outputs. That assignment is performed using a standard linear interpolation to calculate the new color values at each point in the One-to-One Point Cloud given the colors of the nearby points in the dense point cloud containing the model output, the result being a point cloud made up of all points in P but with novel color values determined by the style transfer. Finally, the opacity, scaling, and rotation vectors from the original 3D Gaussian are appended to the point and novel color vectors to produce a new, style transferred 3D Gaussian splat.

Our expectation is that the approach outlined so far will be capable of performing a variety of convolution based machine learning tasks and will be capable of transferring the weights of any 2D convolution model. For our present purposes, we have chosen to implement style transfer. This choice was informed by fact that style transfer on 2D images can be performed through 3x3 convolution kernels. Limiting the kernel size reduces the computational costs, first, but also reduces the complexity of the operation and the likelihood of pernicious but hard to characterize effects on our final results, and so could lead to poor feedback loops. Given that at the current stage our goal is in part the fine-tuning of the various mechanisms involved in this novel approach, a responsive feedback loop is of considerable importance. Hence 3x3 kernels, but with the expectation of other kernel types and learning operators in future work. And so the end-to-end pipeline for style transfers on Gaussian Splats using models trained on 2D image data is: from Gaussian splat to point cloud via filtering and Gaussian Sampling, then KNN selection maps points to a modified graph network for style transfer using Interpolated Selectionconv, and finally back to point cloud and then Gaussian splat. A scheme of the whole pipeline is included as Figure 3.9

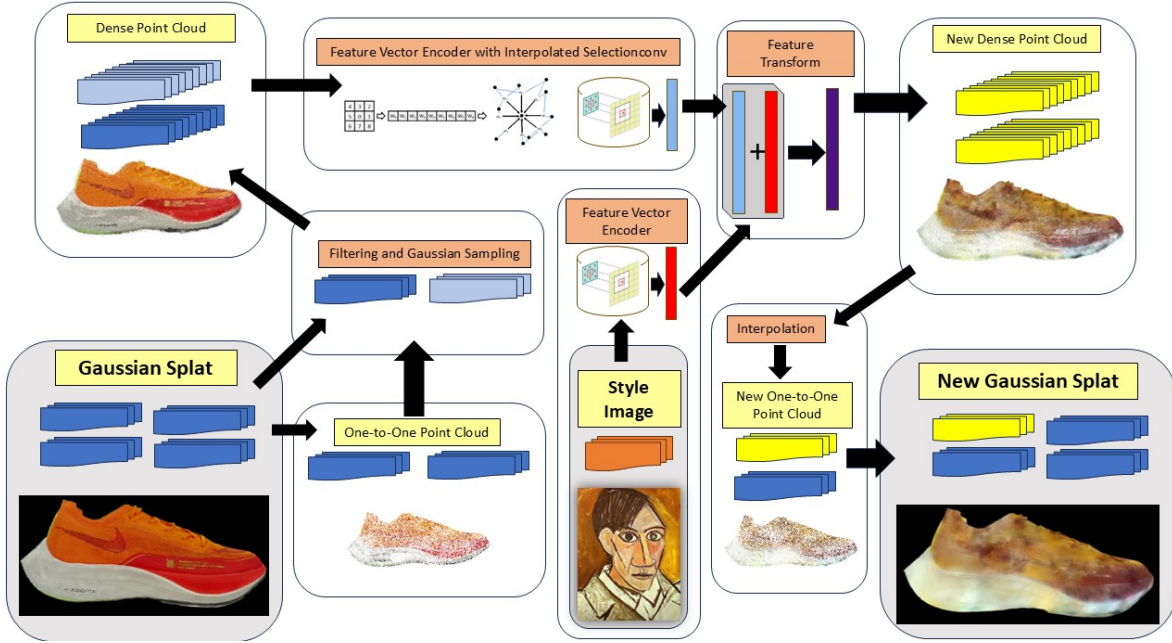


Figure 3.9: Our pipeline, from Original Gaussian splat to Style Transferred Gaussian splat. Stages immediately after the Original 3DGS and before the new 3DGS are both One-to-One Point Clouds. Filtering and Gaussian Sampling produce a Dense Point Cloud which is transformed to a Modified Graph Network and operated on by Interpolate Selectionconv to produce a Feature Vector. A normal CNN operation on the Style image likewise produces a Feature Vector. Vectors are Transformed and the model output is mapped back to a Dense Point Cloud. A final Interpolation yields the point cloud from which the Style Transferred 3D Gaussian splat is obtained.

Chapter 4

Results and Discussion

The results demonstrate the effectiveness of the new method for style transfers on Gaussian splats using Neural Networks trained on 2D data. Additionally, ablation studies show significant impacts for the Neighborhood-Approximated-Normal-Based Filtering, for Ball-Pivoting Normal Estimation, and in particular for Gaussian Sampling.

4.1 Ablation Study: Filtering

Neighborhood-Approximated-Normal-Based Filtering showed a high degree of reliability in segregating between individual Gaussian splats centered on points on and along the surface of objects being represented and those with large offsets from that surface. Some well-trained 3D Gaussian splats from public repositories, in particular those made available in [27], required no filtering. Other publicly available 3DGSs, and in particular user-generated Gaussian splats, do require filtering. User-generated splats suffered from low training times and low-resolution reference images, and thus contained a high percentage of artifacts and noise.

To study the effects of Neighborhood-Approximated-Normal-Based Filtering, we examined several different cases. The first was a test case constructed by generating a geometrically simple Gaussian splat containing noisy points. The second and third cases involved user-generated Gaussian splats.

For the first case, a programmatically-derived Gaussian splat made up of 5000 equal sized

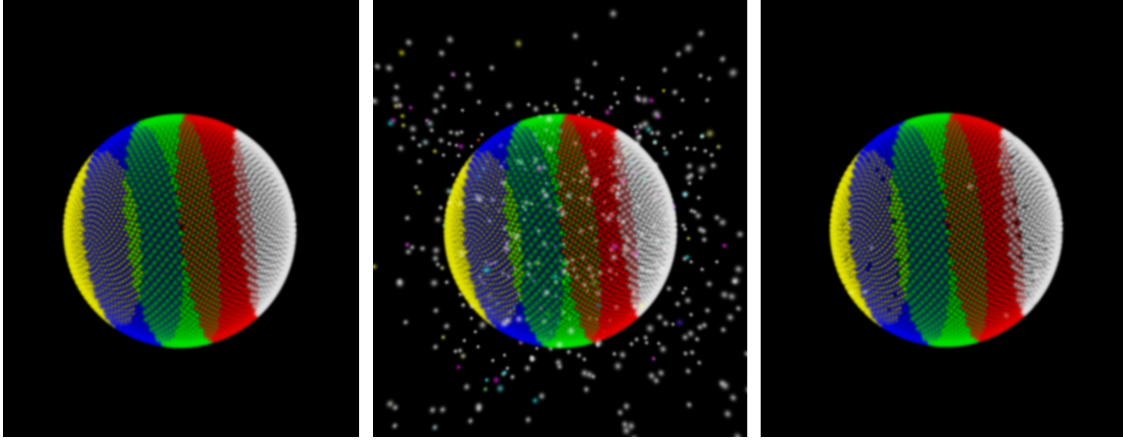


Figure 4.1: A 3D Gaussian of a sphere (Left) with a large number of points added at random distances from the surface to simulate noise (Middle) and the same after Neighborhood-Approximated-Normal-Based Filtering (Right)

Gaussians distributed at equidistant points along the surface of a unit sphere was generated. This was seeded with an additional 500 Gaussian splats centered on points offset from the surface. Offsetting was accomplished by randomly dispersing the points within a $2 \times 2 \times 2$ volume centered on the unit sphere. Neighborhood-Approximated-Normal-Based Filtering was applied to remove points not along the surface of the sphere. Demonstrating remarkable accuracy, the filtering being 98.87% correct. Results are shown in Table 4.1 and in Figure 4.1

In the second and third cases, the Scaniverse mobile app was used to create 3D Gaussian splats of household objects, in this case a chair and a child’s play table with a train set. Those files were found to be particularly noisy and so were excellent candidates for this study. Before and after filtering results are presented. Figure 4.2 shows the 3D Gaussian splat in its original state and after filtering along the top row, as well as point cloud representations created by collecting all the points the individual Gaussian splats are centered on. Given that Neighborhood-Approximated-Normal-Based Filtering operates on point clouds generated in this manner, this second representation is particularly helpful in revealing how the filtering behaves. Isolated points are preferentially filtered, while points along the implicit surface of the object, in this case a chair, are not filtered. Also not well filtered are points that are

noisy but tightly clustered.

The third case, shown in Figure 4.3, showcases the power of this filtering method. In addition to some highly visible artifacts along the bottom edges of the object, entire regions of this particular 3D Gaussian splat were occluded with a haze of noise. After converting the 3DGS to a point cloud, applying Neighborhood-Approximated-Normal-Based Filtering, and converting the point cloud back to a 3DGS, it was evident that much of the noise had been removed, while few, though a non-zero amount, of the individual Gaussian splats representing the table itself were removed.

Table 4.1: For a programmatically generated Gaussian splat of a sphere seeded with additional points with random offsets from the surface, Neighborhood-Approximated-Normal-Based Filtering was applied to remove points not along the surface of the sphere. The filtering method removed all but 0.09% of points not along the surface, while correctly leaving 98.87% of points along the surface.

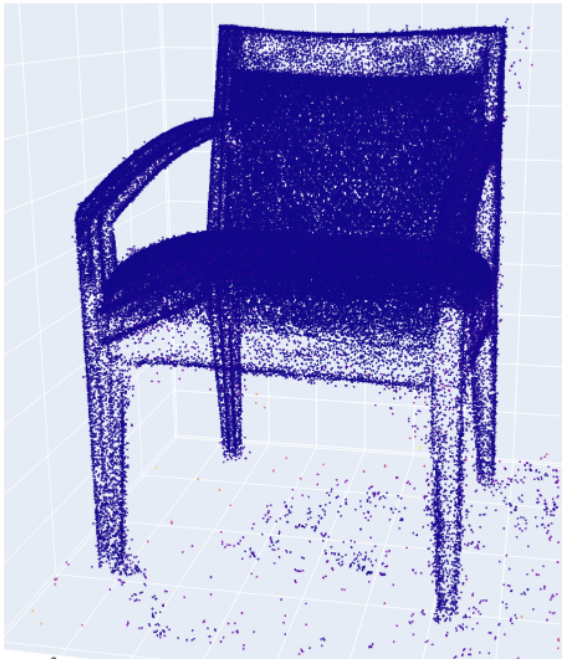
Failure Mode	Rate
False Positives	1.13%
False Negatives	0.09%
Total Success Rate	98.78%



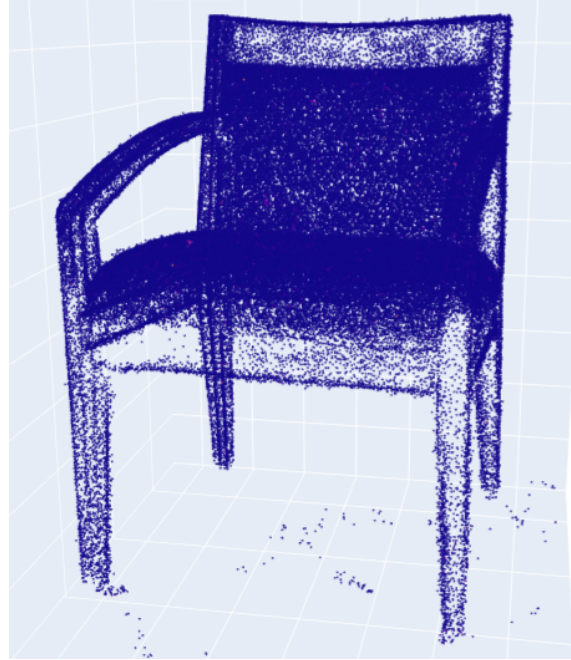
Original 3DGS



Filtered 3DGS



Original 3DGS as a Point Cloud



Filtered 3DGS as a Point Cloud

Figure 4.2: A 3D Gaussian splat that contains a large number of artifacts (Top) and noise both before (Left) and after (Right) being filtered using Neighborhood-Approximated-Normal-Based Filtering. Simplified representations of the 3DGS formed by creating a point cloud with the central point of each individual Gaussian splat (Bottom) demonstrate how isolated points and points well-outside the implicit surface are filtered.

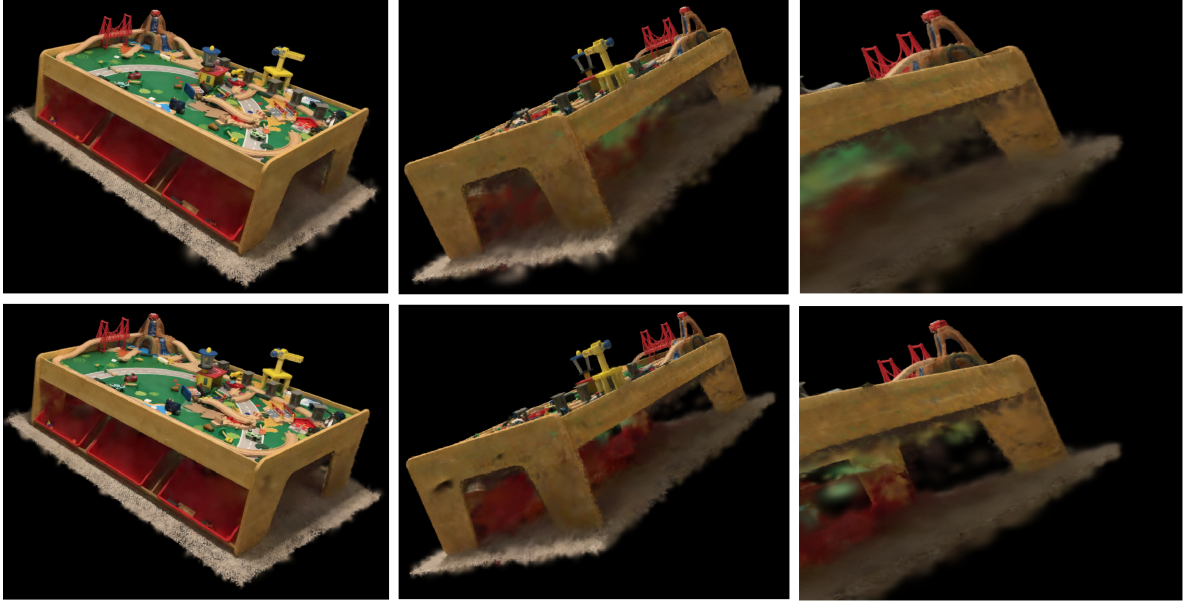


Figure 4.3: A real world Gaussian splat with significant levels of noise (Top) and the same after Neighborhood-Approximated-Normal-Based Filtering (Bottom). Different views of provided. Particularly noisy regions include the edges of the carpet, with plume-like artifacts in several locations, and in particular the underneath of the back side of the table. The carpet-edge noise in the filtered view is entirely absent, and while artifacts remain the hazy underside of the table is no longer largely opaque, a result we consider a significant improvement.

4.2 Ablation Study: Sampling

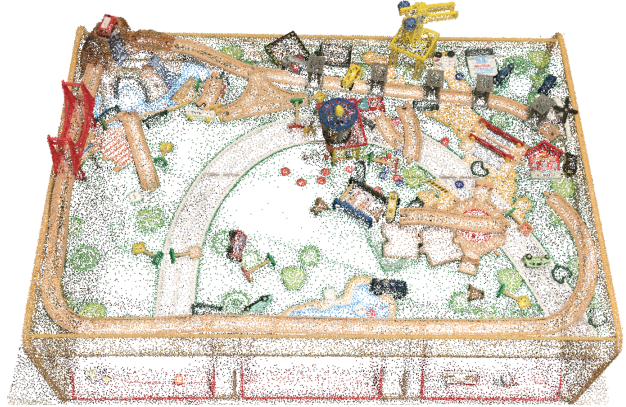
In addition to studies on Neighborhood-Approximated-Normal-Based Filtering, two studies of the effects and efficacy of Gaussian Sampling were undertaken.

An initial point cloud is created by collecting all points around which individual Gaussian splats are centered. Neighborhood-Approximated-Normal-Based Filtering is performed on this point cloud, a purpose to which the point cloud is well-suited. But such a sparse point cloud proved inadequate for preserving important geometric information, as discussed in Chapter 3, and so Gaussian Sampling is employed to increase point cloud densities and preserve more content information and improve the style transfer results.

Figure 4.4 shows a down-facing view of a 3D Gaussian splat of a child’s play table with a train set. Small items are dispersed about the table, and there are illustrations painted on the table itself, both of fine details and also of large monochrome areas with few to no features. When converted to the initial One-to-One point cloud, some small items, particularly those with numerous edges, are well-preserved. Additionally, the fine details painted on the table are moderately well-preserved. However, the less detailed of the small items show some loss of content, and the featureless painted sections of the table are almost entirely lost. In contrast, the dense point cloud after Gaussian Sampling, while not perfect in preserving content and geometries, is substantially less lossy. All small items are well-preserved, as are fine details of the painted surface. Additionally, with the exception of a few particularly large regions, most of the monochrome, featureless sections of the painted table were preserved. Even in the large regions, point density was found to be several times higher, albeit still much lower than on other regions of the table. Regardless, in general even these regions contained sufficient points to preserve content, which enabled the graph network built from the Gaussian Sampled point cloud, upon which the style transfer was performed, to also preserve that content.



Original 3D Gaussian



Point Cloud of Original



Point Cloud after Gaussian Sampling

Figure 4.4: A point cloud of the 3D Gaussian from the previous figure demonstrating how sparse a dataset is produced when naively selecting only the points available in the source file, i.e. the centers of each individual Gaussian splat, and how much information is lost (Top) and a point cloud produced by performing Gaussian Sampling showing a considerably more faithful representation of the object (Bottom). The graph network on which style transfer is performed is constructed from a point cloud representation, and so any content lost in the point cloud will be lost in the final style-transferred 3D Gaussian

Our second result, Figure 4.5, highlights the effects of Gaussian Sampling on the success and accuracy of the style transfer itself. A high quality 3D Gaussian splat of a coffee machine with considerable intricate details and even sections of legible lettering was converted to an initial point cloud, that point cloud was then mapped to a modified graph network equipped with the convolution operator from Interpolated Selectionconv and style transferred. In a second run, a point cloud was generated from that same 3D Gaussian splat using Gaussian Sampling, mapped to a graph network, and style transferred. As the figure demonstrates, the run without Gaussian Sampling showed considerable loss of detail, and even some content, such as some holes in the drip tray and much of the shape and contours of the milk steamer and the buttons, were lost. Additionally, the style transfer itself, while successful, shows a high degree of blurring and loss of contrast. Whereas the run with Gaussian Splatting preserved much of the content of the original Gaussian splat, included some of the lettering and even many of the button icons. Additionally, sections of the drip tray show very good style transfer results, as does the milk steamer and sections of the spout. Figure 4.5 shows a clear example of the extent to which Gaussian Sampling both preserves content and improves style transfer results.

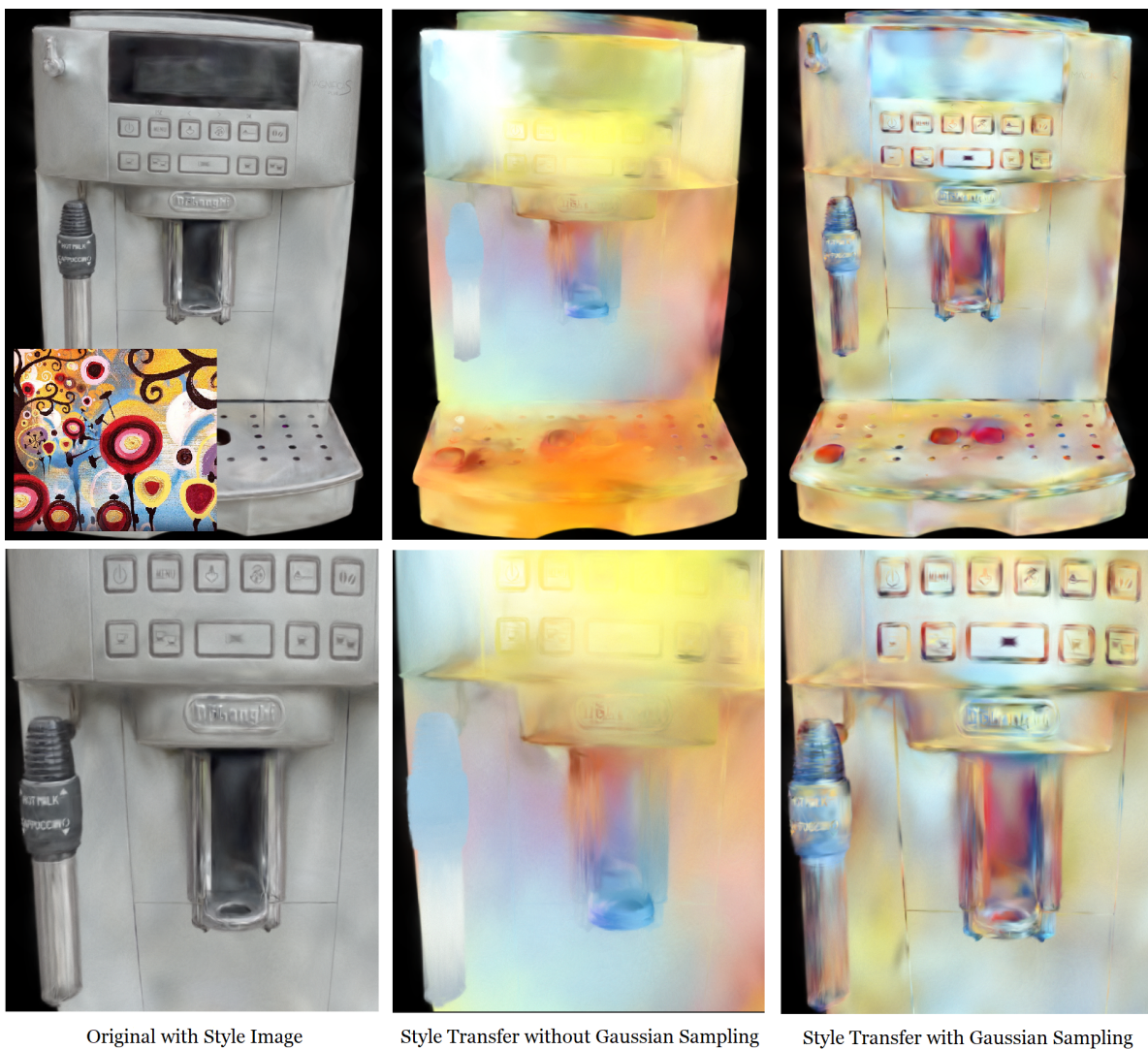


Figure 4.5: A high quality 3D Gaussian splat (Left) after style transfer using a point cloud created from selecting the centers of individual Gaussians (Middle) and using a point cloud with Gaussian Sampling to reduce information loss (Right) demonstrating the effectiveness of applying Gaussian Sampling. Lettering detail shown in bottom row.

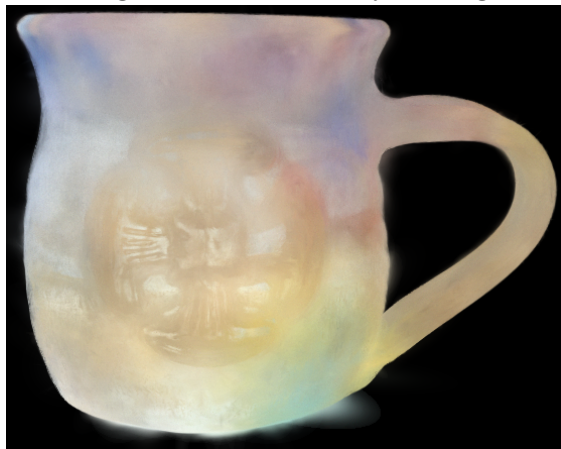
4.3 Ablation Study: Normals

Two different normal vector calculation methods are employed in this work. For filtering, a calculation method that produces normal vector magnitudes that correlate strongly with the spatial properties of the local neighborhood of points was desired. However, normals used in the interpolation step of Interpolated Selectionconv do not benefit from this characteristic, and instead should be unit normal vectors calculated with respect to the global surface of the object being represented in the 3D Gaussian splat. As such, after Gaussian Sampling, normals were re-calculated using the Ball-Pivoting algorithm. Additionally, the Ball-Pivoting algorithm implementation used in this work has proven to be more performant than the calculation used for Neighborhood-Approximated normal vectors, though neither calculation is especially costly.

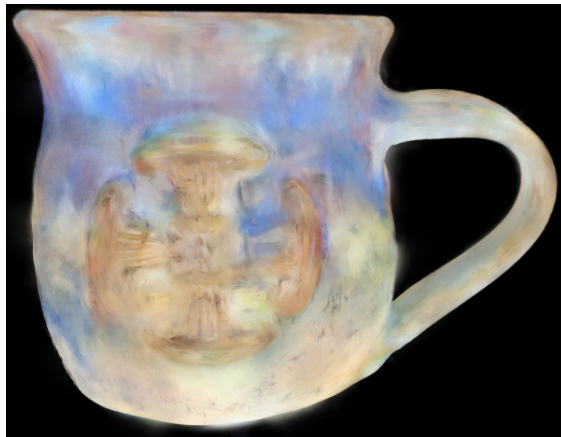
The requirement for accurate normals is evident even for geometrically simple objects. Figure 4.6 shows a style transfer performed on a glazed, handmade pottery mug or cup. To highlight the impact of good accuracy, a style transfer with random normals was compared with a style transfer with Ball-Pivoting calculated normals. In the randomized normals example, the style transfer showed the characteristic blurriness of a successful but imperfect style transfer, likely due to the convolution operator not being able to preserve a global orientation. The style transfer with calculated normals, meanwhile, shows a particularly good style and content match, with not only the decorative figure remaining distinct but even different sections of the glazing showing distinct elements of the style.



Original 3DGS and Style Image



Style Transfer with Randomized Normals



Style Transfer with Calculated Normals

Figure 4.6: A 3D Gaussian splat of a simple object (Top) would normally be a straightforward subject for style transfer. Accurately calculating normals is shown to be an important step however, as style transfer with randomized normals (Middle) was excessively blurry compared to style transfer with normals calculated with the Ball-Pivoting algorithm (Bottom).

4.4 Style Transfer Results

We show the impact of Gaussian splat quality on style transfer quality in Figure 4.7. High quality 3D Gaussian splats demonstrated excellent style transfer results, with good color alignment and excellent content preservation. Lower quality 3D Gaussian splats suffered at times from poor mapping of style elements to content and regions of sparse data. Noisy data also degrades the overall result, though that effect is considerably reduced after Neighborhood-Approximated-Normal-Based Filtering and Gaussian Sampling. Color alignment between the style image and the style transferred Gaussian splats however is consistently good.

Figure 4.8 demonstrates superior results for style transfers. Content is preserved, lettering remains legible, color alignment with the style images is very good, and elements of the style image map well to sections of content, often with high contrasts and sharp transitions in style between different sections of content.

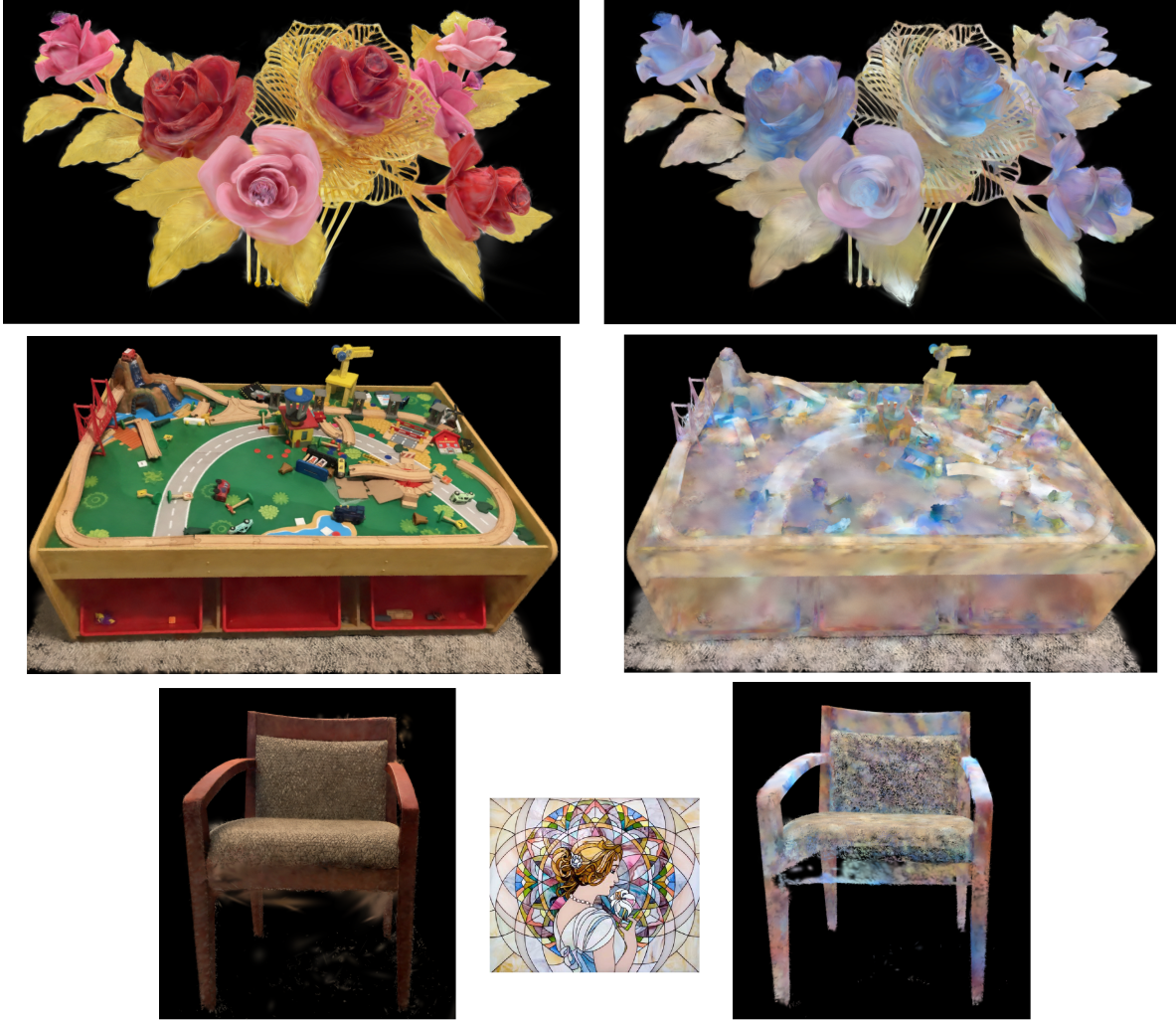


Figure 4.7: Three different Gaussian splats of various degrees of quality under the same style transfer. Style Image shown in bottom-center. Superior differentiation of content is achieved in the highest quality Gaussian splat (Top) despite the very high geometric complexity of the object. In the lower quality Gaussian splats (Middle and Bottom), significant issues can be found in areas dominated by noise or in areas of with sparse data. Color alignment with the style image remains high, however, in all three.

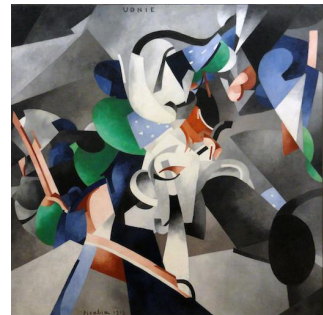
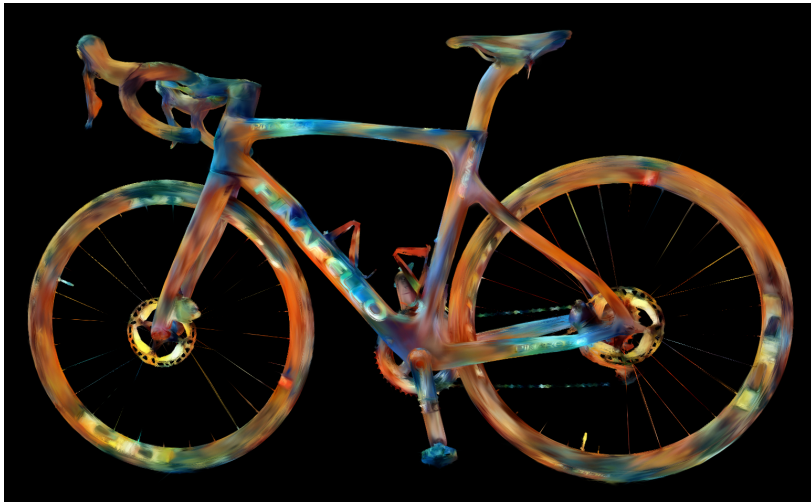


Figure 4.8: A 3D Gaussian under two different style transfers demonstrating superior results in content preservation and color fidelity.



Figure 4.9: A comparison of our results with StyleGaussian with two different style transfers. While content is similar across both, StyleGaussian achieves sharper results. However, our results show better color alignment.

4.5 Comparison with StyleGaussian

StyleGaussian is a recently proposed 3D Convolutional Neural Network that performs style transfers on 3DGSs at near-real time speeds of 10 frames per second. It is based on a novel KNN 3D Convolutional Neural Network and employs several optimization strategies to achieve this performance [37]. As mentioned in Chapter 2 however, StyleGaussian, like all other related and concurrent work on style transfers on 3D Gaussian splats, relies on a resource-intensive training step for custom 3D models prior to being able to perform any machine learning task.

Here, we compare our outputs with those of StyleGaussian. Figure 4.9 includes three different views of a 3DGS under two different style transfers, both for our results and for StyleGaussian. StyleGaussian exhibits good content preservation and in particular has excellent sharpness. Our results have comparable content preservation, less sharpness, but better color alignment with the style images. Figure 4.10 and Figure 4.11 both show a complex scene under style transfer. For Figure 4.10, the results emphasized different sections



Figure 4.10: A 3D Gaussian (Top) our result (Middle) and StyleGaussian (Bottom) showing comparable success in style transfer.

of the style image but are otherwise very similar. For Figure 4.11, StyleGaussian achieved considerably better content preservation, however, color alignment is substantially worse, in particular with overly rich reds compared to the style image and with a paucity of blues from the style image.

In conclusion, we find the results to be comparable with each other. Content preservation is somewhat higher in results generated by StyleGaussian. The color alignment between the style images and the style-transferred 3D Gaussian splat ranges from equally good in both to somewhat better in our results. Our approach, then, demonstrates results equal to the state of the art, all the while our approach involves zero training time and can run for any content 3D Gaussian splat and style image.



Figure 4.11: A 3D Gaussian (Top) our result (Middle) and StyleGaussian (Bottom) showing differential success in style transfer. StyleGaussian shows better content preservation while our result shows closer color matching to the style image

Chapter 5

Conclusion and Future Work

We find that our approach presents both a novel and an effective method for performing convolution on 3D Gaussian splats that largely obviates the issues typically involved with machine learning in this medium. We are considering a variety of improvements, further questions, and future work to build on the success of our approach. Our filtering method is effective and efficient, but refinements suggest themselves. Isolated points are easily detected, groups of noisy points that cluster together less so without over-filtering. A clustering-then-filtering step may address this. Density-aware regional adjustments to the filtering process may also prove useful. We would also like to explore further variations on our formula for the Importance vector in the Gaussian Sampling step, in particular, enabling it to dynamically account for difference in the relative scale of the smallest and largest individual Gaussian splat or enabling it to adjust the bias between the size and opacity terms. Beyond refinements of our approach, in our future work we anticipate also extending it to other operations and tasks. Larger kernel sizes are a particularly low-hanging fruit, though questions about the increase in error propagation in larger kernels, in particular errors introduced by the interpolation steps and by distortions from projecting down to 2D to perform convolution, are largely unanswered. Attention networks may prove to be an interesting operation to define on Gaussian splats. More immediately, we plan to extend our results to other 3x3 convolution based models to perform tasks such as image—or in this case volume—segmentation and object detection and recognition.

BIBLIOGRAPHY

- [1] AJ CHAVAR, M. Pushing the limits of gaussian splatting for spatial storytelling. <https://web.archive.org/web/20241223202907/https://rd.nytimes.com/projects/gaussian-splatting/>. Accessed: 2025-02-10.
- [2] ALEKSANDROV, M., ZLATANOVA, S., AND HESLOP, D. J. Voxelisation algorithms and data structures: a review. *Sensors* 21, 24 (2021), 8241.
- [3] BERNARDINI, F., MITTLEMAN, J., RUSHMEIER, H., SILVA, C., AND TAUBIN, G. The ball-pivoting algorithm for surface reconstruction. *IEEE transactions on visualization and computer graphics* 5, 4 (2002), 349–359.
- [4] CARON, M., TOUVRON, H., MISRA, I., JÉGOU, H., MAIRAL, J., BOJANOWSKI, P., AND JOULIN, A. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision* (2021), pp. 9650–9660.
- [5] CHAN, A. Real-time lighting with gaussian splats. <https://web.archive.org/web/20240715123309/https://andrewkchan.dev/posts/lit-splat.html>. Accessed: 2025-02-8.
- [6] CHEN, S. E., AND WILLIAMS, L. View interpolation for image synthesis. In *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*. Association for Computing Machinery, 1993, pp. 423–432.
- [7] CRASSIN, C., NEYRET, F., LEFEBVRE, S., AND EISEMANN, E. Gigavoxels: Ray-guided streaming for efficient and detailed voxel rendering. In *Proceedings of the 2009 symposium on Interactive 3D graphics and games* (2009), pp. 15–22.
- [8] FUKUSHIMA, K. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics* 36, 4 (1980), 193–202.
- [9] GALERNE, B., WANG, J., RAAD, L., AND MOREL, J.-M. Sgsst: Scaling gaussian splatting styletransfer. *arXiv preprint arXiv:2412.03371* (2024).
- [10] GATYS, L. A., ECKER, A. S., AND BETHGE, M. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 2414–2423.

- [11] HAN, X.-F., JIN, J. S., WANG, M.-J., JIANG, W., GAO, L., AND XIAO, L. A review of algorithms for filtering the 3d point cloud. *Signal Processing: Image Communication* 57 (2017), 103–112.
- [12] HART, D., WHITNEY, M., AND MORSE, B. Selectionconv: convolutional neural networks for non-rectilinear image data. In *European Conference on Computer Vision* (2022), Springer, pp. 317–333.
- [13] HART, D., WHITNEY, M., AND MORSE, B. Interpolated selectionconv for spherical images and surfaces. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision* (2023), pp. 321–330.
- [14] HE, T., HONG, L., KAUFMAN, A., VARSHNEY, A., AND WANG, S. Voxel based object simplification. In *Proceedings Visualization'95* (1995), IEEE, pp. 296–303.
- [15] HEDMAN, P., PHILIP, J., PRICE, T., FRAHM, J.-M., DRETTAKIS, G., AND BROSTOW, G. Deep blending for free-viewpoint image-based rendering. *ACM Transactions on Graphics (ToG)* 37, 6 (2018), 1–15.
- [16] HERMOSILLA, P., RITSCHER, T., AND ROPINSKI, T. Total denoising: Unsupervised learning of 3d point cloud cleaning. In *Proceedings of the IEEE/CVF international conference on computer vision* (2019), pp. 52–60.
- [17] HU, Y., GONG, R., SUN, Q., AND WANG, Y. Low latency point cloud rendering with learned splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2024), pp. 5752–5761.
- [18] HUANG, X., AND BELONGIE, S. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE international conference on computer vision* (2017), pp. 1501–1510.
- [19] IOANNOU, E., AND MADDOCK, S. Depth-aware neural style transfer using instance normalization. *arXiv preprint arXiv:2203.09242* (2022).
- [20] JAGANATHAN, V., HUANG, H. H., IRSHAD, M. Z., JAMPANI, V., RAJ, A., AND KIRA, Z. Ice-g: Image conditional editing of 3d gaussian splats. *arXiv preprint arXiv:2406.08488* (2024).
- [21] JAIN, S., KUTHIALA, A., SETHI, P. S., AND SAXENA, P. Stylesplat: 3d object style transfer with gaussian splatting. *arXiv preprint arXiv:2407.09473* (2024).
- [22] JOHNSON, J., ALAHI, A., AND FEI-FEI, L. Perceptual losses for real-time style transfer and super-resolution. In *Computer Vision—ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part II 14* (2016), Springer, pp. 694–711.
- [23] KATO, H., USHIKU, Y., AND HARADA, T. Neural 3d mesh renderer. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018), pp. 3907–3916.

- [24] KERBL, B., KOPANAS, G., LEIMKÜHLER, T., AND DRETTAKIS, G. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.* 42, 4 (2023), 139–1.
- [25] KIM, H. J., AND SWAN, C. C. Voxel-based meshing and unit-cell analysis of textile composites. *International Journal for Numerical Methods in Engineering* 56, 7 (2003), 977–1006.
- [26] KIPF, T. N., AND WELLING, M. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).
- [27] KOBRANOV, V. Vladkobranov/splats. <https://huggingface.co/VladKobranov/splats/tree/main>. Accessed: 2025-03-23.
- [28] KOVÁCS, Á. S., HERMOSILLA, P., AND RAIDOU, R. G. G-style: Stylized gaussian splatting. In *Computer Graphics Forum* (2024), vol. 43, Wiley Online Library, p. e15259.
- [29] KOVÁCS, Á. S., HERMOSILLA, P., AND RAIDOU, R. G. Surface-aware mesh texture synthesis with pre-trained 2d cnns. In *Computer Graphics Forum* (2024), vol. 43, Wiley Online Library, p. e15016.
- [30] KRIZHEVSKY, A., SUTSKEVER, I., AND HINTON, G. E. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems* 25 (2012).
- [31] KUMAR, B. Convolutional neural networks: A brief history of their evolution. <https://web.archive.org/web/20240419014844/https://medium.com/appyhigh-technology-blog/convolutional-neural-networks-a-brief-history-of-their-evolution-ee34055685971>. Accessed: 2025-02-10.
- [32] LAINE, S., AND KARRAS, T. Efficient sparse voxel octrees. In *Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games* (2010), pp. 55–63.
- [33] LECUN, Y., BOTTOU, L., BENGIO, Y., AND HAFFNER, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86, 11 (1998), 2278–2324.
- [34] LI, C., AND WAND, M. Precomputed real-time texture synthesis with markovian generative adversarial networks. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part III* 14 (2016), Springer, pp. 702–716.
- [35] LI, X., LIU, S., KAUTZ, J., AND YANG, M.-H. Learning linear transformations for fast image and video style transfer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2019), pp. 3809–3817.
- [36] LIANG, Y., HE, F., AND ZENG, X. 3d mesh simplification with feature preservation based on whale optimization algorithm and differential evolution. *Integrated Computer-Aided Engineering* 27, 4 (2020), 417–435.

- [37] LIU, K., ZHAN, F., XU, M., THEOBALT, C., SHAO, L., AND LU, S. Stylegaussian: Instant 3d style transfer with gaussian splatting. In *SIGGRAPH Asia 2024 Technical Communications*. SIGGRAPH, 2024, pp. 1–4.
- [38] LIU, M., XU, C., JIN, H., CHEN, L., VARMA T, M., XU, Z., AND SU, H. One-2-3-45: Any single image to 3d mesh in 45 seconds without per-shape optimization. *Advances in Neural Information Processing Systems* 36 (2024).
- [39] MAGLO, A., LAVOUÉ, G., DUPONT, F., AND HUDELOT, C. 3d mesh compression: Survey, comparisons, and emerging trends. *ACM Computing Surveys (CSUR)* 47, 3 (2015), 1–41.
- [40] MILDENHALL, B., SRINIVASAN, P. P., TANCIK, M., BARRON, J. T., RAMAMOORTHY, R., AND NG, R. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM* 65, 1 (2021), 99–106.
- [41] MÜLLER, T., EVANS, A., SCHIED, C., AND KELLER, A. Instant neural graphics primitives with a multiresolution hash encoding. *ACM transactions on graphics (TOG)* 41, 4 (2022), 1–15.
- [42] NEYRET, F. Modeling, animating, and rendering complex scenes using volumetric textures. *IEEE Transactions on Visualization and Computer Graphics* 4, 1 (1998), 55–70.
- [43] NIEMEYER, M., BARRON, J. T., MILDENHALL, B., SAJJADI, M. S., GEIGER, A., AND RADWAN, N. Regnerf: Regularizing neural radiance fields for view synthesis from sparse inputs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), pp. 5480–5490.
- [44] POMERLEAU, F., COLAS, F., SIEGWART, R., ET AL. A review of point cloud registration algorithms for mobile robotics. *Foundations and Trends® in Robotics* 4, 1 (2015), 1–104.
- [45] REDMON, J., DIVVALA, S., GIRSHICK, R., AND FARHADI, A. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 779–788.
- [46] REISER, C., PENG, S., LIAO, Y., AND GEIGER, A. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In *Proceedings of the IEEE/CVF international conference on computer vision* (2021), pp. 14335–14345.
- [47] REN, S., HE, K., GIRSHICK, R., AND SUN, J. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence* 39, 6 (2016), 1137–1149.
- [48] ROMAN-RIVERA, L.-R., PEDRAZA-ORTEGA, J. C., SOTELO-RODRÍGUEZ, I., GUEVARA-GONZÁLEZ, R. G., AND TOLEDANO-AYALA, M. 3d point cloud outliers and noise reduction using neural networks. In *International Congress of Telematics and Computing* (2023), Springer, pp. 323–341.

- [49] SAROHA, A., GLADKOVA, M., CURRELI, C., MUHLE, D., YENAMANDRA, T., AND CREMERS, D. Gaussian splatting in style. *arXiv preprint arXiv:2403.08498* (2024).
- [50] SCHWARZ, K., SAUER, A., NIEMEYER, M., LIAO, Y., AND GEIGER, A. Voxgraf: Fast 3d-aware image synthesis with sparse voxel grids. *Advances in Neural Information Processing Systems 35* (2022), 33999–34011.
- [51] SIMONYAN, K. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).
- [52] SITZMANN, V., THIES, J., HEIDE, F., NIESSNER, M., WETZSTEIN, G., AND ZOLLHOFER, M. Deepvoxels: Learning persistent 3d feature embeddings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), pp. 2437–2446.
- [53] SITZMANN, V., ZOLLHÖFER, M., AND WETZSTEIN, G. Scene representation networks: Continuous 3d-structure-aware neural scene representations. *Advances in Neural Information Processing Systems 32* (2019).
- [54] WANG, Q., AND KIM, M.-K. Applications of 3d point cloud data in the construction industry: A fifteen-year review from 2004 to 2018. *Advanced engineering informatics 39* (2019), 306–319.
- [55] XU, J., CHENG, W., GAO, Y., WANG, X., GAO, S., AND SHAN, Y. Instantmesh: Efficient 3d mesh generation from a single image with sparse-view large reconstruction models. *arXiv preprint arXiv:2404.07191* (2024).
- [56] XU, Y., TONG, X., AND STILLA, U. Voxel-based representation of 3d point clouds: Methods, applications, and its potential use in the construction industry. *Automation in Construction 126* (2021), 103675.
- [57] YU, A., YE, V., TANCIK, M., AND KANAZAWA, A. pixelnerf: Neural radiance fields from one or few images. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2021), pp. 4578–4587.
- [58] ZHANG, D., YUAN, Y.-J., CHEN, Z., ZHANG, F.-L., HE, Z., SHAN, S., AND GAO, L. Stylizedgs: Controllable stylization for 3d gaussian splatting. *arXiv preprint arXiv:2404.05220* (2024).
- [59] ZHANG, K., KOLKIN, N., BI, S., LUAN, F., XU, Z., SHECHTMAN, E., AND SNAVELY, N. Arf: Artistic radiance fields. In *European Conference on Computer Vision* (2022), Springer, pp. 717–733.