IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 7 October 2025, accepted 11 November 2025, date of publication 18 November 2025,
date of current version 25 November 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3634276

RESEARCH ARTICLE

MVS-Splatting: Fast Multi-View Stereo Depth Fusion for 3D Gaussian Splatting Initialization

JULIE ARTOIS, PETER LAMBERT, (Senior Member, IEEE), AND GLENN VAN WALLENDAEL, (Member, IEEE)
IDLab, Department of Electronics and Information Systems, imec, Ghent University, 9000 Ghent, Belgium
Corresponding author: Julie Artois (julie.artois@ugent.be)

This work was supported in part by IDLab (Ghent University-imec), in part by the Fund for Scientific Research Flanders (FWO Flanders) under Grant 1SD8221N, in part by the imec.prospect project SitSens; and in part by European Union.

ABSTRACT Images and videos allow us to explore places and connect to people all around the world, in the present or past. What if we could break through the glass screen in front of us and step into those camera captures. Although challenging, in recent years, light field technology has developed some promising techniques, such as 3D Gaussian Splatting, that is able to render high-quality views of a scene reconstructed using only camera captures. However, creating the scene model takes a significant amount of time and compute power, which makes it unviable for the multimedia industry which outputs terabytes of new content daily. In this paper, we present a method of speeding up the modeling process, not by optimizing the training, but by initializing the pipeline with an already semi-finished reconstruction. This is done by estimating the depth maps of the camera images, fusing them and converting this to a dense set of Gaussian splats which already closely resembles the scene. Afterwards, the default training process is applied to fine-tune and quickly synthesize new high-quality views. We show that our method on average, after 1000 iterations, improves PSNR by +1.27 dB, SSIM by +0.065 and LPIPS by -0.10, compared to the default initialization.

INDEX TERMS Depth estimation, Gaussian splatting, immersive media, light fields, multi-view stereo.

I. INTRODUCTION

Every year, advancements in computing hardware and display technologies bring immersive media experiences closer to a wider audience. The goal is to provide the viewer with an experience that is not merely watching from a distance, but *walking around* in a scene. Aside from affordable hardware, another limiting factor at the moment is content. Handcrafting all immersive content using traditional 3D modeling tools is too expensive. Instead, one can capture real-world scenes using camera arrays, and use these to synthesize new views with a degree of photorealism already built in. In this way, we can create content, for example digital twins [1], for applications such as Virtual Reality (VR), Augmented Reality (AR) and telepresence [2], [3].

Although the concept of light fields as a way of synthesizing novel views has existed for over two decades, it is only in the last few years that systems achieve both photorealism *and* real-time rendering for sparsely sampled scenes. Most of the methods that fall into this sweet spot are based on training a representation of the scene using iterative approaches, such as gradient descent. These methods are often optimized to render the scene model in real-time, which also benefits the training duration. Currently, the two methods at the forefront of this category are Neural Radiance Fields (NeRF) [4] and 3D Gaussian Splatting (3DGS) [5], [6]. Of the two, 3DGS takes less time to train and achieves a higher render frame rate [7], making it the most viable option to process a lot of content and display it on mainstream hardware.

3DGS works by modeling a scene as a cloud of 3D ellipsoids called *splats*. Gradient descent is used to optimize the parameters of the splats (e.g. position, rotation, color, transparency, etc.) by rasterizing the splats to the original cameras and calculating the image loss. Every 100 iterations, there is also a densification process, where splats are pruned or cloned where necessary. The iterative nature of the 3DGS

The associate editor coordinating the review of this manuscript and approving it for publication was Jiachen Yang.

training process is what results in the high-quality reconstruction, since it allows for occlusion-aware optimizations. However, it inevitably results in a long training process, with thousands of training iterations. Therefore, producing vast content libraries requires a lot of computational resources.

This paper aims to speed up the training process of 3DGS, which will allow for more scenes to be reconstructed with less computational resources. Typically, the scene model is initialized with a low number of splats, constructed from the sparse point cloud delivered during the Structure-from-Motion (SfM) step (which determined the intrinsics and extrinsics of the cameras). During the training process, over time, the splats move to their correct position and grow into their correct size and opacity. At an even slower pace, more and more splats are introduced where necessary, for example in detailed areas.

Our key idea is to start with a much larger number of already (close to) ''correct'' splats. This effectively skips the early stages of the training process and accelerates the convergence to a high-quality reconstruction. To estimate a good, dense initialization of splats, we propose to use Multi-View-Stereo (MVS) to estimate the depth maps for a subset of the light field images. Each depth map is then turned into a 3D point cloud, which together can be converted to a big cloud of splats. The position, scale and color of these splats will already be close to correct, reducing the need for many training iterations.

To be more precise, in this paper, we propose MVS-Splatting, a framework that is able to deliver a higher-quality 3DGS scene reconstruction using fewer training iterations. It uses a highly-optimized MVS implementation to estimate depth maps, which are converted to point clouds. A fast consistency check removes duplicate points as well as uncertain depth predictions. The point cloud is then converted into 3D Gaussian splats, which make up the initialization of the 3DGS training process. The training hyperparameters are also optimized to output a high-quality result in as few training iterations as possible. The code is made openly available to facilitate reproducibility of the results.[1]

The remainder of this paper is organized as follows. Section II reviews the related work and situates our approach within the existing literature. Section III describes the proposed methodology in detail. Section IV presents the experimental results. Section V discusses the implications of these findings and their limitations. Finally, Section VI concludes the paper and outlines directions for future research.

II. RELATED WORK

Our method builds on prior work in several areas, including SfM, depth estimation, and 3DGS. In this section, we briefly review the most relevant developments in each of these domains and explain how they relate to our proposed approach. The literature around 3DGS is quite extensive, so we focus primarily on the research around improving the initialization of splats, to speed up the training process and improve the scene reconstruction. For a broader overview of the field, we refer the reader to the comprehensive survey by Dalal et al. [7].

A. STRUCTURE FROM MOTION

We want to reconstruct a scene from a dataset of camera captures. The first step consists of estimating the intrinsics and extrinsics of each camera. Methods such as SfM are convenient because they do not require calibration patterns. Several SfM toolkits exist, including OpenMVG [8], AliceVision [9] and COLMAP [10], which is a widely used GPU-accelerated implementation that can typically perform SfM on a medium-resolution image dataset in under a minute. Inherently, SfM also outputs a sparse point cloud of the scene. Note that these points originate only from textured areas and do not sample the scene evenly. Additionally, depending on how the cameras overlap, either the background or foreground will have fewer matched points. Therefore, the sparse point cloud is a suboptimal initialization for methods like 3DGS.

B. DEPTH ESTIMATION

A computationally efficient way to reconstruct a scene from a set of images is to calculate depth maps (or disparity maps), and convert them to a point cloud and/or mesh [11]. By leveraging hardware-accelerated structures from computer graphics, these can be rendered in real time [2], [3], even in demanding use cases such as VR. The simplest case is stereo depth estimation, which estimates a disparity map given two perfect horizontally-aligned rectified images [12]. This is fast, but in most datasets, the cameras have an arbitrary orientation and position. MVS is a group of methods that is able to estimate depth maps for datasets with a wide variety of camera setups. For example, Feng et al. [13] propose an MVS framework with an accelerated patch matching and point cloud generation implementation. COLMAP also offers a GPU-accelerated MVS implementation. However, these MVS frameworks take much longer than 3DGS to arrive at a high-quality scene reconstruction.

In recent years, many depth estimation frameworks rely on machine learning to overcome some of the common challenges such as textureless areas, fine details, and occlusions. One example is monocular depth estimation, where a large neural network is trained to receive an image and directly outputs a depth map. Although fast, the produced ''depth'' does not correspond to any real world scene geometry. Some papers try to rescale the monocular depth to the scene's actual geometry [14], but this is a slow process. We find that our MVS framework estimates depth maps faster than this method of monocular depth estimation and rescaling, while the quality of the 3DGS initialization is similar.

Many papers also use machine learning for MVS [15], [16]. A large dataset with groundtruth depths is used to train a

[1] https://idlabmedia.github.io/mvs-splatting/

neural network to output depth maps, using MVS techniques such as cost volume optimization. Many learning-based MVS methods are trained on datasets such as DTU [17], which feature highly controlled scenes with limited variability, typically a single object placed on a white table with a black background and a small depth range. As a result, models trained on such data often struggle to generalize to real-world scenarios, particularly outdoor environments with complex geometry, variable lighting, and large depth ranges. While more diverse datasets like BlendedMVS [18] have been proposed to address this issue, they still fall short of capturing the full variability encountered in arbitrary scenes.

In conclusion, we find that many existing MVS frameworks are not faster than 3DGS, or do not generalize well to a wide variety of scenes. If the MVS step takes too long, or a manual step to finetune the scenes is necessary, then it would be faster to simply use the default 3DGS initialization and training setup. For this reason, we propose a highly-optimized MVS implementation.

C. 3DGS TRAINING

3DGS models a scene as a set of Gaussian splats, each with a position, rotation, opacity, scale (using a covariance matrix). The color is determined using Spherical Harmonics (SH). The original and consecutive works have achieved high rendering speeds, due to the rasterization pipeline, and a high degree of photorealism [5], [6], [19]. One downside of this technique is that for detailed areas, a lot of small splats are necessary, reducing render and storage efficiency. Regardless, 3DGS lends itself well to photorealistic view synthesis for applications such as VR and telepresence.

However, processing many datasets using 3DGS takes a lot of computing time. There are a number of papers already that try to reduce the necessary training time. For example, Seibt et al. [20] propose to alter the SfM step to allow for more features, which produces a much denser initial point cloud. They report a higher quality 3DGS scene reconstruction in fewer training iterations compared to default 3DGS. However, their approach as a whole ends up being slower than default 3DGS for reaching a certain quality, due to the slow SfM pass. Our method also produces a dense splat initialization, improving reconstruction quality. However, instead of a computationally heavy SfM step, we use a highly-optimized MVS implementation, which cuts the preprocessing time down to only a couple of seconds.

Another method called MVSGaussian [21] trained a neural network to perform MVS and output a dense, colored point cloud. The point cloud is filtered using a geometric consistency check. The point cloud then serves as the initialization for the Gaussian splats. The quality of the point cloud is high for some datasets, but their neural network was trained on the DTU dataset, which consists of objects on a white table with a black background. Therefore, it does not generalize well to other types of scenes, for example large outdoor environments. Additionally, their technique is slow, resulting in the default 3DGS reaching a higher quality faster, as will be discussed in Section IV-C. The reason for the slow performance of MVSGaussian is that it scales poorly as the number of input views increases. The method is meant to be used for two to three input views, not entire datasets. In contrast, our proposed method is designed to scale well for large datasets, by reducing them to a representative subset of key cameras.

To our knowledge, there is no existing framework that provides a high-quality, dense splat initialization for 3DGS, which is fast enough to outperform the default 3DGS training quality. Therefore, in this paper, we present MVS-Splatting, a system which is capable of:

- Overcoming typical MVS challenges through a cross-consistency check and relying on 3DGS' iterative optimization to fill in textureless areas.
- Through a well-optimized MVS framework, produce a dense set of Gaussian splats in under a few seconds.
- Converging to a high-quality scene reconstruction much faster by initializing the 3DGS training process with already close to correct splats.

III. METHODOLOGY

A. OVERVIEW

We give a high-level overview of the proposed pipeline, as seen in Figure 1, before exploring each component in detail in Section III-B. The input to the system is a dataset of camera captures taken of a scene at one moment in time. Next, the camera intrinsics and extrinsics are estimated using Colmap's SfM algorithm. Note that the *undistorted* versions of the images are used for the rest of the pipeline. The SfM pass also produces a sparse point cloud, which is used by the original 3DGS paper to initialize the position and color of the splats. In the proposed approach, we create a much denser collection of splats by building a point cloud from depth maps and converting the points into Gaussian splats.

The depth maps are created using a computationally fast MVS implementation. Estimating a depth map for all cameras would not only be slow, but also redundant, since there is a lot of overlap between cameras. Therefore, a small representative subset of the cameras, denoted as *key cameras*, is automatically chosen, based on how much overlap they have with the other cameras. We carefully select up to four stereo pairs per key camera, since this step can greatly impact the depth map accuracy.

Now, a depth map per key camera can be estimated through a *plane sweep* step, which is a common MVS technique where the reprojection error for different depth planes is calculated and the depth is chosen to minimize the error [22], [23]. This also produces a certainty map, where the certainty decreases when multiple planes have errors close to the minimum. The MVS implementation used in this paper trades off accuracy for speed. Specifically in areas with little texture, fine geometry or view-dependent effects, the certainty of the predicted depth will be low. We can therefore threshold the certainty maps to get a *final mask* indicating which pixels

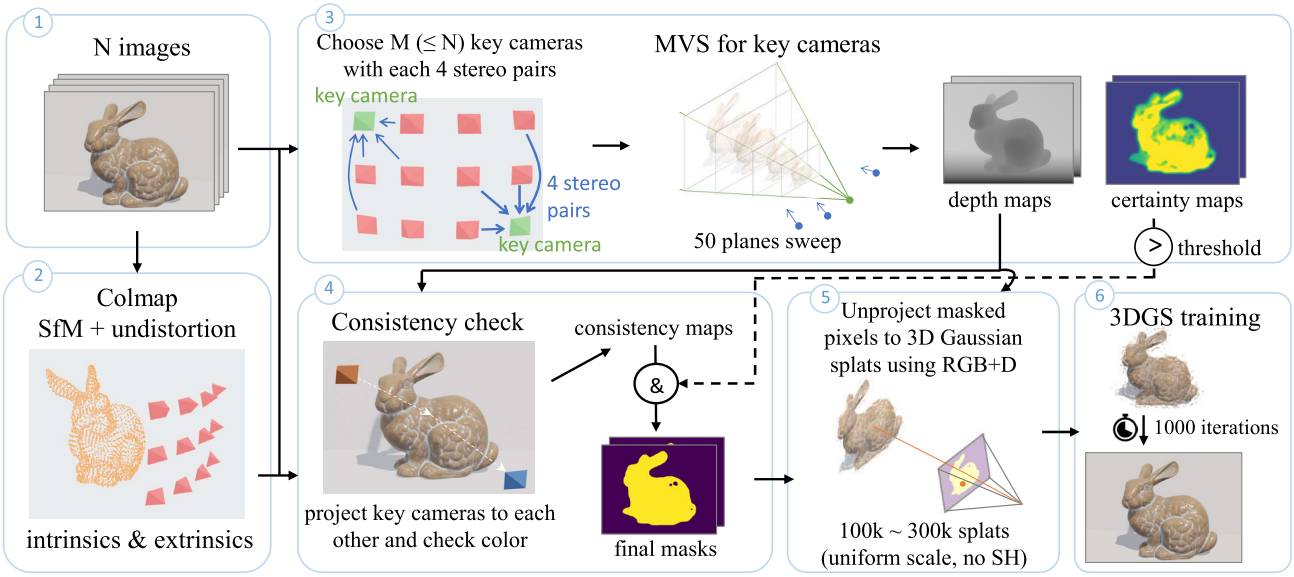

FIGURE 1. Overview of the proposed pipeline. Note that, although not indicated, the (undistorted) images, camera intrinsics and extrinsics from the Colmap step are used in every part of the pipeline.

of the depth map are likely accurate and should be used to produce Gaussian splats.

This is not enough however to eradicate bad depth predictions. We apply an additional *consistency* check, where the RGB+D of each key camera is used to project onto the other key cameras, and a consistency mask is created, indicating if the projected color is similar to the key camera's image. The mask is combined (logical and) with the mask from the certainty maps, resulting in the final mask. This consistency check is helpful to mask off floaters due to depth edges and view-dependent elements.

Finally, the pixels of the images and depth maps are used to initialize the 3D gaussian splats. Only pixels where the mask is true are used. The pixels are subsampled to keep the total number of splats between 100k and 300k. The scale of the splats is chosen so that closely cover the 3D surface. No high-degree spherical harmonics are defined, only the RGB color. From here, the 3DGS training process can be started, just as when the Colmap SfM point cloud would have been used as initialization. However, due to the scale of the splats being initialized in a different way, some of the training parameters need to be slightly tweaked.

B. MULTI-VIEW STEREO

1) KEY CAMERAS

A dataset consists of a set of camera captures of a scene. Each image captures a different part of the scene, but most images have at least some, and often significant, overlap with certain others. If we were to estimate depth maps for every image, and fuse those into a point cloud, most of these points would be duplicates. It is therefore more efficient to boil down the dataset to a small subset of *key cameras*, which have limited overlap *within* the subset, but still see all the objects captured by the original images.

The method for choosing the key cameras is designed to handle a large variety of camera setups. For each pair of cameras (i, j), a small number of evenly-spaced points in the image plane of camera j is defined, then unprojected with a fixed depth $z = 10$, and 3D warped to camera i. By keeping track of which points of camera j camera i sees, we know how the cameras' field of perception overlap with one another. However, if the viewing angle between cameras (i, j) is more than 20°, this calculation is skipped and it is considered as them not seeing any of each others points.

The key camera subset is initialized with the first camera of the dataset. Adding a key camera to the subset comes down to finding the camera that sees the most new points that were not already seen by the previously added key cameras. We stop adding key cameras if 90% of all points is seen by the key cameras.

2) MVS STEREO PAIRS

To estimate an image's depth using MVS, one or more neighboring images are required, forming the *stereo pairs*. For each key image, four neighbor images are chosen, as is common in other work [2], since a lower number would leave the MVS approach more vulnerable to overfitting. Using the overlap calculations from the previous step, we can calculate which camera i should be added to the set of neighbors of key camera j as follows. Camera i is the camera which sees the most 3D points, unprojected from j, which were not already seen by the current set of neighbors of j.

3) PLANE SWEEP

A plane sweep is a standard MVS technique to estimate the depth of an image, given a set of neighbor images. Note that despite the use of the term "neighbors", it is not required that the neighbor cameras are the closest ones to the target image.

Say that key image $\boldsymbol{I}_{\text{key}}$ has four neighbors $\{\boldsymbol{I}_0, \ldots, \boldsymbol{I}_3\}$ and the number of planes considered is $N = 50$, each associated with a certain depth $z_i, i \in [0, N-1]$. For each depth z_i, four error maps (one per neighbor) are calculated as follows. A pixel (row, col) in $\boldsymbol{I}_{\text{key}}$ with color $\boldsymbol{I}_{\text{key}}[\text{row}, \text{col}]$ is unprojected to depth z_i and projected to a neighbor $\boldsymbol{I}_n$ at a pixel denoted as (row′, col′). The error for neighbor n at depth z_i is defined as:

$$\textbf{error}_{n,i}[\text{row}, \text{col}] = \|\boldsymbol{I}_{\text{key}}[\text{row}, \text{col}] - \boldsymbol{I}_n[\text{row}', \text{col}']\| \quad (1)$$

For clarity, the notation $\|.\|$ denotes the Euclidean norm of the vector inside, in this case a difference of (red, green, blue) color values. Next, each error map is smoothed using a weighted box blur, where the weight of each pixel in the box is inversely proportional to its color difference with the center pixel of the box. This will ensure that the final depth map is smooth, except at color edges. Next, the four error maps per plane are reduced to one with a winner-takes-all approach:

$$\textbf{error}_i[\text{row}, \text{col}] = \min_{n \in [0,3]} \textbf{error}_{n,i}[\text{row}, \text{col}] \quad (2)$$

This makes it so that if an object in $\boldsymbol{I}_{\text{key}}$ is occluded in some of its neighbors, leading to a high error, those neighbors will not be used to estimate the depth. Lastly, the depth map of $\boldsymbol{I}_{\text{key}}$ is calculated by, for each pixel, choosing the depth z_i with the lowest $\textbf{error}_i$:

$$\textbf{depth}[\text{row}, \text{col}] = z_{\text{argmin}_i \, \textbf{error}_i[\text{row},\text{col}]} \quad (3)$$

At the same time, the certainty map for $\boldsymbol{I}_{\text{key}}$ is defined as, for each pixel, the number of planes that have an $\textbf{error}_i$ close to the minimal error, divided by he total number of planes to normalize the certainty to range [0, 1].

That concludes the explanation of the plane sweep algorithm. However, we have yet to assign a value to z_i for each plane i. First, the closest and furthest point in $\boldsymbol{I}_{\text{key}}$'s point cloud of the Colmap SfM pass is found. These determine the depth range [near, far] along the Z axis. The depth range is widened slightly, because the SfM point cloud does not necessarily cover the full range of the scene. For a high-quality scene reconstruction, depth precision becomes increasingly important the lower the depth value, which should be reflected in the choice of z_i. We therefore derive a quadratic function that goes through points (0, near) and $(N-1, \text{far})$:

$$z_i = z(i) = \frac{\text{far} - \text{near}}{(N-1)^2} i^2 + \text{near}\,, \; i \in [0, N-1] \quad (4)$$

The depth values follow a quadratically increasing curve, which, if we convert the depths to disparities, roughly translates to evenly-distributed disparities. This is ideal for the plane-sweep algorithm.

4) CONSISTENCY CHECK

The proposed plane-sweep method trades off depth accuracy for execution speed. It often fails for textureless areas, specular highlights, fine geometry, inconsistencies in exposure, etc. The depth map pixels will be used to generate the initial Gaussian splats for 3DGS, and incorrectly positioned splats will significantly lower the quality of the scene reconstruction. It is therefore crucial to delete, or *mask away* badly estimated depth map pixels. We propose a two-step approach. Firstly, the certainty maps (one per key camera) of the previous step are thresholded to create an initial *certainty mask*. This makes sense since, if the certainty is low, there are by definition a lot of planes with a similar error to one another, so it is unlikely that the chosen depth is correct.

Secondly, it is not uncommon for the certainty to be high, but the estimated depth to still be incorrect. This phenomenon happens mainly for specular highlights, exposure inconsistencies and around depth edges. Therefore, a *consistency map* is calculated per key camera, indicating how well 3D warping with the depth maps holds up in terms of projection image error. Each key cameras' color and depth map (RGB+D) is used to create a point cloud, which is projected onto all other key cameras. In other words, of key camera A, a pixel $\boldsymbol{p}$ is unprojected to 3D point $\boldsymbol{P}$, which is projected onto pixel $\boldsymbol{p}'$ of another key camera B. If the following two conditions hold, A's *consistency mask* for pixel $\boldsymbol{p}$ is set to *false* (meaning inconsistent):

- The Z-depth of $\boldsymbol{P}$ (from the perspective of B) is smaller than $\textbf{depth}_B[\boldsymbol{p}']$.
- $\|\textbf{rgb}[\boldsymbol{p}] - \textbf{rgb}[\boldsymbol{p}']\| > 0.1$, with $\textbf{rgb} \in [0, 1]^3$ being the (red, green, blue) color values.

In other words, if 3D warping (with Z-depth testing) $\boldsymbol{p}$ to $\boldsymbol{p}'$ results in a big difference in color, the depth is likely incorrect. The final consistency mask per key camera is simply the *logical and*(&) operation of its consistency mask and certainty mask. In general, this final mask quite accurately indicates where large textureless areas, specular highlights and floaters are located in the depth map.

5) GAUSSIAN SPLAT GENERATION

We aim to generate between 100k and 300k Gaussian splats to initialize the 3DGS training process, regardless of the scene extent. Having more splats leads to more detailed scene reconstructions, but slows down training speeds, partially because VRAM becomes more of a bottleneck. Additionally, it is not necessary to reconstruct large areas of uniform color with a lot of splats if a few suffice. It is therefore recommended to initialize with 100k~300k splats and let the 3DGS default densification process increase the level of detail.

At this point, there are M key cameras, each with an image, depth map and mask, all of resolution $W \times H$. Turning all $M \times W \times H$ pixels into splats would result in millions of splats, so the pixels are subsampled with a factor S, resulting in a new resolution $\frac{W}{S} \times \frac{H}{S}$:

$$S = \frac{\sqrt{0.6MWH}}{300000} \quad (5)$$

The 0.6 means that it is estimated that about 60% of the mask will be true. Each pixel of every subsampled key image is, if the mask equals true, unprojected to its 3D position in world space and becomes a splat. The RGB color of the image is used, with the degree of the SH being 0. The opacity is set to 0.1, just like the 3DGS implementation. The scale of the splat is:

$$\text{scale} = \|\boldsymbol{P} - \boldsymbol{O}_{\text{key}}\| \frac{S}{2} \text{focal}_x \tag{6}$$

where $\boldsymbol{P}$ is the 3D position of the splat, $\boldsymbol{O}_{\text{key}}$ the position of its key camera, and focal_x the horizontal focal length in pixel units. This results in splats that just touch each other. Since the scaling is uniform, the rotation does not matter and is set to 0. The point cloud of the original Colmap SfM step is also converted to splats in the same way. The only difference with the original 3DGS approach is the scale, which in the proposed approach is calculated using Equation 6. Here, $\boldsymbol{O}_{\text{key}}$ becomes the position of the camera from which the SfM point $\boldsymbol{P}$ originates.

C. IMPLEMENTATION

1) GPU ACCELERATION

The proposed MVS pipeline is implemented in C++ and OpenGL, to benefit from the massively parallel nature of the GPU for 3D warping and texture processing. Writing the splats efficiently to a file is implemented using OpenGL's Transform Feedback logic. Section IV-C discusses the computational performance of the implementation in greater detail.

2) 3DGS TRAINING PARAMS

By default, 3DGS initializes its splat using Colmap's SfM sparse point cloud, meaning that the initial scene consists mainly of sparse, large, very low opacity splats. During the training process, the splats shrink slowly, become more opaque and after 600 iterations, densification kicks in. This behavior is not how the splats evolve during training when using our proposed initialization. Using our method, the scene starts of with dense, small, opaque splats in certain areas, and no splats in areas of the scene where large, textureless objects will sit. As the training progresses, the existing splats do not change all that much, except for the splats that quickly grow to fill the empty, uniformly-colored areas.

To get the best possible scene reconstruction using the method of this paper, some of the 3DGS training parameters need to be adapted:

- scaling_lr = 0.02 instead of 0.005
- densify from iteration 200 instead of 600
- lambda_dssim = 0.3 instead of 0.2

In other words, the growing of splats is encouraged by quadrupling the scale learning rate. These large splats need to be densified as fast as possible, so the densification is moved up 400 iterations earlier. Increasing the contribution of the SSIM loss from 20% to 30% also improves the subjective quality. Section IV-D2 contains an ablation study that looks into detail how these parameter changes affect the objective quality.

IV. RESULTS

All experiments were executed on a consumer-grade system with a i9-9900K processor and an RTX 2080 Ti GPU with 11 GB of VRAM.

A. DATASETS

The specifics of a light field dataset significantly impact the training process, visual quality and rendering performance of 3DGS. Important factors are the number of cameras, their resolution and placement, and the contents of the scene (fine geometry, texture, size, exposure uniformity, reflections, etc.). Therefore, we test our approach on a wide variety of datasets, summarized in Table 5. We selected 16 challenging datasets from multiple, widely used benchmarks/papers: DTU, LLFF, MipNerf360, Ommo, Shiny, Tanks & Temples (T&T), UCL. By testing our method on all these benchmarks, we can reason about its effectiveness in forward-facing (LLFF, UCL), circling (MipNerf360, T&T), and straight-line (Ommo) scenes, as well as those with complex reflections (Shiny). We also publish a novel static light field benchmark, consisting of 29 datasets, called *Ghent29*. The choice of scenes and positioning of the cameras were made to ensure a large variety. This is necessary to verify the generalizability of the proposed method. Traditionally challenging elements such as fine geometry, reflections, specular highlights, texture-less areas are abundantly present. Moreover, different camera setups are represented, for example planar configurations, circling around an object, walking along a line, and scanning large areas. Lastly, the extent of the camera setup is also varied, from small, densely sampled datasets to sparsely sampled paths of hundreds of meters in length.

B. QUALITATIVE EVALUATION

In this section, the visual quality of the proposed system is compared to three other state-of-the-art approaches. The three methods differ only in how they initialize the Gaussian splats before 3DGS begins its first training iteration:

1) *Colmap*: each point in Colmap's SfM point cloud is converted to a splat of uniform scaling, opacity 0.1, and SH degree 0. The scaling depends on the distance to the closest neighboring splat.
2) *MVSGaussian*: MVSGaussian [21] generates a point cloud similar to the previous method, but denser. The points to splat conversion is also the same.
3) *Ours*: the proposed method, which directly outputs Gaussian splats build from depth and consistency maps.

For each method, the default 3DGS training process is executed, for up to 1000 iterations. The goal of this paper is to have a high-quality scene reconstruction in the least amount of time possible, which is why this section focuses on the

TABLE 1. Quality metrics on test sets after 1000 training iterations, averaged per benchmark, for three methods of initializing the Gaussian splats: Colmap SfM, MVSGaussian (mvsgs) and our method. The best results per benchmark and metric are highlighted.

	PSNR↑			SSIM↑			LPIPS↓		
	colmap	mvsgs	**ours**	colmap	mvsgs	**ours**	colmap	mvsgs	**ours**
DTU	24.24	21.38	**24.38**	0.823	0.805	**0.861**	0.278	0.275	**0.223**
LLFF	24.99	24.84	**27.82**	0.852	0.860	**0.889**	0.224	0.189	**0.144**
MipNerf360	24.48	20.27	**25.46**	0.704	0.537	**0.777**	0.362	0.522	**0.252**
Ommo	25.61	22.91	**28.53**	0.792	0.703	**0.880**	0.338	0.416	**0.198**
Shiny	26.74	24.47	**27.89**	0.895	0.846	**0.919**	0.218	0.261	**0.172**
T&T	19.69	14.98	**20.35**	0.696	0.546	**0.732**	0.396	0.594	**0.334**
UCL	27.32	24.52	**28.46**	0.868	0.821	**0.909**	0.213	0.244	**0.124**
Ghent29	23.05	20.74	**24.34**	0.704	0.654	**0.779**	0.367	0.413	**0.257**

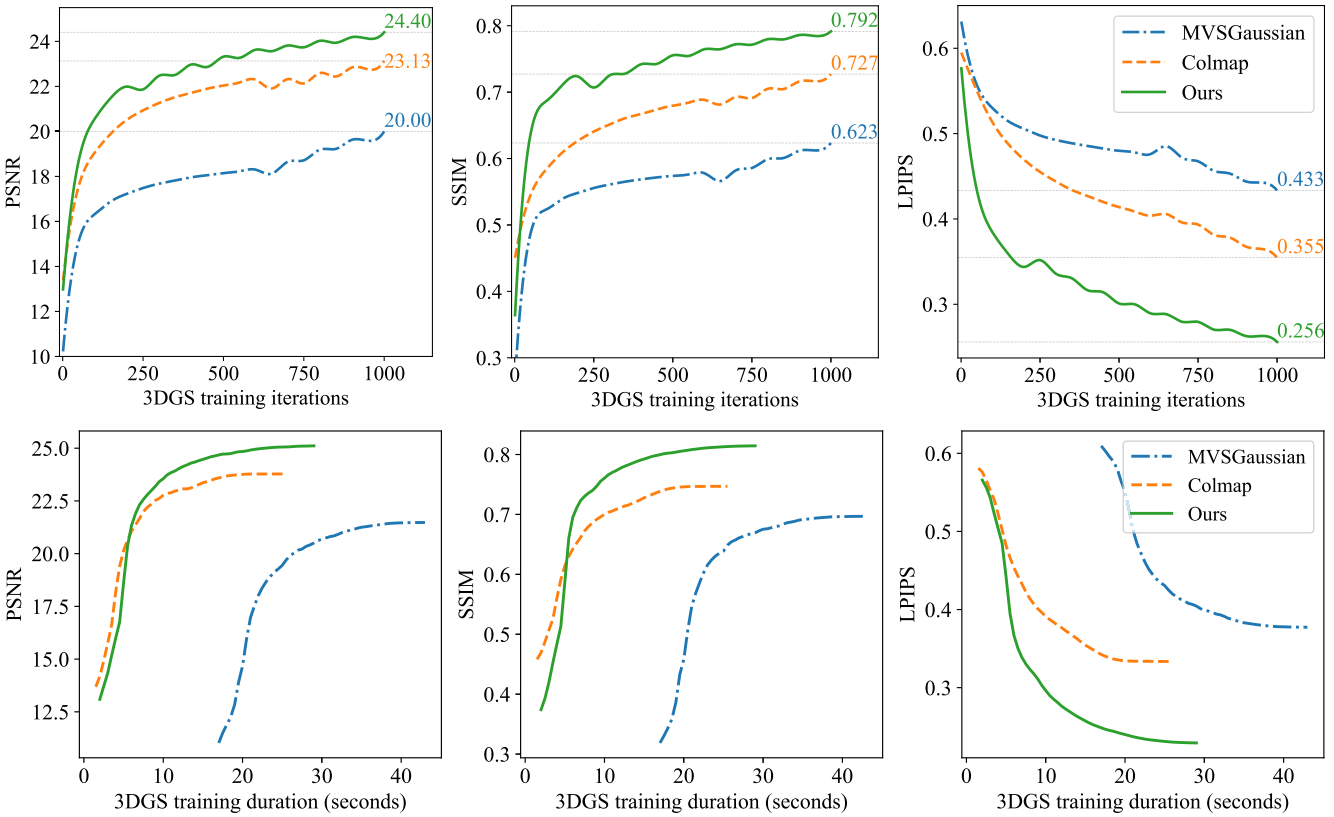

FIGURE 2. (top) The PSNR, SSIM (higher is better) and LPIPS (lower is better) averaged over every dataset's test set, to show the trend over 1000 3DGS training iterations. (bottom) The same metrics, but now the horizontal axis is the duration in seconds of each method, starting right after the Colmap SfM step. So these times include those in Table 2.

first 1000 iterations. Note that if 3DGS training is resumed, eventually all methods will reach a high quality, as shown in Section IV-D1. The objective quality is measured through PSNR, SSIM and LPIPS, of which the implementation provided by 3DGS [5] is used. The N images of each dataset are split into a train-test as follows: starting from the third image, every 8th image is moved from the training to the test set.

$$S_{\text{test}} = \{\boldsymbol{I}_i \,|\, i = 8k + 2\,, k \in \mathbb{N}\,, i < N\} \tag{7}$$

Note that the test set images are used during the Colmap SfM step, but not during MVSGaussian's or our proposed splat generation step. Our method also does not use the points in the SfM point cloud originating from the test set. Whenever PSNR, SSIM or LPIPS is mentioned in this paper, it was calculated solely on the test set.

The measured objective quality after 1000 training iterations, averaged per benchmark, can be found in Table 1. Additionally, the trajectory of the metrics across the first 1000 iterations, averaged over all datasets, are plotted in the top row of Figure 2. The bottom row shows the same data, but with the duration in seconds as the horizontal axis. Figure 3 shows some of the rendered images for each method.

Our method, due to a denser initialization of the Gaussian splats, is able to better reconstruct all the details in the scene. The main reason is that Colmap SfM initialization often lacks

FIGURE 3. A comparison of three splat initialization methods. The results shown are (closeup) renders after 1000 3DGS training iterations.

points and thus requires more training iterations to complete the reconstruction. This becomes the most noticeable in areas with less distinct features, or background areas, where SfM detects less SIFT feature (matches). MVSGaussian's

TABLE 2. The duration (in seconds), averaged per benchmark, of the splat initialization step, which sits between SfM and 3DGS training.

Splat initialization duration (s)		
	mvsgs	**ours**
DTU	18.59	1.62
LLFF	17.58	0.87
MipNerf360	18.23	1.78
Ommo	15.58	0.79
Shiny	16.46	0.84
T&T	18.09	1.44
UCL	16.20	0.54
Ghent29	16.52	1.26

visual quality varies a lot between datasets, and on average performs much worse than the other methods. The reason is twofold. Firstly, because it uses a neural network pretrained on (the whole dataset of) DTU, which consists of close-ups of objects, it does not generalize well to, for example, large-scale scenes. Secondly, only four key cameras are considered for depth estimation and point cloud fusion, so scenes with many more cameras are not initialized with enough splats. Increasing the number of key cameras would significantly increase the runtime of MVSGaussian, which is already high (see Table 2), which is against the goal of this paper. The objective metrics confirm the subjective findings. Our method consistently outperforms the others for all three metrics. On average, on the test set, our method improves PSNR by $+1.27$ dB, SSIM by $+0.065$ and LPIPS by -0.10 compared to the default 3DGS initialization scheme.

TABLE 3. The time (in seconds), averaged over every dataset's test set, it takes to reach the specified PSNR, SSIM and LPIPS. The initialization of splats using the Colmap SfM point cloud versus our proposed approach are compared. These results are derived from the bottom row of Figure 2.

Time (seconds) to reach			
	PSNR = 23.5 dB	SSIM = 0.73	LPIPS = 0.34
Colmap	16.2	15.4	17.6
Ours	9.8	7.5	7.0

Instead of looking at the difference in quality at a fixed iteration, it is also possible to set a predetermined quality, and see how long it takes to achieve this. Table 3 shows how long it takes for each technique to reach a fixed PSNR $= 23.5$ dB, SSIM $= 0.73$ and LPIPS $= 0.34$, when averaged over every dataset's test set. Our method of splat initialization results in shorter training times compared to using Colmap's SfM point cloud.

C. COMPUTATIONAL PERFORMANCE

The objective of this paper is to have a high-quality scene reconstruction in as little time as possible. The bottom row of Figure 2 shows that our method indeed produces, on average, a higher-quality result in less time than the default 3DGS implementation and MVSGaussian. In those plots, we can also see that our method of splat initialization makes it so that the 3DGS training loop starts slightly later (about one second) when compared to 3DGS. For MVSGaussian,

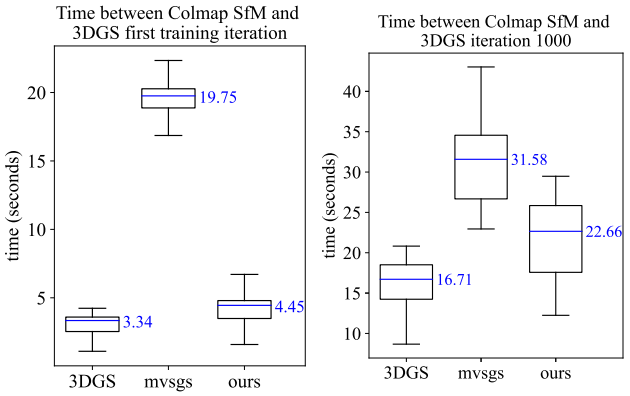

FIGURE 4. Boxplots of the time (in seconds, for each dataset) between the Colmap SfM step and the first training iteration of 3DGS. The median times are printed in blue.

the start of the training process is even delayed by more than ten seconds. This is because MVSGaussian and our method add an additional processing step between Colmap SfM and 3DGS training. This delays the moment at which training can begin, making it so that the Colmap SfM point cloud initialization method has a head start in reaching the final scene reconstruction. Table 2 shows the duration, averaged per benchmark, of the splat initialization step, which takes place between the SfM and 3DGS training. So for the Colmap initialization method, this takes zero seconds since the SfM point cloud is used. For MVSGaussian, it is the time to estimate the depth maps, fuse them and output a point cloud. Our method is optimized for execution speed and manages to output the initial Gaussian splats in 1.2 seconds on average. In comparison, MVSGaussian takes 16.7 seconds on average to output its dense point cloud.

Figure 4 shows the time (in seconds) between Colmap SfM and the first training iteration. This time interval consists of the durations mentioned in the previous paragraph (see Table 2) and 3DGS' setup code to load in and parse the libraries, dataset and splat initialization. We can conclude that loading in MVS-Splatting's initialization takes the same time as that of 3DGS and MVSGaussian. The right plot shows the *total duration*, meaning the time between Colmap SfM and the last training iteration. The median total duration is 5.9 seconds longer for our method than 3DGS default splat initialization, and 14.9 seconds longer for MVSGaussian. The reason is twofold. Firstly, MVSGaussian and our method introduce a much higher splat count at the first iteration, slowing down the training process. Our experimental setup is bottle necked by its limited VRAM. If the default Colmap SfM initialization method's training process were to continue, that same bottleneck situation of a high number of splats and long training times would occur. Second and most importantly, our method executes the densification step, which is the slowest part of the 3DGS pipeline, an additional four times compared to the other two methods.

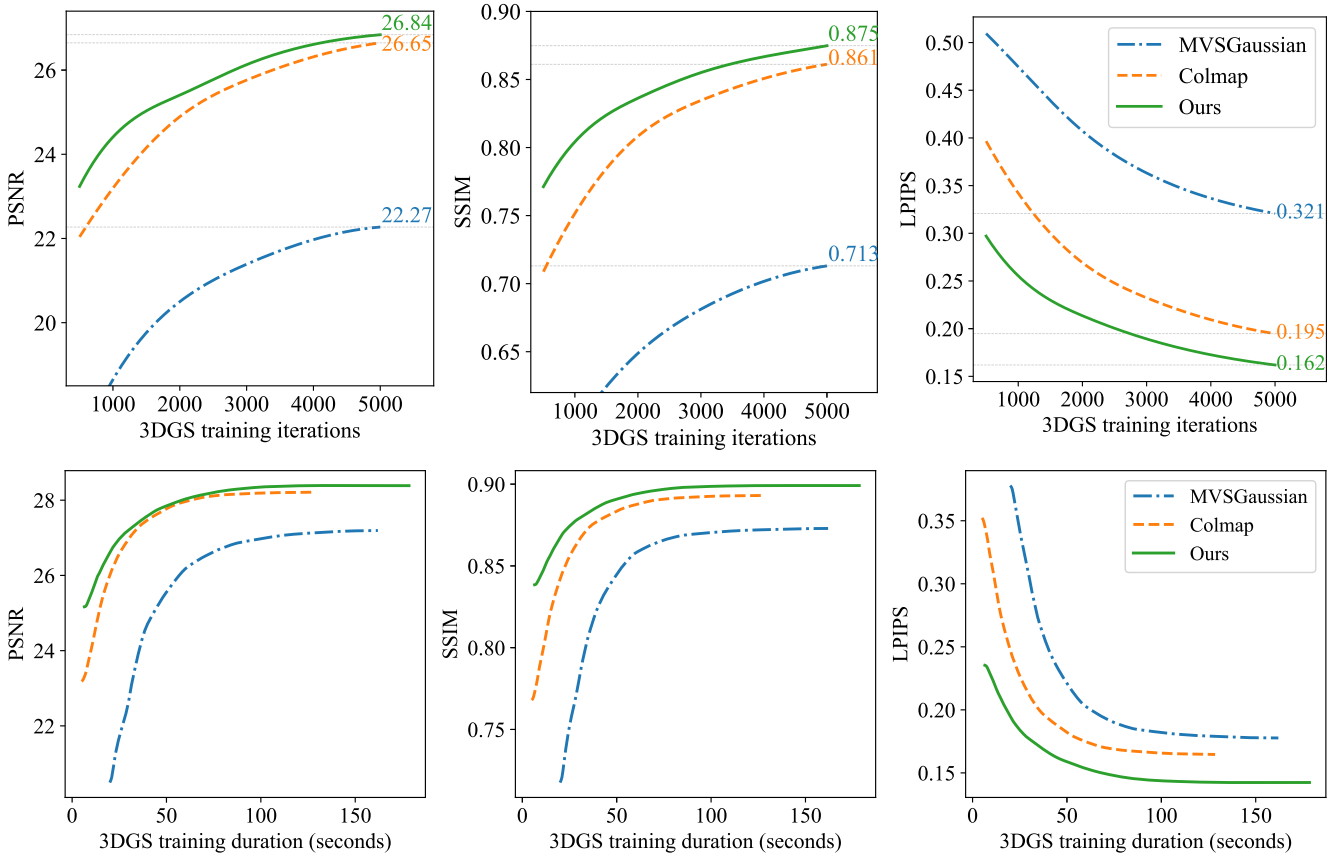

FIGURE 5. (top) The PSNR, SSIM and LPIPS averaged over every dataset's test set, to show the trend over 5000 3DGS training iterations. (bottom) The same metrics, but now the horizontal axis is the duration in seconds of each method, starting right after the Colmap SfM step, just like Figure 2.

D. MISC. EXPERIMENTS AND ABLATION STUDY

1) TRAINING 5K ITERATIONS

In the previous Section, each dataset was trained for up to 1000 iterations. It is possible to continue training though, although a limited VRAM will bottleneck the computational performance. For consistency, we turn off 3DGS' *opacity reset* option at iteration 3000. Figure 5 shows the PSNR, SSIM and LPIPS averaged over all datasets' test set (except those of Ghent29), when training for up to 5000 iterations. Again, our technique consistently outperforms the others in all three metrics.

2) SAME TRAINING PARAMETERS

As explained in Section III-C, the proposed method, due to outputting Gaussian splats with a predefined scale, needs slightly different training parameters compared to the default 3DGS approach. In this ablation study, we look at the visual quality when the same parameters as the Colmap SfM and MVSGaussian method are used:

- scaling_lr = 0.005
- densify_from_iter = 500
- lambda_dssim = 0.2

Note that the 3DGS implementation interprets this as starting the densification from iteration 600, so 100 after the 500 specified above. The objective quality metrics over the test set after 1000 training iterations can be found in Table 4. Our method performs slightly worse with these parameters compared to back in Table 1, which used the recommended parameters. However, the SSIM and LPIPS remain better than those for default 3DGS splat initialization. The PSNR of our approach is worse than 3DGS for benchmarks DTU, T&T and UCL. This is because for our method, the splats could sometimes not grow fast enough to cover the black background adequately, which lowers the PSNR. In all other areas, our method is able to more accurately reconstruct the fine details of the scene, resulting in an improved SSIM and LPIPS.

V. DISCUSSION

Our method relies on MVS, which typically suffers from textureless areas, reflections etc., but these challenges are mostly overcome by the consistency check. However, our method performs poorly for datasets where the exposure is inconsistent between images. This is mainly the case in the Tanks&Temples dataset. It is therefore advised to preprocess the images to a uniform exposure. MVS also struggles with estimating the depth when there is no overlap between stereo pairs. For example, near the outer edges of a forward-facing dataset, there is often no camera overlap. This will

TABLE 4. Ablation study with the same parameters for all methods. Quality metrics on test sets after 1000 training iterations, averaged per benchmark.

	PSNR↑		SSIM↑		LPIPS↓	
	colmap	**ours**	colmap	**ours**	colmap	**ours**
DTU	**24.24**	22.03	0.823	**0.833**	0.278	**0.255**
LLFF	24.99	**26.57**	0.852	**0.881**	0.224	**0.164**
MipNerf360	24.48	**24.65**	0.704	**0.749**	0.362	**0.288**
Ommo	25.61	**27.43**	0.792	**0.860**	0.338	**0.233**
Shiny	26.74	**26.83**	0.895	**0.905**	0.218	**0.186**
T&T	**19.69**	19.62	0.696	**0.720**	0.396	**0.352**
UCL	**27.32**	26.94	0.868	**0.879**	0.213	**0.176**
Ghent29	23.05	**23.42**	0.704	**0.745**	0.367	**0.304**

result in floaters that also will not be masked off during the consistency check, due to the lack of overlap. Depth-guided training techniques [24], [25] can alleviate this problem.

This paper focuses on perspective cameras, since 3DGS is limited to this. The proposed MVS method, however, can be extended to also work for 360° cameras, or fisheye lenses. More experiments would need to be performed to evaluate the quality of such a system though. Another avenue for future work is to apply the proposed method in the context of dynamic content, and how the four instead of three dimensions of the Gaussian Splats influence the reconstruction quality after.

VI. CONCLUSION

The iterative nature of the 3DGS training process is its main strength, because it can optimize the splats in an *occlusion-aware* way. At the same time, the iterative process is also its greatest drawback: by definition, the training has to take a certain number of iterations before the scene can reach its final high-quality reconstruction. In this paper, we have shown that by initializing the Gaussian splats correctly, we can skip ahead several hundred iterations in the training process. The key insight is that the "easy" parts of the scene can quickly be reconstructed using MVS depth estimation. With easy, we mean smooth, textured areas with not too much specularity. For the remaining "hard" parts (textureless areas, etc.), we can rely on the iterative, occlusion-aware nature of 3DGS. With this new configuration, the learning rate can even be increased to shorten the training process further.

Our proposed method would mainly be useful in scenarios where a lot of static content needs to be visualized using light field rendering. Any speedup in the training process will lead to noticeable savings at large-scale. In future work, the execution speed of the MVS implementation could be further optimized, since there is still a lot of room for improvement. It can also be worthwhile to initialize the SH to a higher degree, since this is still a slow process for 3DGS in general.

APPENDIX
DATASETS

See Table 5.

TABLE 5. Overview of datasets used in this work.

Benchmark	Dataset name	#Img	Resolution
DTU [17]	scan1	49	1440 × 1079
	scan6	49	1440 × 1079
LLFF [26]	orchids	25	1589 × 1191
	room	41	1426 × 1069
	santarex	20	1554 × 1164
MipNerf360 [27]	bicycle	194	988 × 656
	garden	185	1037 × 671
Ommo [28]	05_opera_house	65	1313 × 679
	10a_new_york	41	1272 × 715
	10b_new_york	52	1244 × 699
	15_monastery	103	959 × 539
Shiny [29]	glass	66	1248 × 934
	shiny_cd	75	977 × 541
T&T [30]	train	301	978 × 543
	truck	251	767 × 431
UCL [31]	hugo	24	1023 × 679
Ghent29	01_large_pond	141	844 × 633
	02_classroom	37	1595 × 1196
	03_botanics_entrance	35	1600 × 1199
	04_botanics_treestump	32	1597 × 1197
	05_botanics_dead_tree	41	1554 × 1165
	06_botanics_pond	35	1598 × 1198
	07_greenhouse_plant	53	1377 × 1031
	08_greenhouse_walk	52	1379 × 1034
	09_greenhouse_glass	37	1603 × 1201
	10_greenhouse_corner	54	1358 × 1018
	11_greenhouse_middle	70	1198 × 898
	12_greenhouse_lamp	43	1527 × 1145
	13_cactus_half_circle	63	1260 × 945
	14_cactus_table	38	1594 × 1195
	15_cactus_pathway	37	1597 × 1197
	16_statue_claus	60	1290 × 967
	17_statue_park	42	1536 × 1152
	18_statue_pond_360	108	965 × 723
	19_statue_three	102	996 × 747
	20_mansion_chandelier	79	1124 × 843
	21_mansion_candles	22	1592 × 1193
	22_mansion_fireplace	44	1499 × 1124
	23_mansion_bust	33	1588 × 1191
	24_mansion_ballroom	67	1221 × 916
	25_mansion_table	51	1391 × 1043
	26_mansion_tearoom	70	1196 × 897
	27_mansion_desk	44	1500 × 1125
	28_church_statue	30	1593 × 1194
	29_church_relief	43	1514 × 1135

REFERENCES

[1] P. Barra, M. Giammetti, A. Tortora, and A. D. Greca, "Redefining interaction in a digital twin laboratory with mixed reality," in *Artificial Intelligence in HCI*. Cham, Switzerland: Springer, 2024, pp. 295–307.

[2] D. Bonatto, S. Fachada, S. Rogge, A. Munteanu, and G. Lafruit, "Real-time depth video-based rendering for 6-DoF HMD navigation and light field displays," *IEEE Access*, vol. 9, pp. 146868–146887, 2021.

[3] J. Artois, M. Courteaux, G. Van Wallendael, and P. Lambert, "OpenDIBR: Open real-time depth-image-based renderer of light field videos for VR," *Multimedia Tools Appl.*, vol. 83, no. 9, pp. 25797–25815, Aug. 2023.

[4] 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," in *Proc. ECCV*, 2020, pp. 405–421.

[5] B. Kerbl, G. Kopanas, T. Leimkuehler, and G. Drettakis, "3D Gaussian splatting for real-time radiance field rendering," *ACM Trans. Graph.*, vol. 42, no. 4, pp. 1–14, Jul. 2023. [Online]. Available: https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/

[6] 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 Trans. Graph.*, vol. 43, no. 4, pp. 1–15, Jul. 2024.

[7] A. Dalal, D. Hagen, K. G. Robbersmyr, and K. M. Knausgård, "Gaussian splatting: 3D reconstruction and novel view synthesis: A review," *IEEE Access*, vol. 12, pp. 96797–96820, 2024.
[8] P. Moulon, P. Monasse, and R. Marlet, "Global fusion of relative motions for robust, accurate and scalable structure from motion," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 3248–3255.
[9] C. Griwodz, S. Gasparini, L. Calvet, P. Gurdjos, F. Castan, B. Maujean, G. De Lillo, and Y. Lanthony, "AliceVision meshroom: An open-source 3D reconstruction pipeline," in *Proc. 12th ACM Multimedia Syst. Conf.*, Jun. 2021, pp. 241–247.
[10] J. L. Schönberger and J.-M. Frahm, "Structure-from-motion revisited," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 4104–4113.
[11] Y. Cai, M. Cao, L. Li, and X. Liu, "An end-to-end approach to reconstructing 3D model from image set," *IEEE Access*, vol. 8, pp. 193268–193284, 2020.
[12] H. Hirschmuller, "Accurate and efficient stereo processing by semi-global matching and mutual information," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR05)*, vol. 2, Jun. 2005, pp. 807–814.
[13] W. Feng, S. K. Ali, and R. O. K. W. Rahmat, "Efficient 3D reconstruction through enhanced PatchMatch techniques for accelerated point cloud generation," *IEEE Access*, vol. 12, pp. 144588–144598, 2024.
[14] S. Liu, M. Li, X. Zhang, S. Liu, Z. Li, J. Liu, and T. Mao, "Image-based rendering for large-scale outdoor scenes with fusion of monocular and multi-view stereo depth," *IEEE Access*, vol. 8, pp. 117551–117565, 2020.
[15] K. Zhang, M. Liu, J. Zhang, and Z. Dong, "PA-MVSNet: Sparse-to-dense multi-view stereo with pyramid attention," *IEEE Access*, vol. 9, pp. 27908–27915, 2021.
[16] R. Weilharter and F. Fraundorfer, "HighRes-MVSNet: A fast multi-view stereo network for dense 3D reconstruction from high-resolution images," *IEEE Access*, vol. 9, pp. 11306–11315, 2021.
[17] R. Jensen, A. Dahl, G. Vogiatzis, E. Tola, and H. Aanaes, "Large scale multi-view stereopsis evaluation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 406–413.
[18] Y. Yao, Z. Luo, S. Li, J. Zhang, Y. Ren, L. Zhou, T. Fang, and L. Quan, "BlendedMVS: A large-scale dataset for generalized multi-view stereo networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 1787–1796.
[19] B. Fei, J. Xu, R. Zhang, Q. Zhou, W. Yang, and Y. He, "3D Gaussian splatting as new era: A survey," *IEEE Trans. Vis. Comput. Graphics*, vol. 31, no. 8, pp. 4429–4449, Aug. 2025.
[20] S. Seibt, T. V. S.-L. Chang, B. V. R. Lipinski, and M. E. Latoschik, "Dense 3D Gaussian splatting initialization for sparse image data," in *Proc. Eurographics Posters*, 2024, pp. 5–6.
[21] T. Liu, G. Wang, S. Hu, L. Shen, X. Ye, Y. Zang, Z. Cao, W. Li, and Z. Liu, "MVSGaussian: Fast generalizable Gaussian splatting reconstruction from multi-view stereo," 2024, *arXiv:2405.12218*.
[22] R. T. Collins, "A space-sweep approach to true multi-image matching," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 1996, pp. 358–363.
[23] D. Gallup, J.-M. Frahm, P. Mordohai, Q. Yang, and M. Pollefeys, "Real-time plane-sweeping stereo with multiple sweeping directions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2007, pp. 1–8.
[24] J. Chung, J. Oh, and K. M. Lee, "Depth-regularized optimization for 3D Gaussian splatting in few-shot images," 2023, *arXiv:2311.13398*.
[25] J. Li, J. Zhang, X. Bai, J. Zheng, X. Ning, J. Zhou, and L. Gu, "DNGaussian: Optimizing sparse-view 3D Gaussian radiance fields with global-local depth normalization," 2024, *arXiv:2403.06912*.
[26] B. Mildenhall, P. P. Srinivasan, R. Ortiz-Cayon, N. K. Kalantari, R. Ramamoorthi, R. Ng, and A. Kar, "Local light field fusion: Practical view synthesis with prescriptive sampling guidelines," *ACM Trans. Graph.*, vol. 38, no. 4, pp. 1–14, Aug. 2019.
[27] J. T. Barron, B. Mildenhall, D. Verbin, P. P. Srinivasan, and P. Hedman, "Mip-NeRF 360: Unbounded anti-aliased neural radiance fields," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 5460–5469.
[28] C. Lu, F. Yin, X. Chen, T. Chen, G. YU, and J. Fan, "A large-scale outdoor multi-modal dataset and benchmark for novel view synthesis and implicit scene reconstruction," 2023, *arXiv:2301.06782*.
[29] S. Wizadwongsa, P. Phongthawee, J. Yenphraphai, and S. Suwajanakorn, "NeX: Real-time view synthesis with neural basis expansion," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 8530–8539.
[30] A. Knapitsch, J. Park, Q.-Y. Zhou, and V. Koltun, "Tanks and temples: Benchmarking large-scale scene reconstruction," *ACM Trans. Graph.*, vol. 36, no. 4, pp. 1–13, Aug. 2017.
[31] P. Hedman, J. Philip, T. Price, J.-M. Frahm, G. Drettakis, and G. Brostow, "Deep blending for free-viewpoint image-based rendering," *ACM Trans. Graph.*, vol. 37, no. 6, pp. 1–15, Dec. 2018, doi: 10.1145/3272127.3275084.

JULIE ARTOIS received the M.Sc. degree in computer science engineering from Ghent University, Belgium, in 2020. She is currently pursuing the Ph.D. degree with IDLab, imec, Ghent University.

Her main research interests include image processing, light field rendering, virtual reality, and 3D graphics.

PETER LAMBERT (Senior Member, IEEE) received the M.Sc. degree in science (mathematics), the M.Sc. degree in applied informatics, and the Ph.D. degree in computer science from Ghent University, Belgium, in 2001, 2002, and 2007, respectively.

In 2009, he was a Technology Developer, which he combined with a part-time Assistant Professor with IDLab, imec, Ghent University, from 2010 to 2013, where he is currently a full-time Associate Professor. His research interests include (mobile) multimedia applications, multimedia coding and adaptation technologies, and 3D graphics.

GLENN VAN WALLENDAEL (Member, IEEE) received the master's and Ph.D. degrees in computer science engineering from Ghent University, Belgium, in 2008 and 2013, respectively.

After his Doctoral and Post-Doctoral work financed by the prestigious Research Foundation Flanders (FWO). He is currently a Professor with IDLab, imec, Ghent University, in 2019. His research interests include video compression, immersive visual modalities, video security, visual quality of experience, and standardization.