BRUM: Robust 3D Vehicle Reconstruction from 360° Sparse Images

Davide Di Nucci[1], Matteo Tomei[2], Guido Borghi[1], Luca Ciuffreda[2], Roberto Vezzani[1], Rita Cucchiara[1]
[1]University of Modena and Reggio Emilia, [2]Prometeia

***Abstract*—Accurate 3D reconstruction of vehicles is vital for applications such as vehicle inspection, predictive maintenance, and urban planning. Existing methods like Neural Radiance Fields and Gaussian Splatting have shown impressive results but remain limited by their reliance on dense input views, which hinders real-world applicability. This paper addresses the challenge of reconstructing vehicles from sparse-view inputs, leveraging depth maps and a robust pose estimation architecture to synthesize novel views and augment training data. Specifically, we enhance Gaussian Splatting by integrating a selective photometric loss, applied only to high-confidence pixels, and replacing standard Structure-from-Motion pipelines with the DUSt3R architecture to improve camera pose estimation. Furthermore, we present a novel dataset featuring both synthetic and real-world public transportation vehicles, enabling extensive evaluation of our approach. Experimental results demonstrate state-of-the-art performance across multiple benchmarks, showcasing the method's ability to achieve high-quality reconstructions even under constrained input conditions. Code and data will be made publicly available[1].**

I. Introduction

Novel view synthesis and 3D scene representation have been driven by techniques such as Neural Radiance Fields (NeRF) [1] and Gaussian splatting (GS) [2] in the last few years. Both are capable of generating photorealistic images of complex scenes from input views with their corresponding camera poses: NeRF leverages an implicit representation through a neural network which models color and density of the scene as a function of the spatial coordinates and viewing directions; GS employs an explicit representation of the scene using a set of 3D Gaussian primitives with associated attributes (*e.g.*, color, opacity).

While the focus of these methods has often been on general (both synthetic and real-world) scenes or human-centered tasks, their application to specific object classes or types of scenes – *e.g.*, the vehicle domain – is still under-investigated. Accurate 3D vehicle reconstruction is essential for several critical applications. One primary motivation is automated *vehicle inspection* [3], [4], where precise 3D models enable detailed analysis of structural integrity, surface condition, and potential defects. In addition, 3D reconstruction facilitates tracking the status of vehicles over time, providing a digital record of wear, modifications, or damage, and supporting predictive maintenance. Autonomous driving is increasingly benefiting from novel view synthesis techniques, too [5], [6].

Beyond individual monitoring, these models can also enhance fleet management systems by providing actionable insights into vehicle health and optimizing operations. Building on these advantages, the public transportation sector stands to benefit significantly from accurate reconstruction. Furthermore, in the context of smart cities, 3D models of public transportation vehicles can contribute to broader initiatives such as traffic management, urban planning, and environmental monitoring [7]–[9].

In practical scenarios, such as tracking a vehicle's condition over time (*e.g.*, daily inspections for buses), capturing images quickly and consistently is crucial. While video recordings enable rapid data collection, they are often constrained by significant storage and processing requirements. Human-operated data collection could be significantly faster and more reliable if it required only a predetermined set of sparse images. Fixed camera setups offer a cost-effective alternative for image acquisition but inherently limit the number of available views to the number of cameras deployed.

In this work, we tackle the challenge of 3D vehicle reconstruction under the constraint of limited sparse views. Specifically, starting with a handful of scene captures and their associated camera parameters, we leverage ground truth or estimated depth maps to project view-dependent point clouds into synthetic camera poses. These new poses are generated by systematically rotating and translating the original camera positions within a constrained range, effectively augmenting the available views for downstream Gaussian splatting training. Importantly, for these views, the photometric loss is applied exclusively to pixels with high-confidence re-projection.

Camera position estimation for real-world scenes is traditionally performed using standard structure-from-motion (S*f*M) pipelines, such as COLMAP [10]. Moreover, these methods often struggle or fail to converge when provided with only a few images or when image overlap is minimal [4], [11]. To address this limitation, we leverage the recently proposed DUSt3R architecture [12] for estimating both camera poses and an initial point cloud.

We show that our proposed method achieves results comparable or even better than the state of the art in forward-facing 360° vehicle scenes from the Carpatch [3] and KRONC [4] datasets, using as few as 4-8 images, without adding considerable computation time. To assess the effectiveness of our approach in large public transportation vehicle scenes, we also introduce BRUM-dataset, of both synthetic and realistic bus instances, highlighting promising results on it.

To sum up, our key contributions can be outlined as follows:

- We propose to enhance Gaussian splatting robustness in sparse-view forward-facing 360° vehicle scenes. The

[1]https://aimagelab.ing.unimore.it/go/brum

proposed BRUM synthesizes novel images from coarse point clouds, effectively augmenting the input data.
- We adopt DUSt3R as a replacement for the standard COLMAP and incorporate a masking strategy to exclude low-confidence pixels from the loss computation.
- We present BRUM-dataset featuring 6 synthetic and 6 real public transportation vehicles. Our method achieves state-of-the-art performance on this dataset and other standard car inspection benchmarks.

II. Related Work

Novel View Synthesis. Novel view synthesis aims to generate photorealistic renderings of a 3D scene from unseen viewpoints. In recent years, this challenge has been predominantly addressed by NeRF-like approaches [1], with various works proposing solutions to overcome the limitations of the original formulation. For faster training, [13] proposed to encode the 3D space using a multiresolution hash table, followed by a small MLP. Plenoxels [14] replaces neural networks with a voxel grid that directly stores density and spherical harmonic coefficients at each voxel, optimized using the standard MSE reconstruction loss. Other works focus on removing the dependency on S*f*M preprocessing: [15], [16] use bundle adjustment to refine camera parameters starting from noisy ones, [17] optimizes predicted depth maps distortion parameters and camera poses together with NeRF, [18] introduces a feature synthesizer to render a dense feature map based on a coarse absolute camera pose predicted by an absolute pose regressor, minimizing the error between this feature map and the output of a pre-trained feature extractor. Recently, Gaussian splatting [2] has gained traction as a promising alternative to NeRF, providing advantages in training and rendering speed without compromising image quality. To address its shortcomings, the research community has undertaken various efforts. For instance, several studies have focused on reducing the storage footprint and memory requirements for deployment on edge devices, by compressing the Gaussian representation [19]–[22]. Other research has explored scene segmentation in 3D to enable editing capabilities [23]–[25]. Moreover, text-to-3D scene generation is rapidly emerging as a prominent task, combining 3D Gaussian splatting with the generative power of 2D diffusion models [26]–[29].

These methods highlight the capabilities of novel view synthesis techniques in efficiently reconstructing 3D scenes. However, they typically require capturing tens or hundreds of images to achieve accurate reconstructions, which can be both challenging and costly. Our work focuses on addressing this limitation by targeting environments with limited input images.

Few-shot Novel View Synthesis. In literature, many studies have attempted to enhance the robustness of NeRF and Gaussian splatting under sparse-view settings. For instance, [30] introduces a method that leverages a fully-convolutional, pre-trained image encoder to predict a feature grid from the input image. The grid is then used to compute color and density at given spatial locations using NeRF. Similarly, IBRNet [31] proposes a ray-based transformer to aggregate information from neighboring views, enabling accurate predictions of color and density for novel target views. Other approaches focus on leveraging multi-view correspondences and geometric constraints among training views to improve reconstruction under sparse input conditions [11], [32], [33]. Depth supervision, derived from S*f*M point clouds, monocular depth estimation, or depth completion models, has also been explored to enhance reconstruction quality [34], [35]. The sparse-view problem similarly affects Gaussian splatting, prompting the development of several solutions. SplatFields [36] identifies overfitting to training views due to low spatial autocorrelation between splats and introduces a neural framework to promote feature sharing among nearby primitives. CoherentGS [37] employs monocular depth to initialize 3D Gaussians and refines them using single-view and multi-view regularization based on depth and optical flow constraints. Other methods utilize robust pre-trained image matchers to incorporate multi-view matching priors [38], [39].

Unlike existing methods that often rely on sophisticated architectures with considerable training or inference time overhead, BRUM involves minimal additional complexity. Our work focuses on 360° forward-facing vehicle scenes with sparse views, a scenario typically explored in synthetic settings but rarely addressed in real-world environments.

III. Method

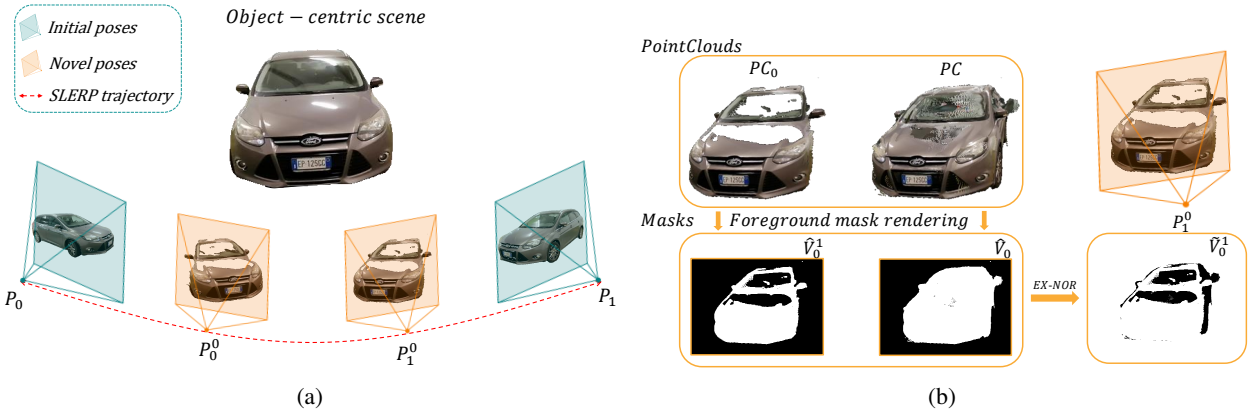
A. Augmenting the available training views

The input to our algorithm is a set of N sparse images, denoted as $\mathcal{I} = \{I_i\}_{i=1}^{N}$, capturing an object-centric scene. We assume depth maps for each input image $\mathcal{D} = \{D_i\}_{i=1}^{N}$ being available, together with corresponding extrinsic camera parameters $\mathcal{P} = \{P_i\}_{i=1}^{N}$, with $P_i = [R_i, \mathbf{t}_i]$, where $R_i \in SO(3)$ and $\mathbf{t}_i \in \mathbb{R}^3$ are the rotation matrix and translation vector of camera i, respectively, relative to a common world coordinate system. We also suppose known camera intrinsic parameters, shared among all views.

Sampling novel poses. For the i^{th} camera, we propose to generate M different novel poses starting from its parameters, $P_i = [R_i, \mathbf{t}_i]$, by applying rotations and translations within a limited range. To facilitate this, we first identify the closest camera, indexed as k, among the other $N-1$ cameras, characterized by its parameters $P_k = [R_k, \mathbf{t}_k]$. The pair of cameras P_i, P_k defines the starting and ending points for interpolation. To smoothly transition between these two poses, we adopt the Spherical Linear Interpolation (SLERP) [40] algorithm. SLERP interpolates between two unit quaternions along the shortest geodesic path on the unit sphere in $\mathbb{R}^4$, ensuring constant-speed rotation transitions. This approach maintains the object-centric nature of the scene while enabling precise and smooth interpolation for generating novel views.

P_i and P_k are first both converted to quaternions $\mathbf{q}_i$ and $\mathbf{q}_k$, then SLERP computes the intermediate quaternion $\mathbf{q}_h$ for interpolation parameter $h \in [0, 1]$ as follows:

$$\mathbf{q}_h = \frac{\sin((1-h)\theta)}{\sin\theta}\mathbf{q}_i + \frac{\sin(h\theta)}{\sin\theta}\mathbf{q}_k, \tag{1}$$



(a) (b)

Fig. 1: (a) illustrates an object-centric scene, where ground-truth camera poses are augmented along a SLERP (Sec. III-A) trajectory to generate novel poses. (b) shows how to obtain the final foreground mask $\tilde{V}_0^1$, starting from point clouds and intermediate masks $\hat{V}_0^1$ and $\hat{V}_0$.

where θ is the angle subtended by the arc between $\mathbf{q}_i$ and $\mathbf{q}_k$. Finally, $\mathbf{q}_h$ is converted back to $[R_h, \mathbf{t}_h]$. In practice, instead of relying solely on the nearest camera, we independently apply SLERP between the i^{th} camera and its two closest cameras. This enables interpolation along two distinct arcs, enhancing the diversity of the generated poses. The number of generated poses can be controlled by sampling various values of h, while constraining h within a specific range of values allows for limiting the deviation from the i^{th} camera. This results in a total of M new poses $\hat{\mathcal{P}}_i = \{\hat{P}_i^j\}_{j=1}^M$ sampled for camera i.

Generating novel views. Starting from poses $\hat{\mathcal{P}}_i$, by leveraging the RGB information in I_i and the depth D_i, we aim to produce synthetic novel views of the scene. Specifically, we first back-project pixels from I_i to 3D using the depth map D_i, obtaining a 3D point cloud PC_i:

$$PC_i = \pi_i^{-1}(I_i, D_i), \tag{2}$$

where π_i^{-1} represents the back-projection from the 2D image plane defined by P_i to the common 3D world's reference system. The point cloud PC_i is then warped to the novel views defined by camera poses $\hat{\mathcal{P}}_i$. Given a synthetic camera pose $\hat{P}_i^j \in \hat{\mathcal{P}}_i$, we project PC_i to its image plane as follows:

$$\hat{I}_i^j = \pi_j(PC_i), \tag{3}$$

where π_j is the projection from the common 3D world's reference system to the 2D image plane defined by $\hat{P}_i^j$. In practice, while we use the naïve back-projection for π^{-1} as a one-to-one mapping from 2D to 3D, we adopt the solution proposed by [41] for projection π. Specifically, a 3D point p from PC_i is splatted onto a 2D disk with radius r and center p_c, and its influence on a pixel u in $\hat{I}_i^j$ is inversely proportional to the 2D Euclidean distance between u and the disk's center p_c:

$$w(p,u) = \begin{cases} 0 & \text{if } \|p_c - u\|_2 > r \\ 1 - \frac{\|p_c - u\|}{r^2} & \text{otherwise,} \end{cases} \tag{4}$$

where $w(p,u)$ represents a weight quantifying how much the 3D point p affects pixel u. As in [41], the projected points are then stored in a z-buffer, sorted by their distance from the new camera pose $\hat{P}_i^j$ and only the K closest points are retained. Finally, alpha over-compositing is adopted for points accumulation. The weighting scheme helps ensure accurate novel views, particularly in cases of overlapping or occluded points, and guarantees smoother renderings.

Binary foreground masks $\hat{\mathcal{V}} = \{\hat{V}_i^j\}_{j=1}^M$ are also gathered, identifying the pixels that have been successfully projected from PC_i to $\hat{I}_i^j$.

B. Training objective

We can now exploit the original N images $\mathcal{I} = \{I_i\}_{i=1}^N$ together with the novel $N \times M$ images $\hat{\mathcal{I}} = \{\{\hat{I}_i^j\}_{j=1}^M\}_{i=1}^N$ generated from the original ones, for downstream Gaussian splatting training. We recall the original Gaussian splatting objective in the following equation:

$$\mathcal{L} = (1-\lambda)\mathcal{L}_1(\tilde{I}, I) + \lambda\mathcal{L}_{SSIM}(\tilde{I}, I), \tag{5}$$

which is a combination of the L1 and SSIM [42] losses between rendered and ground truth images, $\tilde{I}$ and I, respectively.

First, for the 3D Gaussians optimization, we exclude pixels from the generated images that are not accurately projected from the 3D point cloud, as indicated by the foreground masks. However, in object-centric scenes - where objects are segmented, and the background is removed - simply masking out these pixels in the loss can inadvertently exclude background Gaussians from optimization. This happens because background points are absent from the 3D point cloud and are therefore not reported in the foreground masks. As a result, background Gaussians would remain unoptimized, leading to poor 3D reconstruction where the background is only initialized but never refined. To address this, we distinguish between background pixels and those that are actually not reprojected. For each foreground mask in $\hat{\mathcal{V}} = \{\{\hat{V}_i^j\}_{j=1}^M\}_{i=1}^N$, we compute its *exclusive nor* with the foreground mask obtained by rendering the complete point cloud $PC = \bigcup_{i=1}^N PC_i$ from the same camera pose:

$$\tilde{V}_i^j = 1 - (\hat{V}_i^j \oplus \hat{V}_i), \tag{6}$$

where $\hat{V}_i$ is the foreground of PC rendered from $\hat{P}_i^j$ and $\oplus$ is the XOR operator. We consider only pixels retained in $\tilde{V}_i^j$ in the final loss. This approach retains background pixels while excluding those not reprojected from PC_i, ensuring a more accurate optimization. More details in Figure 1.

Second, we associate a *reliability* measure with each pixel in the generated images. Beyond masking-out pixels that are incorrectly projected from 3D, we apply weights to the remaining pixels using the influence values computed in Eq. 4. Intuitively, a pixel largely affected by multiple 3D points is considered more reliable, and its weight in the loss function increases. Formally, for pixel u in $\hat{I}_i^j$, its loss weight becomes:

$$w_K(u) = \sum_{k=1}^{K} w(p_k, u), \tag{7}$$

where p_k is the k^{th} point from point cloud PC_i among the K used for alpha over-compositing, as mentioned in Sec. III-A. This leads to weights $\hat{W}_i^j$ for image $\hat{I}_i^j$, after computing the above equation for all the pixels. $\hat{W}_i^j$ is further normalized as:

$$\tilde{W}_i^j = \frac{\hat{W}_i^j - \min(\hat{W}_i^j)}{\max(\hat{W}_i^j) - \min(\hat{W}_i^j)} \tag{8}$$

Finally, as the SSIM loss evaluates the structural similarity at the image level rather than on a per-pixel basis, and given that generated images are often less reliable and may lack consistent overall structure, we exclude the SSIM loss from the optimization process for generated images.

Our overall objective for generated images $\hat{\mathcal{I}}$ in Gaussian splatting training is:

$$\mathcal{L} = \begin{cases} (1-\lambda)\mathcal{L}_1(\tilde{I}_i, I_i) + \lambda\mathcal{L}_{SSIM}(\tilde{I}_i, I_i) & ,\forall\, I_i \in \mathcal{I} \\ \dfrac{\sum_u \tilde{V}_i^j(u) \cdot \tilde{W}_i^j(u) \cdot |\tilde{I}_i^j(u) - \hat{I}_i^j(u)|}{\sum_u \tilde{V}_i^j(u)} & ,\forall\, \hat{I}_i^j \in \hat{\mathcal{I}}, \end{cases} \tag{9}$$

where the standard Gaussian splatting loss is applied to the original images, while the weighted and masked L1 loss is used for the generated images. Here $\tilde{I}_i$ and $\tilde{I}_i^j$ denote Gaussian splatting renderings from P_i and P_i^j, respectively.

C. Preprocessing for real-world scenes

For real-word scenes, obtaining camera poses $\mathcal{P}$ and depth maps $\mathcal{D}$ is often challenging due to the limited accessibility of precise sensors and the expense associated with specialized imaging devices. Standard Gaussian splatting and NeRF pipelines usually rely on structure-from-motion as a preprocessing step to estimate both camera parameters and an initial point cloud. However, commonly used SfM methods, such as COLMAP [10], often struggle to converge in sparse-view scenarios with a limited number of images.

To address this challenge, we adpot DUSt3R [12] in place of COLMAP. DUSt3R processes image pairs using a shared ViT [43] encoder to extract representations, which are then fed into two Transformer-based decoders [44] equipped with cross-attention layers. These decoders predict two aligned point clouds with associated per-point confidence through regression heads. A subsequent global optimization step allows DUSt3R to determine, for each image I_i, the corresponding camera pose P_i, point cloud PC_i, and scale-aligned depth map D_i (derived from the z-coordinate of the predicted point cloud), as well as a confidence map C_i.

Since our focus is vehicle reconstruction, we further refine the images by leveraging Segment Anything [45] to remove backgrounds. This step generates a set of segmentation masks, $\mathcal{F} = \{F_i\}_{i=1}^N$.

The overall pipeline is kept consistent with Sec. III-A and III-B, except for two modifications:

- Each image I_i is initially filtered using the segmentation mask, yielding $I_i = I_i \odot F_i$.
- The point cloud PC_i, obtained from Eq. 2, is refined based on both the segmentation mask and the confidence map, as follows:

$$PC_i = \{p_k \in PC_i \mid F_i(u_k) = 1 \text{ and } C_i(u_k) > c\}, \tag{10}$$

where u_k is the 2D pixel location corresponding to the 3D point p_k in PC_i, and c is a fixed confidence threshold.

IV. Experiments

All experiments were conducted on a TITAN RTX GPU.

Competitor Implementation. BRUM performance has been compared in both synthetic and real-world scenarios against Gaussian Splatting and two state-of-the-art architectures designed for sparse input settings: DNGaussians [46] and SplatFields [36]. To ensure fair comparisons, we introduced minor yet essential modifications to these methods to better align them with the requirements of our use case. Specifically, for SplatFields, the original implementation proposes scaling the initial learning rate by a constant value. However, we found that dynamically adjusting the learning rate based on the characteristics of each training scene was necessary to achieve optimal performance in our experimental setup. In the case of DNGaussians, instead of optimizing parameters individually for each scene, we employed the default settings provided for the LEGO scene in the Blender dataset. This approach ensured the reproducibility of the results over different scenarios.

Evaluation Metrics. We evaluate our method using standard visual quality metrics: Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index Measure (SSIM) [42], and Learned Perceptual Image Patch Similarity (LPIPS) [47], which assess reconstruction fidelity, structural consistency, and perceptual quality, respectively. To provide a more intuitive and comprehensive comparison, we also report the Average Error (AVGE) [48]. This metric integrates multiple aspects of visual quality by combining $MSE = 10^{-\text{PSNR}/10}$, $\sqrt{1-\text{SSIM}}$, and LPIPS as a geometric mean, capturing both pixel-wise and perceptual errors in a unified score.

Rendering. For the rendering process, we use the `PointsRenderer` module from PyTorch3D. The number of points contributing to a pixel's color (*i.e.* the K parameter introduced in Sec. III-A) is set to 16 for both synthetic and

TABLE I: Quantitative results averaged over the BRUM-dataset and CarPatch scenes.

Method	BRUM-dataset PSNR ↑	SSIM ↑	LPIPS ↓	AVGE ↓	CarPatch PSNR ↑	SSIM ↑	LPIPS ↓	AVGE ↓
3DGS [2]	21.08	0.861	0.132	0.073	22.42	0.867	0.122	0.063
DNGaussians [46]	17.33	0.617	0.187	0.129	21.23	0.824	0.144	0.077
SplatFields [36]	23.80	0.887	0.118	0.055	23.39	0.889	0.116	0.056
BRUM	**24.65**	**0.917**	**0.083**	**0.043**	**25.57**	**0.911**	**0.079**	**0.040**

Fig. 2: Overview of the BRUM-dataset.

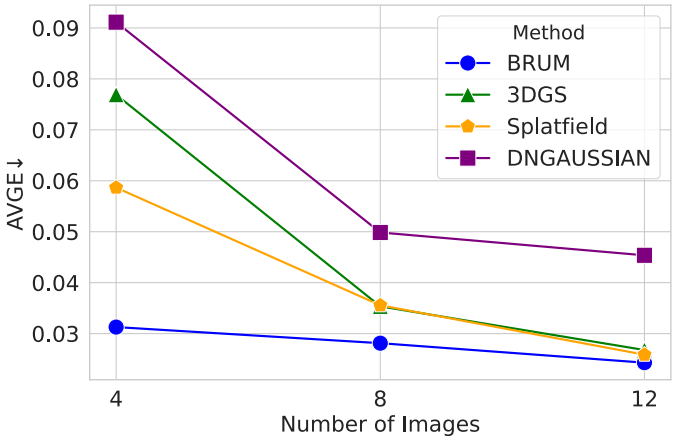


Fig. 3: AVGE results by varying number of training images.

real-world datasets to ensure smooth and consistent outputs. The r parameter (Eq. 4), which controls the projected size of points, is assigned a value of 0.003 for synthetic data and 0.1 for real-world data, allowing for an optimal balance between preserving fine details and minimizing overlapping effects. These parameter configurations facilitate high-quality renderings with minimal visual artifacts.

A. The BRUM-dataset

Given the lack of data in literature representing public transportation vehicles, we publicly release the BRUM-dataset. This dataset comprises 12 scenes including 6 real-world 360° captures of buses and 6 synthetic bus models. For both settings, the buses are labeled as bus_1 through bus_6.

Synthetic. BRUM-dataset consists of six distinct synthetic scenes, each featuring a unique 3D bus model. All the 3D models have been downloaded from Sketchfab[2] and 3DExport[3]. These scenes were generated using Blender[4], inspired by the Google Blender dataset setup [1]. This approach allowed precise control over lighting conditions and camera viewpoints, providing a proof of concept for evaluating our method under challenging conditions. Each scene was designed with realistic rendering settings: the vehicle was positioned at the origin $(0, 0, 0)$, surrounded by nine lights with varying emission strengths to produce realistic shadows and reflections. Objects were resized to match real-world dimensions, the camera and lighting were configured to mimic real-world environments. Each scene includes 100 training images and 200 test images, accompanied by ground truth depth annotations and camera positions. For training, the camera was randomly positioned on a hemisphere above the ground, with rotation angles sampled uniformly. Instead, test images were captured with the camera at a fixed height, rotating around the Z-axis in steps of $\frac{2\pi}{\#test\_views}$ radians per frame.

Real-World. BRUM-dataset also comes with six distinct real bus scenes, selected to represent a diverse range of designs and manufacturers. The data collection was performed in a controlled environment using a DJI MINI 2 SE drone. For each bus, a 360° video was captured by flying multiple laps around the vehicle at varying altitudes. The initial lap was performed at eye level, followed by subsequent laps with the altitude gradually increased by approximately 3 meters, ensuring thorough coverage of each bus's structure. The original videos were recorded at 30 fps with a resolution of 1920×1080. For compatibility with novel view synthesis techniques, individual frames were extracted and sub-sampled. For each bus, a masked version of all frames is provided as mentioned in III-C.

B. Results on synthetic scenes

Implementation Details For synthetic evaluation, we use the BRUM-dataset and CarPatch [3] datasets as benchmarks to assess 3D reconstruction performance. To simulate a sparse input scenario during training, we sampled 4 images from the training dataset. All images have a resolution of 800×800 and are chosen from distinct viewpoints around the vehicles to

[2] https://sketchfab.com
[3] https://3dexport.com
[4] http://www.blender.org

TABLE II: Quantitative results averaged over the KRONC and BRUM-dataset scenes.

Method	KRONC				BRUM-dataset			
	PSNR ↑	SSIM ↑	LPIPS ↓	AVGE ↓	PSNR ↑	SSIM ↑	LPIPS ↓	AVGE ↓
3DGS [2]	19.44	0.764	0.153	0.094	19.65	0.734	0.192	0.102
DNGaussians [46]	17.09	0.704	0.243	0.137	17.93	0.701	0.258	0.132
SplatFields [36]	17.81	0.739	0.201	0.119	17.74	0.682	0.207	0.125
BRUM	**20.37**	**0.797**	**0.143**	**0.084**	**20.62**	**0.762**	**0.179**	**0.091**

ensure comprehensive visibility of the object's structure. The h factor of SLERP (Eq. 1) varies from 0.025 to 0.975 (included) with a step size of 0.025. This results in a total of 156 novel views for a scene with only 4 images. Ground-truth (GT) camera poses and GT depth maps are employed during the image augmentation phase (Sec. III-A) to enhance rendering accuracy. To ensure fair comparison, SplatFields is evaluated using GT camera poses, while DNGaussians employs also GT depth maps directly, bypassing the DPT [49] computation suggested in the original work. Given that all these methods utilize the 3DGS codebase, we conducted experiments in the synthetic scenario by initializing the point cloud for 3DGS with random values. This approach ensures a fair comparison of their performance under identical initial conditions.

Main Results. As demonstrated in Table I, BRUM exhibits significant advantages in the context of novel view synthesis in challenging sparse-view scenarios. The results clearly demonstrate the effectiveness of BRUM compared to other state-of-the-art methods. For the BRUM-dataset and CarPatch datasets, BRUM consistently improves all the considered metrics. These results highlight BRUM's ability to preserve perceptual quality and geometric accuracy, while demonstrating scalability and generalization across diverse scenes. Notably, 3DGS, while competitive in some cases, falls short in preserving perceptual details (higher LPIPS) and maintaining geometric accuracy (higher AVGE) compared to BRUM.

Additional Analysis. Figure 3 compares the average error (AVGE) of BRUM with other approaches under varying numbers of input views. The results demonstrate that our method consistently outperforms the others, particularly in sparse view scenarios with 4 or 8 input images, highlighting its capability to reconstruct 3D scenes effectively even in under-constrained conditions. As the number of input images increases to 12, the AVGE converges with that of SplatField and 3DGS, showcasing its robustness and scalability to denser input settings. In terms of runtime, our approach introduces an extra preprocessing overhead ranging from 25 seconds to 10 minutes, depending on the number of rendered images. For sampling and rendering novel views, it does not affect memory usage, which remains consistent. BRUM strikes a balance by achieving superior reconstruction quality compared to 3DGS while maintaining a reasonable computation time.

C. Results on real-world scenes

Implementation details. For the real-world scenario, we evaluate our method on BRUM-dataset and KRONC-dataset [4]. Unlike the synthetic case, real-world datasets lack ground-truth camera poses and depth maps. We adopt DUSt3R as a preprocessing step to estimate these parameters, as mentioned in section III-C. However, the uncertainties in this estimation process pose challenges for accurate 3D reconstruction. We relax the 4-view constraint and train with 8 views to compensate for inaccuracies in camera pose and depth estimation.

TABLE III: Quantitative results for different interpolation factors, computed on the Ford scene of the KRONC dataset.

h	PSNR ↑	SSIM ↑	LPIPS ↓	AVGE ↓
0.05	21.01	0.824	0.125	0.075
0.1	**21.18**	**0.827**	**0.114**	**0.071**
0.3	21.13	0.827	0.120	0.073
0.5	20.95	0.823	0.119	0.074

To minimize rendering distortions caused by imprecisions in the depth maps estimated by DUSt3R, the h factor in SLERP (Eq. 1) for real-world scenes is restricted to a narrower range compared to synthetic scenes, while maintaining a fixed step size of 0.025. Specifically, for KRONC-dataset h is fixed at 0.1 for all experimentis while for the BRUM-dataset is fixed to 0.08. During the preprocessing step (Sec. III-C), we examine the influence of DUSt3R's confidence parameter c. For the KRONC-dataset, the optimal c is found to be 0, allowing all points predicted by DUSt3R to be utilized. In contrast, for BRUM-dataset, the optimal c is determined to be 1.5. In the following sections, we analyze how varying h and c influence reconstruction metrics. To meet DUSt3R's requirements, all input images were downsampled to 512×256.

Main Results. As shown in Table II, BRUM exhibits strong performance in real-world scenarios, though slightly less pronounced than in the synthetic setting (Table I). Real-world datasets present additional challenges, including noise, reflections, and inaccuracies in estimated depth maps and camera poses. Despite these obstacles, BRUM achieves the highest scores across both datasets. Notably, neither DNGaussians nor SplatFields were originally designed to handle sparse 360° forward-facing real scenes in their formulations. This explains their lower performance highlighting the difficulty of our task.

Additional analysis. The results in Table III provide valuable insights into the effect of the maximum interpolation factor h on BRUM's performance in real-world scenarios. Here, h represents the upper limit for interpolation, while the step size remains fixed at 0.025. Higher values of h indicate moving farther away from the ground truth camera position, resulting in the generation of more synthetic views. For smaller

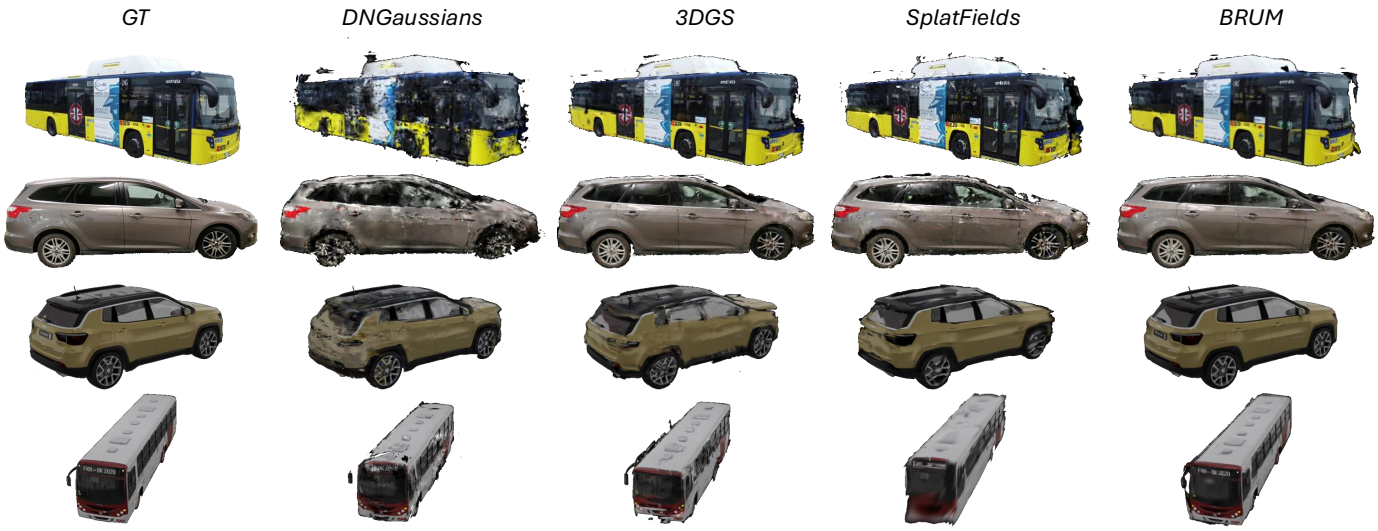


Fig. 4: Qualitative comparison of 3D reconstruction methods for DNGaussian, 3DGS, SplatFields, and BRUM.

TABLE IV: Performance of BRUM evaluated with varying c (Eq. 10) on the bus_1 scene of the BRUM-dataset.

c	PSNR ↑	SSIM ↑	LPIPS ↓	AVGE ↓
0	17.82	0.754	0.200	0.118
0.5	17.81	0.755	0.199	0.118
1	17.67	0.752	0.201	0.120
1.5	**18.41**	**0.789**	**0.167**	**0.103**
2	18.13	0.787	0.169	0.106
2.5	18.18	0.785	0.170	0.106

TABLE V: Quantitative results for various configurations. The best-performing corresponds to BRUM, as described in Sec. III-A, where the SSIM loss is excluded during training.

$\mathcal{L}_{SSIM}$	XNOR	$\hat{W}$	PSNR ↑	SSIM ↑	LPIPS ↓	AVGE ↓
✓	✗	✓	20.07	0.796	0.148	0.087
✓	✓	✗	20.46	0.798	0.131	0.081
✓	✓	✓	20.95	0.808	0.128	0.077
✗	✓	✓	**21.18**	**0.827**	**0.116**	**0.072**

values of h, such as 0.05 and 0.1, BRUM achieves superior preservation of image quality compared to the ground truth. However, as h increases (*e.g.*, 0.3 and 0.5), AVGE experiences a marginal increase. This suggests that larger h values may amplify distortions caused by inaccuracies in depth map and camera pose estimations.

Table IV examines the impact of c (Eq. 10) on reconstruction performance using the bus_1 scene from BRUM-dataset. Higher c values remove more points from the DUSt3R point cloud, leaving larger image portions unprojected. Lower c values (*e.g.*, 0 or 0.5) fail to filter depth estimation uncertainties, while higher values (*e.g.*, 2 or 2.5) overly prune the point cloud, causing information loss. A value of $c = 1.5$ provides the best balance, improving reconstruction accuracy.

Finally, Table V evaluates the impact of BRUM's key components, as outlined in Section III-A. The most significant degradation occurs when the XNOR operation from Eq.6 is removed, and the original foreground mask $\hat{V}$ is used. Additionally, omitting the weighting factor $\tilde{W}$ or reintroducing $\mathcal{L}_{SSIM}$ for generated images in Eq.9 results in noticeable performance drops. The complete method achieves the best metrics, validating the importance of each component.

D. Qualitative results

In Fig. 4, we compare qualitative results on the BRUM-dataset, KRONC, and CarPatch datasets. The proposed method accurately reconstructs vehicles in both synthetic and real domains, preserving fine details and producing outputs closer to ground truth data.

V. Conclusion

This paper addresses the challenge of 3D vehicle reconstruction from sparse views. By leveraging depth maps and the DUSt3R architecture, BRUM significantly enhances the robustness of Gaussian Splatting in scenarios with limited input data. We introduce a novel dataset combining synthetic and real-world public transport vehicles, achieving high-quality reconstructions across challenging scenes. The scalability and adaptability of our approach make it well-suited for practical deployment in resource-constrained environments.

Acknowledgment

This research was partially funded by the International Foundation Big Data and Artificial Intelligence for Human Development (IFAB). The authors also acknowledge SETA S.p.A. for allowing to capture the real public transportation scenes of BRUM-dataset featured in this work.

References

[1] 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," *Communications of the ACM*, 2021.

[2] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3d gaussian splatting for real-time radiance field rendering." *ACM Trans. Graph.*, 2023.

[3] D. Di Nucci, A. Simoni, M. Tomei, L. Ciuffreda, R. Vezzani, and R. Cucchiara, "Carpatch: A synthetic benchmark for radiance field evaluation on vehicle components," in *International Conference on Image Analysis and Processing*, 2023.

[4] ——, "Kronc: Keypoint-based robust camera optimization for 3d car reconstruction," in *Eur. Conf. Comput. Vis. Worksh.*, 2024.

[5] A. Tonderski, C. Lindström, G. Hess, W. Ljungbergh, L. Svensson, and C. Petersson, "Neurad: Neural rendering for autonomous driving," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2024.

[6] X. Zhou, Z. Lin, X. Shan, Y. Wang, D. Sun, and M.-H. Yang, "Drivinggaussian: Composite gaussian splatting for surrounding dynamic autonomous driving scenes," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2024.

[7] H. Turki, D. Ramanan, and M. Satyanarayanan, "Mega-nerf: Scalable construction of large-scale nerfs for virtual fly-throughs," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2022.

[8] H. Xie, Z. Chen, F. Hong, and Z. Liu, "Citydreamer: Compositional generative model of unbounded 3d cities," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2024.

[9] Y. Liu, C. Luo, L. Fan, N. Wang, J. Peng, and Z. Zhang, "Citygaussian: Real-time high-quality large-scale scene rendering with gaussians," in *Eur. Conf. Comput. Vis.*, 2024.

[10] J. L. Schonberger and J.-M. Frahm, "Structure-from-motion revisited," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2016.

[11] P. Truong, M.-J. Rakotosaona, F. Manhardt, and F. Tombari, "Sparf: Neural radiance fields from sparse and noisy poses," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2023.

[12] S. Wang, V. Leroy, Y. Cabon, B. Chidlovskii, and J. Revaud, "Dust3r: Geometric 3d vision made easy," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2024.

[13] T. Müller, A. Evans, C. Schied, and A. Keller, "Instant neural graphics primitives with a multiresolution hash encoding," *ACM Trans. Graph.*, 2022.

[14] S. Fridovich-Keil, A. Yu, M. Tancik, Q. Chen, B. Recht, and A. Kanazawa, "Plenoxels: Radiance fields without neural networks," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2022.

[15] C.-H. Lin, W.-C. Ma, A. Torralba, and S. Lucey, "Barf: Bundle-adjusting neural radiance fields," in *Int. Conf. Comput. Vis.*, 2021.

[16] Y. Chen, X. Chen, X. Wang, Q. Zhang, Y. Guo, Y. Shan, and F. Wang, "Local-to-global registration for bundle-adjusting neural radiance fields," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2023.

[17] W. Bian, Z. Wang, K. Li, J.-W. Bian, and V. A. Prisacariu, "Nope-nerf: Optimising neural radiance field with no pose prior," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2023.

[18] S. Chen, Y. Bhalgat, X. Li, J.-W. Bian, K. Li, Z. Wang, and V. A. Prisacariu, "Neural refinement for absolute pose regression with feature synthesis," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2024.

[19] W. Morgenstern, F. Barthel, A. Hilsmann, and P. Eisert, "Compact 3d scene representation via self-organizing gaussian grids," in *Eur. Conf. Comput. Vis.*, 2024.

[20] S. Niedermayr, J. Stumpfegger, and R. Westermann, "Compressed 3d gaussian splatting for accelerated novel view synthesis," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2024.

[21] Z. Fan, K. Wang, K. Wen, Z. Zhu, D. Xu, and Z. Wang, "Lightgaussian: Unbounded 3d gaussian compression with 15x reduction and 200+ fps," in *Adv. Neural Inform. Process. Syst.*, 2024.

[22] X. Zhang, X. Ge, T. Xu, D. He, Y. Wang, H. Qin, G. Lu, J. Geng, and J. Zhang, "Gaussianimage: 1000 fps image representation and compression by 2d gaussian splatting," in *Eur. Conf. Comput. Vis.*, 2024.

[23] J. Cen, Z. Zhou, J. Fang, W. Shen, L. Xie, D. Jiang, X. Zhang, Q. Tian *et al.*, "Segment anything in 3d with nerfs," in *Adv. Neural Inform. Process. Syst.*, 2023.

[24] X. Hu, Y. Wang, L. Fan, J. Fan, J. Peng, Z. Lei, Q. Li, and Z. Zhang, "Semantic anything in 3d gaussians," *arXiv:2401.17857*, 2024.

[25] Y. Chen, Z. Chen, C. Zhang, F. Wang, X. Yang, Y. Wang, Z. Cai, L. Yang, H. Liu, and G. Lin, "Gaussianeditor: Swift and controllable 3d editing with gaussian splatting," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2024.

[26] B. Poole, A. Jain, J. T. Barron, and B. Mildenhall, "Dreamfusion: Text-to-3d using 2d diffusion," *arXiv:2209.14988*, 2022.

[27] S. Zhou, Z. Fan, D. Xu, H. Chang, P. Chari, T. Bharadwaj, S. You, Z. Wang, and A. Kadambi, "Dreamscene360: Unconstrained text-to-3d scene generation with panoramic gaussian splatting," in *Eur. Conf. Comput. Vis.*, 2024.

[28] Z. Chen, F. Wang, Y. Wang, and H. Liu, "Text-to-3d using gaussian splatting," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2024.

[29] Y. Liang, X. Yang, J. Lin, H. Li, X. Xu, and Y. Chen, "Luciddreamer: Towards high-fidelity text-to-3d generation via interval score matching," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2024.

[30] A. Yu, V. Ye, M. Tancik, and A. Kanazawa, "pixelnerf: Neural radiance fields from one or few images," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021.

[31] Q. Wang, Z. Wang, K. Genova, P. P. Srinivasan, H. Zhou, J. T. Barron, R. Martin-Brualla, N. Snavely, and T. Funkhouser, "Ibrnet: Learning multi-view image-based rendering," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021.

[32] M. Mihajlovic, A. Bansal, M. Zollhoefer, S. Tang, and S. Saito, "Keypointnerf: Generalizing image-based volumetric avatars using relative spatial encoding of keypoints," in *Eur. Conf. Comput. Vis.*, 2022.

[33] Y. Lao, X. Xu, X. Liu, H. Zhao *et al.*, "Corresnerf: Image correspondence priors for neural radiance fields," in *Adv. Neural Inform. Process. Syst.*, 2023.

[34] K. Deng, A. Liu, J.-Y. Zhu, and D. Ramanan, "Depth-supervised nerf: Fewer views and faster training for free," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2022.

[35] B. Roessle, J. T. Barron, B. Mildenhall, P. P. Srinivasan, and M. Nießner, "Dense depth priors for neural radiance fields from sparse input views," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2022.

[36] M. Mihajlovic, S. Prokudin, S. Tang, R. Maier, F. Bogo, T. Tung, and E. Boyer, "Splatfields: Neural gaussian splats for sparse 3d and 4d reconstruction," in *Eur. Conf. Comput. Vis.*, 2024.

[37] A. Paliwal, W. Ye, J. Xiong, D. Kotovenko, R. Ranjan, V. Chandra, and N. K. Kalantari, "Coherentgs: Sparse novel view synthesis with coherent 3d gaussians," in *Eur. Conf. Comput. Vis.*, 2024.

[38] R. Yin, V. Yugay, Y. Li, S. Karaoglu, and T. Gevers, "Fewviewgs: Gaussian splatting with few view matching and multi-stage training," in *Adv. Neural Inform. Process. Syst.*, 2024.

[39] R. Peng, W. Xu, L. Tang, L. Liao, J. Jiao, and R. Wang, "Structure consistent gaussian splatting with matching prior for few-shot novel view synthesis," in *Adv. Neural Inform. Process. Syst.*, 2024.

[40] K. Shoemake, "Animating rotation with quaternion curves," in *Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, 1985, pp. 245–254.

[41] O. Wiles, G. Gkioxari, R. Szeliski, and J. Johnson, "Synsin: End-to-end view synthesis from a single image," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020.

[42] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Trans. Image Process.*, 2004.

[43] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv:2010.11929*, 2020.

[44] A. Vaswani, "Attention is all you need," *Adv. Neural Inform. Process. Syst.*, 2017.

[45] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo *et al.*, "Segment anything," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2023.

[46] 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," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2024.

[47] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018.

[48] M. Niemeyer, J. T. Barron, B. Mildenhall, M. S. Sajjadi, A. Geiger, and N. Radwan, "Regnerf: Regularizing neural radiance fields for view synthesis from sparse inputs," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2022.

[49] R. Ranftl, A. Bochkovskiy, and V. Koltun, "Vision transformers for dense prediction," in *Int. Conf. Comput. Vis.*, 2021.