SCube: Instant Large-Scale Scene Reconstruction using VoxSplats

Xuanchi Ren[1,2,3]*, **Yifan Lu**[1,4]*, **Hanxue Liang**[1,5], **Zhangjie Wu**[1,6], **Huan Ling**[1,2,3], **Mike Chen**[1], **Sanja Fidler**[1,2,3], **Francis Williams**[1], **Jiahui Huang**[1]

[1]NVIDIA, [2]University of Toronto, [3]Vector Institute, [4]Shanghai Jiao Tong University
[5]University of Cambridge, [6]National University of Singapore
https://research.nvidia.com/labs/toronto-ai/scube/

Abstract

We present SCube, a novel method for reconstructing large-scale 3D scenes (geometry, appearance, and semantics) from a sparse set of posed images. Our method encodes reconstructed scenes using a novel representation VoxSplat, which is a set of 3D Gaussians supported on a high-resolution sparse-voxel scaffold. To reconstruct a VoxSplat from images, we employ a hierarchical voxel latent diffusion model conditioned on the input images followed by a feedforward appearance prediction model. The diffusion model generates high-resolution grids progressively in a coarse-to-fine manner, and the appearance network predicts a set of Gaussians within each voxel. From as few as *3 non-overlapping input images*, SCube can generate millions of Gaussians with a 1024^3 voxel grid spanning *hundreds of meters* in *20 seconds*. Past works tackling scene reconstruction from images either rely on per-scene optimization and fail to reconstruct the scene away from input views (thus requiring dense view coverage as input) or leverage geometric priors based on low-resolution models, which produce blurry results. In contrast, SCube leverages high-resolution sparse networks and produces sharp outputs from few views. We show the superiority of SCube compared to prior art using the Waymo self-driving dataset on 3D reconstruction and demonstrate its applications, such as LiDAR simulation and text-to-scene generation.

1 Introduction

Recovering 3D geometry and appearance from images is a fundamental problem in computer vision and graphics which has been studied for decades. This task lies at the core of many practical applications spanning robotics, autonomous driving, and augmented reality; just to name a few. Early algorithms tackling this problem use stereo matching and structure from motion (SfM) to recover 3D signals from image data (*e.g.*[44]). More recently, a line of work starting from Neural Radiance Fields [32] (NeRFs) has augmented traditional SfM pipelines by fitting a volumetric field to a set of images, which can be rendered from novel views. NeRFs augment traditional reconstruction pipelines by encoding dense geometry, and view-dependent lighting effects. While radiance-field methods present a drastic step forward in our ability to recover 3D information from images, they require a time-consuming per-scene optimization scheme. Furthermore, since each scene is recovered in isolation, radiance fields do not make use of data priors, and cannot extrapolate reconstructions away from the input views. Thus, radiance-field methods require dense view coverage in order to produce high-quality 3D reconstructions.

*Equal contribution.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

https://doi.org/10.52202/079017-3099

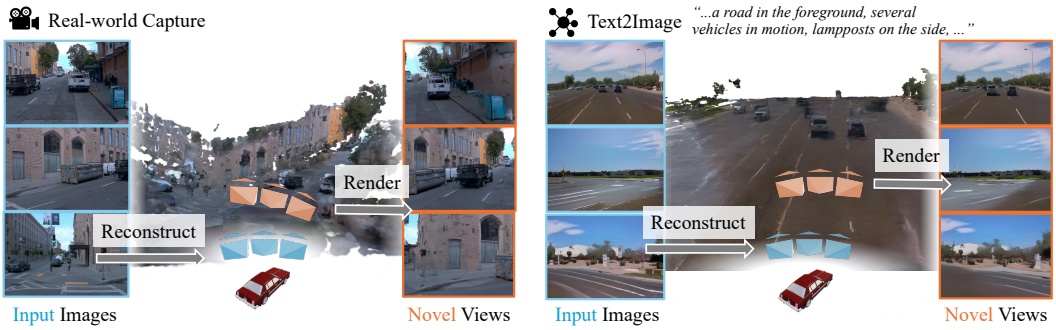


Figure 1: **SCube.** Given sparse input images with little or no overlap, our model reconstructs a high-resolution and large-scale scene in 3D represented with VoxSplats, ready to be used for novel view synthesis or LiDAR simulation.

Another recent line of work applies deep learning to predict 3D from images. These methods either meta-learn an initialization to the radiance-field optimization problem [7, 30, 49], or directly predict 3D from images using a feed-forward network [17, 57, 73]. While learning-based approaches can produce reconstructions from sparse views, they have only been used successfully for the case of single objects at low resolutions. Furthermore, these methods often suffer from 3D inconsistencies (*e.g.* the multi-layer surface or the Janus problem). In order to solve the general 3D reconstruction from images problem, we need methods that can *(1)* generalize reconstruction to general scenes over the pure object case, *(2)* produce accurate and high-quality reconstructions in the presence of dense views, leveraging data priors to produce plausible reconstructions in the sparse-view regime, and *(3)* run quickly and efficiently (in terms of runtime and memory) on large-scale and high-resolution inputs. These demands are difficult to satisfy in practice since high-quality ground-truth 3D data is not widely available for scenes, 3D representations for deep learning that scale to large and diverse inputs are under-explored in the literature, and corresponding scalable and easy-to-train model designs need to be developed alongside any new 3D representation.

Nevertheless, we remark that some of these issues have been resolved in isolation: Gaussian Splatting [23] enables fast, differentiable rendering and high reconstruction quality (but is not being used with data priors), and sparse voxel hierarchies [40] have been successfully used to build generative models of large-scale 3D scenes with attributes such as semantics and colors, and have been trained on partial data such as LiDAR scans from autonomous vehicle captures.

In light of the above observations, we introduce SCube, a feed-forward method for large 3D scene reconstruction from images. Our method encodes 3D scenes as a hybrid of Gaussian splats (which enable fast rendering), supported on a sparse-voxel-hierarchy (which enables efficient generative modeling of large 3D scenes with semantics). We call this hybrid representation VoxSplats and predict a VoxSplat from images using a feed-forward process consisting of two steps: *(1)* A generative geometry network that predicts a sparse voxel hierarchy conditioned on input images, and *(2)* an appearance network that predicts the Gaussian attributes within the voxels as well as a skybox texture to represent the background. The networks are implemented using highly efficient sparse convolution [14, 40] designed for 3D data which enables us to reconstruct a full scene from images in under 20 seconds. We evaluate our performance on the Waymo Open Dataset [53] on the challenging task of reconstructing a scene from sparse images with low overlap. We show that SCube significantly outperforms existing methods on this task. Furthermore, we demonstrate that SCube enables downstream applications such as LiDAR simulation and text-to-scene generation.

2 Related Work

3D Scene Representation. Scenes in the wild are often large in scale and contain complicated internal structures which cause representations such as tri-planes [12], dense voxel grids [36], or meshes [19, 46] to fail due to capacity or memory limitations. Optimization-based reconstruction methods [15, 32] use high-resolution hash grids [1, 33], but these are non-trivial to infer using a neural

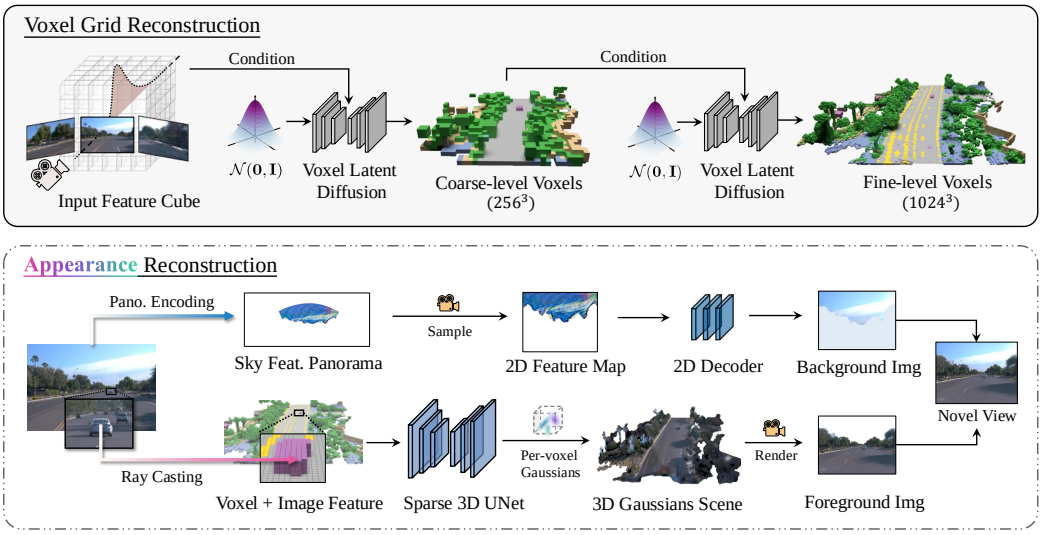


Figure 2: **Framework.** SCube consists of two stages: **(1)** We reconstruct a sparse voxel grid with semantic logit conditioned on the input images using a conditional latent diffusion model based on XCube [40]. **(2)** We predict the appearance of the scene represented as VoxSplats and a sky panorama using a feedforward network. Our method allows us to synthesize novel views in a fast and accurate manner, along with many other applications.

network [30]. In contrast, sparse voxel grids are effective for learning scene-reconstruction [40, 75] thanks to efficient sparse neural operators [8, 54]. Recently, Gaussian splatting [23] has enabled real-time neural rendering and has been applied to overfitting large scenes [66, 76]. [31, 39] use a hybrid of the above two representations, but the voxel grid or octree is only used to regularize the Gaussian positions without any data priors learned. This is in contrast to our VoxSplat that allows reconstruction in a direct inference pass thanks to the efficiency of sparse grids and the high representation power of Gaussian splats. We support operating only on sparse-view images, significantly lifting the input requirements by learning from large datasets.

Sparse-view 3D Reconstruction. Sparse-view images often contain insufficient correspondences required by traditional reconstruction methods [44]. One line of work uses learned image-space priors such as depth [9], normal maps [70], and appearance from GANs [43] or diffusion models [63] to augment an optimization process such as NeRF. To speed up inference, another line of work uses a feed-forward model to predict renderable features [4, 6, 22, 29, 57, 73]. Alternatively, some papers perform learning directly in 3D space, which yields better consistency and less distortion [5, 13, 17, 68]. Our setting is similar to [13] where input images come from the same rig, but ours is more challenging since we do not use temporally-sequenced inputs with high overlap. We remark that semantic scene completion works [21, 26, 51, 61] also reconstruct voxels but at much lower resolutions and without appearance.

Generative Models for 3D. 3D reconstruction can also be formulated as a conditional generative problem (i.e.modeling the distribution of scenes given partial observations). Text and single-image to-3D generation has been explored for objects [17, 28, 38, 47, 55, 56, 60, 69]. Extending this task to large-scenes is comparatively unexplored, and object-based methods often fail due to scaling limitations or assumptions on the data. [48, 72] recursively apply an image generative model to inpaint missing regions in 3D, but produces blurry reconstructions at a limited scale. XCube [40] is among the first to directly learn high-resolution 3D scene priors. Here, we extend this model with multiview image conditioning and enable it to predict appearance on top of geometry.

3 Method

Our method reconstructs a high-resolution 3D scene from N sparse images $\mathcal{I} = \{\mathbf{I}^i\}_{i=1}^N$ in two stages: *(1)* We reconstruct the scene geometry represented as a sparse voxel grid $\mathcal{G}$ with semantic

https://doi.org/10.52202/079017-3099

features (§ 3.1). *(2)* We predict the appearance $\mathcal{A}$ of the scene that allows for high-quality novel view synthesis (§ 3.2) using VoxSplats and a sky panorama. We can express our pipeline as taking samples from the distribution $p(\mathcal{G}, \mathcal{A}|\mathcal{I}) = p(\mathcal{A}|\mathcal{G}, \mathcal{I})p(\mathcal{G}|\mathcal{I})$. In order to improve the final view quality of the output, we apply an optional post-processing step discussed in § 3.3.

3.1 Voxel Grid Reconstruction

Background: 3D Generation with XCube. XCube [40] is a 3D generative model that produces high-quality samples for both objects and large outdoor scenes. XCube uses a hierarchical latent diffusion model to generate *sparse voxel hierarchies*, i.e., a hierarchy of sparse voxel grids where each fine voxel is contained within a coarse voxel. XCube learns a distribution over latent $\mathbf{X}$ encoded by a sparse structure Variational Autoencoder (VAE). Both the VAE and the diffusion model are instantiated with sparse convolutional neural networks [14], and can generate geometry at up to 1024^3 resolution. We use XCube as the backbone for our geometry reconstruction module. We remark that while the original paper only focused on unconditional or text-conditioned generation, we introduce a novel image-based conditioning $\mathbf{C}$.

Image Conditioned Geometry Generation. To condition XCube on posed input images, we lift DINO-v2 [34] features computed on the input images to 3D space as follows. First, we use the pre-trained DINOv2 model to extract robust visual features for input images, and process the DINO feature using several trainable 2D conv layers to reduce the feature channel to $C + D$. We then split the channel $C + D$ into two parts for each pixel j and input image index i: one part is a regular C-dimensional feature $\mathbf{F}_j^i$ and the other will be a D-dimensional Softmax-normalized vector $\theta_j^i \in \mathbb{R}^D$. Here θ_j^i can be viewed as a distribution over the depth of the corresponding pixel, and we follow a strategy similar to LSS [37] to unproject the images into a dense 3D voxel grid Ω where v denotes the index of a voxel and $d \in [1, D]$ indexes the depth buckets:

$$\mathbf{F}_{jd}^i = \theta_{jd}^i \cdot \mathbf{F}_j^i, \quad \mathbf{C}_v = \sum_{(i,j,d)} \mathbf{F}_{jd}^i \in \mathbb{R}^C. \tag{1}$$

Note that we quantize the depth into D bins dividing the range from a predefined z_{near} to z_{far}. Unlike image-conditioning techniques used in object-level or indoor-level datasets where the camera frusta have significant overlap, our large-scale outdoor setting only takes sparse low-overlapping views captured from an ego-centric camera. Hence previous methods [28, 50, 52] that broadcast the same features to all the voxels along the rays corresponding to the pixel are not suitable here to precisely locate the geometries such as vehicles. The use of the weight θ allows us to handle occlusions effectively and produce a more accurate conditioning signal. After building $\mathbf{C}$, we directly concatenate it with the latent $\mathbf{X}$ and feed it into the XCube diffusion network as conditioning.

Training and Inference. Our training pipeline is similar to [40], where we first train a VAE to learn a latent space over sparse voxel hierarchies. We add semantic logit prediction as in [40] to the grid and empirically find that it helps the model to learn better geometry. Then we train the diffusion model conditioned on $\mathbf{C}$ using the following loss:

$$\mathcal{L} = \mathcal{L}_{\text{Diffusion}} + \lambda \mathcal{L}_{\text{Depth}}, \quad \mathcal{L}_{\text{Depth}} = \mathbb{E}_{\mathbf{X},i,j} \text{Focal}(\theta_j^i, [\theta_j^i]_{\text{gt}}), \tag{2}$$

where $\mathcal{L}_{\text{Diffusion}}$ is the loss for diffusion model training (see Appendix A for details). Focal$(\cdot)$ is the multi-class focal loss [27]. This additional depth loss is an explicit supervision to properly weigh the image features and encourage correct placement into the corresponding voxels. Due to the generative nature of XCube, we could learn the data prior to generate complete geometry even if some of the ground-truth 3D data is incomplete.

3.2 Appearance Reconstruction

The VoxSplat Representation. In the second stage, we fix the voxel grid $\mathcal{G}$ generated from the geometry stage and predict a set of Gaussian splats in each voxel to model the scene appearance. Gaussian splatting [23] is a powerful 3D representation technique that models a scene's appearance volumetrically as sum of Gaussians:

$$G(\boldsymbol{x}) = \text{RGB} \cdot \alpha \cdot \mathrm{e}^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})}, \tag{3}$$

where $\alpha \in [0, 1]$ is the opacity, $\boldsymbol{\mu} \in \mathbb{R}^3$ is the center of each Gaussian, and $\mathbf{\Sigma} = \mathbf{RSS}^\top \mathbf{R}^\top \in \mathbb{R}^{3\times3}$ is its covariance. The covariance matrix is factorized into a rotation matrix $\mathbf{R}$ parameterized by a quaternion $\mathbf{q}$ and a scale diagonal matrix $\mathbf{S} = \text{diag}(\boldsymbol{s})$. Each Gaussian additionally stores a color value RGB. Note that the original paper uses a set of SH coefficients for view-dependent colors, but we only use the 0^{th}-order SH in our model (i.e., without view-dependency) which we found to be sufficient for sparse-view reconstruction.

While the original Gaussian Splatting paper and its follow-ups [23, 67, 71] propose many heuristics to optimize the positions of Gaussians for a given scene, we instead choose to predict M Gaussians per-voxel using a feed-forward model. We limit the positions of the Gaussians within a neighborhood of their supporting voxels, thus preserving the geometric structure of the supporting grid. By grounding the splats on a voxel *scaffold*, our reconstructions achieve better geometric quality without resorting to heuristics. Fittingly, we dub our voxel-supported Gaussian splats *VoxSplats*.

The output of our network is $\{[\bar{\boldsymbol{\mu}}_v, \bar{\alpha}_v, \bar{\boldsymbol{s}}_v, \bar{\mathbf{q}}_v, \text{RGB}_v] \in \mathbb{R}^{14}\}_M$ for each voxel v. To compute the per-Gaussian parameters used for rendering we apply the following activations:

$$\boldsymbol{\mu}_v = r \cdot \tanh \bar{\boldsymbol{\mu}}_v + \text{Center}_v, \quad \alpha_v = \text{sigmoid}(\bar{\alpha}_v), \quad \boldsymbol{s}_v = \exp \bar{\boldsymbol{s}}_v, \quad \mathbf{R}_v = \text{quat2rot}(\bar{\mathbf{q}}_v), \tag{4}$$

where Center_v is the centroid of the voxel v, and r is a hyperparameter that controls the range of a Gaussian within its supporting voxel. Here, we set r to three times the voxel size. We can efficiently render the Gaussians predicted by our model using rasterization [23] or raytracing [11].

Sky Panorama for Background. To capture appearance away from the predicted geometry, our model builds a sky feature panorama $\mathbf{L} \in \mathbb{R}^{H_p \times W_p \times C_p}$ from all input images, which can be considered as an expanded unit sphere with an inverse equirectangular projection. For each pixel in the panorama $\mathbf{L}$, we get its cartesian coordinate $\mathbf{P} = (x, y, z)$ on the unit sphere and project $\mathbf{P}$ to the image plane to retrieve the image feature; since only the view direction decides the sky color, we zero the translation part of the camera pose in the projection step. We also apply a sky mask to ensure the panorama only focuses on the sky region.

To render a novel viewpoint with its extrinsics and intrinsics, we recover the background appearance by sampling the sky panorama and decoding it into RGB values. For each camera ray from the novel view, we calculate its pixel coordinate on the 2D sky panorama $\mathbf{L}$ with equirectangular projection and get the sky features via trilinear interpolation, resulting in a 2D feature map for the novel view. We finally decode the 2D feature map with a CNN network to get the background image $\mathbf{I}_{\text{bg}}$, which will be alpha-composited with the foreground image from Gaussian rasterization:

$$\mathbf{I}_{\text{pred}}(u, v) = \mathbf{I}_{\text{GS}}(u, v) + (1 - \mathbf{T}(u, v)) \cdot \mathbf{I}_{\text{bg}}(u, v) \tag{5}$$

where $\mathbf{I}_{\text{GS}}(u, v)$ is the rendered image of Gaussians, (u, v) indicates the pixel coordinate, and $\mathbf{T}(u, v)$ is the accumulated transmittance map of the Gaussians (see [23] for details).

Architecture Details. We predict the $(M \times 14)$-dimensional vector $\{[\bar{\boldsymbol{\mu}}_v, \bar{\alpha}_v, \bar{\boldsymbol{s}}_v, \bar{\mathbf{q}}_v, \text{RGB}_v]\}_M$ for each voxel via a 3D sparse convolutional U-Net which takes as input the sparse voxel grid Ω outputted by the geometry stage, where each voxel contains a feature sampled from the input images as follows: We process each input image $\mathbf{I}^i$ using a CNN to get the image feature, and then cast a ray from each image pixel into Ω, accumulating the feature in the first voxel intersected by that ray. Voxels that are not intersected by any rays receive a zero feature vector.

For the sky panorama model, we use the same image feature as above. In the training stage, we set smaller H_p and W_p for faster training and lower memory usage; in the inference stage, we increase H_p and W_p to get a sharper and more detailed sky appearance.

Given a set of training images $\{\mathbf{I}^i_{\text{gt}}\}_i$ and sky masks $\{\mathbf{M}^i\}_i$ distinct from the inputs, we supervise the appearance model using the loss:

$$\mathcal{L} = \lambda_1 \mathcal{L}_1(\mathbf{I}^i_{\text{pred}}, \mathbf{I}^i_{\text{gt}}) + \lambda_2 \mathcal{L}_1(\mathbf{T}^i, \mathbf{M}^i) + \lambda_{\text{SSIM}} \mathcal{L}_{\text{SSIM}}(\mathbf{I}^i_{\text{pred}}, \mathbf{I}^i_{\text{gt}}) + \lambda_{\text{LPIPS}} \mathcal{L}_{\text{LPIPS}}(\mathbf{I}^i_{\text{pred}}, \mathbf{I}^i_{\text{gt}}), \tag{6}$$

where the training views $\mathbf{I}^i_{\text{gt}}$ are sampled from nearby 10 views of the input images; the predicted views $\mathbf{I}^i_{\text{pred}}$ and transmittance masks $\mathbf{T}^i$ are rendered using Eq (5); and $\mathcal{L}_{\text{LPIPS}}$/$\mathcal{L}_{\text{SSIM}}$ are perceptual and structural metrics defined in [74] and [59].

https://doi.org/10.52202/079017-3099

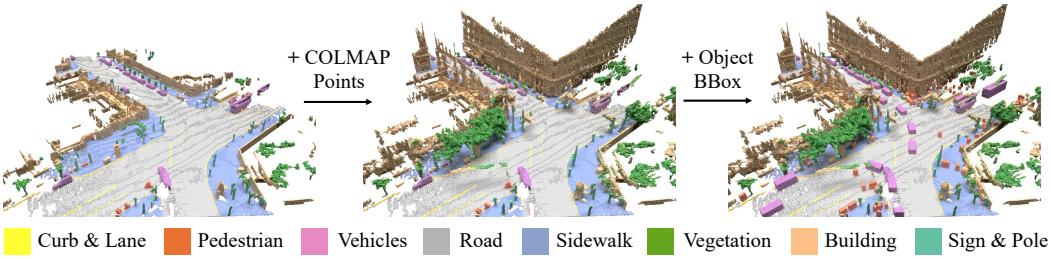


Figure 3: **Data Processing Pipeline.** We add COLMAP [44] dense reconstruction points to the accumulated LiDAR points and compensate for dynamic objects using their bounding boxes. This provides us with a more complete geometry for training.

3.3 Postprocessing and Applications

Optional GAN Postprocessing. The novel views directly rendered from our appearance model sometimes suffer from voxelization artifacts or noise. We resolve this with an optional lightweight conditional Generative Adversarial Network (GAN) that takes the rendered images as input and outputs a refined version. The discriminator of this GAN takes 256×256 image patches sampled from the input sparse view images, as well as the generated images conditioned on the rendered images. Drawing inspiration from [41, 43, 45], we fit the GAN independently for each scene at inference time, which takes $\sim$20min to train. Due to the excessive time cost, we apply this step optionally only when higher-quality images are needed (which we call **SCube+**). Fig. 8 shows the results with and without this step, and we further present a **general postprocessing without per-scene optimization** in Appendix C.

Application: Consistent LiDAR Simulation. LiDAR simulation [77] aims at reproducing the point cloud output given novel locations of the sensor and is an important application for training and verification of autonomous driving systems. The generated LiDAR point clouds should accurately reflect the underlying 3D geometry and a sequence of LiDAR scans should be temporally consistent. Our method enables converting sparse-view images directly into LiDAR point clouds, i.e., a *sensor-to-sensor conversion* scheme. To achieve this, we leverage the output high-resolution Gaussians from our model and ray-trace the LiDAR rays to obtain the corresponding distances. Thanks to our clean voxel scaffold, the reconstructed scene is free of floaters and we set the opacity α to 1 for all the Gaussians to ensure a *hard* intersection that aligns better with the geometry.

Application: Text-to-Scene Generation. Our method can be easily extended to generate 3D scenes from text prompts. Similar to MVDream [47], we train a multi-view diffusion model with the architecture of VideoLDM [2] that generates images from text prompts. The original spatial self-attention layer is inflated along the view dimension to achieve content consistency [25, 65]. For training, we use CogVLM [58] to annotate the images automatically on a large scale. After the model is trained, we directly feed the output of the multi-view model to SCube to lift the observations into 3D space for novel view synthesis.

4 Experiments

In this section, we validate the effectiveness of SCube. First, we present our new data curation pipeline that produces ground-truth voxel grids (§ 4.1). Next, we demonstrate SCube's capabilities in scene reconstruction (§ 4.2), and further highlight its usefulness in assisting the state-of-the-art Gaussian splatting pipeline (§ 4.3). Finally, we showcase other applications of our method (§ 4.4) and perform ablation studies to justify our design choices (§ 4.5).

4.1 Dataset Processing

Accurate 3D data is essential for our method to learn useful geometry and appearance priors. Fortunately, many autonomous driving datasets [3, 53] are equipped with 3D LiDAR data, and one can simply accumulate the point clouds to obtain the 3D scene geometry [20, 40]. However, the LiDAR

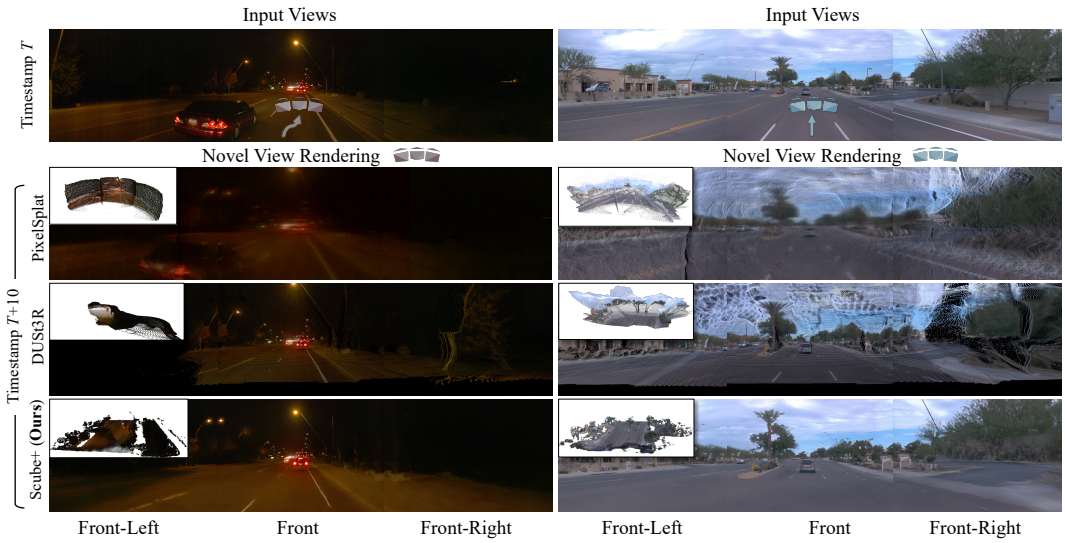
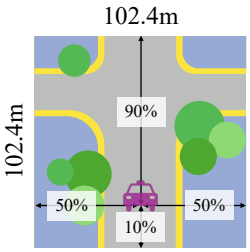


Figure 4: **Novel View Synthesis.** We show the synthesized novel views of SCube+ compared to baselines approaches. The inset of each subfigure shows a top-down visualization (an extreme novel view) of the reconstructed scene geometry.

points usually do not cover high-up regions such as tall buildings and contain dynamic (non-rigid) objects that are non-trivial to accumulate.



We hence build a data processing pipeline based on Waymo Open Dataset [53] as shown in Fig. 3, consisting of three steps: **Step 1**, we accumulate the LiDAR points in the world space, removing the points within the bounding boxes of dynamic objects such as cars and pedestrians. We additionally obtain the semantics of each accumulated LiDAR point, where non-annotated points are assigned the semantics of their nearest annotated neighbors. **Step 2**, we use the multi-view stereo (MVS) algorithm available in COLMAP [44] to reconstruct the dense 3D point cloud from the images, and the semantic labels of the points are obtained by Segformer [64]. **Step 3**, we add point samples for the dynamic objects according to their bounding boxes at a given target frame. This gives us static *and* accumulated ground truths available for training. For each data sample, we crop the point cloud into a local chunk of $102.4\text{m} \times 102.4\text{m}$ centered around a randomly sampled ego-vehicle pose. Since there are no rear-view cameras in the Waymo dataset, we allocate more space for the chunk in the forward direction, as shown in the inset figure. See Appendix A for additional details.

4.2 Large-scale Scene Reconstruction

Evaluation and Baselines. To assess our method's power for 3D scene reconstruction, we follow the common protocol to evaluate the task of novel view synthesis [4, 42, 68]. Given input multi-view images (details about choosing views are in Appendix A) at frame T, we render novel views at future timestamps $T+5$ and $T+10$, and compare them to the corresponding ground-truth frames by calculating Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index Measure (SSIM), and Learned Perceptual Image Patch Similarity (LPIPS) [74]. We exclude the regions of moving objects for $T+5$ and $T+10$ evaluation, and only use three front views when computing the metrics.

We use PixelNeRF [68], PixelSplat [4], DUSt3R [57], MVSplat [6], and MVSGaussian [29] as our baselines for comparison. [4, 6, 29, 68] take images and their corresponding camera parameters as input and reconstruct NeRFs or 3D Gaussian representations. DUSt3R [57] directly estimates per-pixel point clouds from the images. We append additional heads to the its decoder which predicts other 3D Gaussian attributes along with the mean positions and finetune it with a rendering loss. For all other baselines, we take the official code and re-train them on our dataset. We tried to add the

https://doi.org/10.52202/079017-3099

	Reconstruction (T)			Prediction ($T+5$)			Prediction ($T+10$)		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
PixelNeRF [68]	15.26	0.51	0.66	15.21	0.52	0.64	14.61	0.49	0.66
PixelSplat [4]	22.15	0.71	0.61	20.11	0.70	0.60	18.77	0.66	0.62
DUSt3R [57]	17.17	0.60	0.58	17.08	0.62	0.56	16.08	0.58	0.60
MVSplat [6]	21.84	0.71	0.46	20.14	0.71	0.48	18.78	0.69	0.52
MVSGaussian [29]	21.25	0.80	0.51	16.49	0.70	0.60	16.42	0.60	0.59
SCube (**Ours**)	25.90	0.77	0.45	19.90	0.72	0.47	18.78	0.70	0.49
SCube+ (**Ours**)	**28.01**	**0.81**	**0.25**	**22.32**	**0.74**	**0.34**	**21.09**	**0.72**	**0.38**

Table 1: **Quantitative Comparisons on 3D Reconstruction.** The metrics are computed both at the input frame T and future frames. ↑: higher is better, ↓: lower is better.

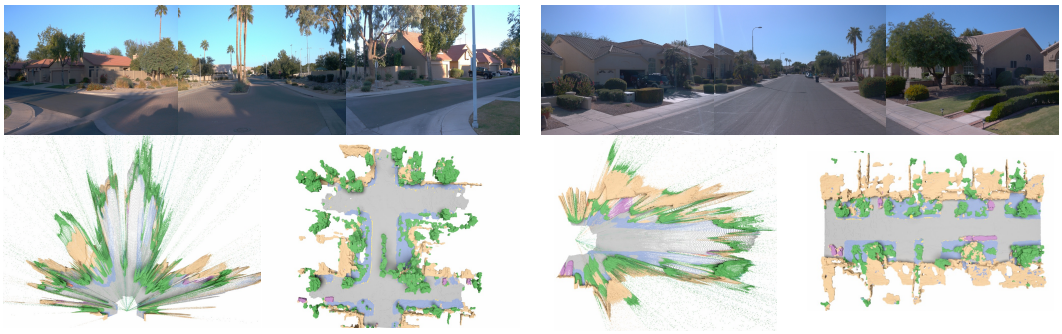


Figure 5: **Geometry Reconstruction from Sparse Views.** We show the comparison between our method and Metric3Dv2 [18]. The semantics of Metric3Dv2 are obtained from Segformer [64].

state-of-the-art depth estimator Metric3Dv2 [18] for depth supervision but empirically found that the performance degraded.

Results and Analysis. We show our results quantitatively in Tab. 1 and visually in Fig. 4. Our method outperforms all the baselines for both the current frame (reconstruction) and future frames (prediction) by a considerable margin on all metrics. PixelNeRF is limited by the representation power of the network, and simply fails to capture high-frequency details in the scene. PixelSplat highly relies on overlapping regions in the input views and cannot adapt well to our sparse view setting. It fails to model the true 3D geometry as shown in the top-down view, and simply collapses the images into constant depths. The multi-view-stereo-based methods [6, 29] cannot enable extreme novel view synthesis such as the bird-eye view, and could not recover highly-occluded regions. Thanks to the effective pre-training of DUSt3R, it is able to learn plausible displacements in the image domain, but the method still suffers from missing regions, misalignments, or inaccurate depth boundaries. In contrast, our method can reconstruct complete scene geometry even for far-away regions. It is both accurate and consistent while producing high-quality novel view renderings.

To better demonstrate the power of learning priors in 3D, we build another baseline using the state-of-the-art metric depth estimator Metric3Dv2 [18] to unproject the images into point clouds using 2D learned priors. As shown in Fig. 5, our method can reconstruct more complete, uniform, and accurate scenes, justifying the power of representing and learning geometry directly in the true 3D space.

4.3 Assisting Gaussian Splatting Initialization

Our method creates scene-level 3D Gaussians with accurate geometry and appearance, which can be used to initialize large-scale 3D Gaussian splatting [23] training. This is particularly useful in outdoor driving scenarios where structure-from-motion (SfM) may fail due to the sparsity of viewpoints.

To demonstrate this, we consider and compare three initialization methods: **Random** initialization is where points are uniformly sampled within the range of $(-20\text{m}, 20\text{m})^3$ around each camera. **Metric3Dv2** initialization is where we use the unprojected cloud from Metric3Dv2 [18]'s monocular

https://doi.org/10.52202/079017-3099

	$R = 10$			$R = 20$			$R = 40$		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Random	21.66	0.72	0.38	24.27	0.78	0.34	24.93	0.80	0.35
Metric3Dv2 [18]	23.30	0.75	0.33	25.21	0.80	0.32	25.58	0.80	0.34
SCube (**Ours**)	**24.10**	**0.77**	**0.32**	**25.94**	**0.81**	**0.30**	**26.07**	**0.82**	**0.32**

Table 2: **Initializations for Gaussian Splatting training.** We train 3D Gaussians with different initialization for R frames. We report the test-set metrics. ↑: higher is better, ↓: lower is better.

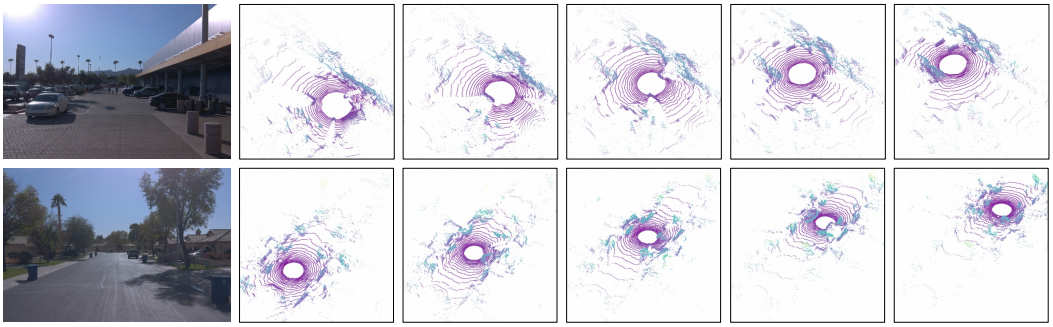


Figure 6: **LiDAR Simulation.** We demonstrate qualitative results of image-to-consistent-LiDAR transfer. The LiDAR sequences are simulated by moving the camera forward by 60 meters.

depth and align its scale to metric-scale LiDAR. **SCube (ours)** initialization directly adopts the positions and colors of the Gaussians from our pipeline. For input to these methods, we choose the views from the first frame T and control the number of initial points to 200k. We then incorporate R subsequent frames into the full training, with every 3 frames skipped to be used in the test set. The number of training iterations is fixed to 30k and the initial positional learning rate is set to $1.6\text{e}{-5}$. We select 15 static scenes for experiments and report their average metrics, which are shown in Tab. 2. The results consistently demonstrate SCube's effectiveness as an initialization strategy that provides accurate 3D grounding and alleviates overfitting on the training views.

4.4 Other Applications

We demonstrate the applications of our method as described in § 3.3. Fig. 6 shows the consistent LiDAR simulation results, where the simulated sequences could effectively cover a long range away from the input camera positions, while resolving intricate geometric structures such as buildings, trees, or poles. Fig. 7 exemplifies the text-to-scene generation capability enabled by our method. The 3D geometry and appearance respect the input text prompt and the corresponding images. Readers are referred to Appendix D.3 for more generative results.

4.5 Ablation Study

Image-Conditioning Strategy. We replace the image conditioning strategy described in Eq (1) in the voxel grid reconstruction stage with a vanilla scheme that broadcasts the same feature to all the voxels along the pixel's ray. The final IoU of the fine-level voxel grid drops from 34.31% to 30.33%, and the mIoU that considers the accuracy of the voxel's semantic prediction drops from 20.00% to 16.61%. This shows the effectiveness of our conditioning strategy being able to disambiguate voxels at different depths.

Two-stage Reconstruction. We disentangle the voxel grid and appearance reconstruction stages to make the best use of different types of models. Using a single-stage model[2] that simultaneously

[2]In practice, we test the upper bound of the single-stage model by feeding in ground-truth 1024^3 voxel grids, because otherwise the fully-dense high-resolution condition will lead to out-of-memory.

https://doi.org/10.52202/079017-3099

Figure 7: **Text-2-Scene Generation.** Given a text prompt, we could generate various multi-view images and lift them to 3D scenes with SCube. See Appendix D.3 for more text-2-scene results.

Figure 8: **Effects of GAN Postprocessing.** Left: SCube+; Right: SCube.

Resolution	M	PSNR↑	LPIPS↓
256^3	4	18.58	0.62
1024^3	1	19.34	0.52
1024^3	4	19.34	0.48

Table 3: **Ablation Study for Appearance Reconstruction.**

predicts the sparse voxels and the appearance from images, we can only achieve a PSNR/LPIPS of 17.88/0.57, in comparison to 19.34/0.48 when using the two-stage model. Here the values are the average of $T + 5$ and $T + 10$ frames. In terms of geometry quality, the single-stage model is also significantly worse (up to 100×) in Chamfer distance than the two-stage model. Please refer to more details about the analysis of geometry quality in Appendix D.1.

Appearance Reconstruction. We validate the effect of voxel grid resolution and the number of Gaussians per voxel M in the appearance reconstruction stage. Results are shown in Tab. 3. We find that higher-resolution voxel grids are crucial for capturing detailed geometry, and using a larger number of Gaussians only slightly increases the performance. Thus, we set $M = 4$ as a moderate value for the final results. Compared in Fig. 8, the GAN-based postprocessing, despite the time cost, is beneficial for producing high-quality images by sharpening the renderings. See Appendix D.2 for more visual comparisons.

5 Discussion

Conclusion. In this work, we have introduced SCube, a feed-forward method for large 3D scene reconstruction from images. Given sparse view non-overlapping images, our method is able to predict a high-resolution 3D scene representation consisting of voxel-supported Gaussian splats (VoxSplat) and a light-weight sky panorama in a single forward pass within tens of seconds. We have demonstrated the effectiveness of our method on the Waymo Open Dataset, and have shown that our method outperforms the state-of-the-art methods in terms of reconstruction quality.

Limitations. Our method does suffer from some limitations. First, the current method is not able to handle complicated scenarios such as dynamic scenes under extreme lighting or weather conditions. Second, the quality of appearance in occluded regions still carries uncertainty. Third, the method itself still requires ground-truth 3D training data which is not always available for generic outdoor scenes. In future work, we plan to address these limitations by incorporating more advanced neural rendering techniques and by exploring more effective ways to generate training data.

https://doi.org/10.52202/079017-3099

References

[1] J. T. Barron, B. Mildenhall, D. Verbin, P. P. Srinivasan, and P. Hedman. Zip-nerf: Anti-aliased grid-based neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19697–19705, 2023.

[2] A. Blattmann, R. Rombach, H. Ling, T. Dockhorn, S. W. Kim, S. Fidler, and K. Kreis. Align your latents: High-resolution video synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22563–22575, 2023.

[3] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020.

[4] D. Charatan, S. Li, A. Tagliasacchi, and V. Sitzmann. pixelsplat: 3d gaussian splats from image pairs for scalable generalizable 3d reconstruction. *arXiv preprint arXiv:2312.12337*, 2023.

[5] Y. Chen, J. Wang, Z. Yang, S. Manivasagam, and R. Urtasun. G3r: Gradient guided generalizable reconstruction. In *ECCV 2024 Workshop on Wild 3D: 3D Modeling, Reconstruction, and Generation in the Wild*, 2024.

[6] Y. Chen, H. Xu, C. Zheng, B. Zhuang, M. Pollefeys, A. Geiger, T.-J. Cham, and J. Cai. Mvsplat: Efficient 3d gaussian splatting from sparse multi-view images. *arXiv preprint arXiv:2403.14627*, 2024.

[7] G. Chou, I. Chugunov, and F. Heide. Gensdf: Two-stage learning of generalizable signed distance functions. *Advances in Neural Information Processing Systems*, 35:24905–24919, 2022.

[8] T. Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023.

[9] K. Deng, A. Liu, J.-Y. Zhu, and D. Ramanan. Depth-supervised nerf: Fewer views and faster training for free. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12882–12891, 2022.

[10] W. Falcon and The PyTorch Lightning team. PyTorch Lightning, Mar. 2019.

[11] J. Gao, C. Gu, Y. Lin, H. Zhu, X. Cao, L. Zhang, and Y. Yao. Relightable 3d gaussian: Real-time point cloud relighting with brdf decomposition and ray tracing. *arXiv preprint arXiv:2311.16043*, 2023.

[12] J. Gao, T. Shen, Z. Wang, W. Chen, K. Yin, D. Li, O. Litany, Z. Gojcic, and S. Fidler. Get3d: A generative model of high quality 3d textured shapes learned from images. *Advances In Neural Information Processing Systems*, 35:31841–31854, 2022.

[13] T. Gieruc, M. Kästingschäfer, S. Bernhard, and M. Salzmann. 6img-to-3d: Few-image large-scale outdoor driving scene reconstruction. *arXiv preprint arXiv:2404.12378*, 2024.

[14] B. Graham and L. Van der Maaten. Submanifold sparse convolutional networks. *arXiv preprint arXiv:1706.01307*, 2017.

[15] J. Hasselgren, N. Hofmann, and J. Munkberg. Shape, light, and material decomposition from images using monte carlo rendering and denoising. *Advances in Neural Information Processing Systems*, 35:22856–22869, 2022.

[16] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[17] Y. Hong, K. Zhang, J. Gu, S. Bi, Y. Zhou, D. Liu, F. Liu, K. Sunkavalli, T. Bui, and H. Tan. Lrm: Large reconstruction model for single image to 3d. *arXiv preprint arXiv:2311.04400*, 2023.

[18] M. Hu, W. Yin, C. Zhang, Z. Cai, X. Long, H. Chen, K. Wang, G. Yu, C. Shen, and S. Shen. Metric3d v2: A versatile monocular geometric foundation model for zero-shot metric depth and surface normal estimation. *arXiv preprint arXiv:2404.15506*, 2024.

[19] S.-M. Hu, Z.-N. Liu, M.-H. Guo, J.-X. Cai, J. Huang, T.-J. Mu, and R. R. Martin. Subdivision-based mesh convolution networks. *ACM Transactions on Graphics (TOG)*, 41(3):1–16, 2022.

[20] S. Huang, Z. Gojcic, J. Huang, A. Wieser, and K. Schindler. Dynamic 3d scene analysis by point cloud accumulation. In *European Conference on Computer Vision*, pages 674–690. Springer, 2022.

 https://doi.org/10.52202/079017-3099

[21] Y. Huang, W. Zheng, Y. Zhang, J. Zhou, and J. Lu. Tri-perspective view for vision-based 3d semantic occupancy prediction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9223–9232, 2023.

[22] M. Z. Irshad, S. Zakharov, K. Liu, V. Guizilini, T. Kollar, A. Gaidon, Z. Kira, and R. Ambrus. Neo 360: Neural fields for sparse view synthesis of outdoor scenes. In *ICCV*, 2023.

[23] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4):1–14, 2023.

[24] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[25] X. Li, Y. Zhang, and X. Ye. Drivingdiffusion: Layout-guided multi-view driving scene video generation with latent diffusion model. *CoRR*, abs/2310.07771, 2023.

[26] Y. Li, Z. Yu, C. Choy, C. Xiao, J. M. Alvarez, S. Fidler, C. Feng, and A. Anandkumar. Voxformer: Sparse voxel transformer for camera-based 3d semantic scene completion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9087–9098, 2023.

[27] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[28] M. Liu, R. Shi, L. Chen, Z. Zhang, C. Xu, X. Wei, H. Chen, C. Zeng, J. Gu, and H. Su. One-2-3-45++: Fast single image to 3d objects with consistent multi-view generation and 3d diffusion. *arXiv preprint arXiv:2311.07885*, 2023.

[29] 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. *arXiv preprint arXiv:2405.12218*, 2, 2024.

[30] J. Lorraine, K. Xie, X. Zeng, C.-H. Lin, T. Takikawa, N. Sharp, T.-Y. Lin, M.-Y. Liu, S. Fidler, and J. Lucas. Att3d: Amortized text-to-3d object synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17946–17956, 2023.

[31] T. Lu, M. Yu, L. Xu, Y. Xiangli, L. Wang, D. Lin, and B. Dai. Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. *arXiv preprint arXiv:2312.00109*, 2023.

[32] 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*, 65(1):99–106, 2021.

[33] T. Müller, A. Evans, C. Schied, and A. Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM transactions on graphics (TOG)*, 41(4):1–15, 2022.

[34] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[35] G. Parmar, T. Park, S. Narasimhan, and J.-Y. Zhu. One-step image translation with text-to-image models. *arXiv preprint arXiv:2403.12036*, 2024.

[36] S. Peng, M. Niemeyer, L. Mescheder, M. Pollefeys, and A. Geiger. Convolutional occupancy networks. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*, pages 523–540. Springer, 2020.

[37] J. Philion and S. Fidler. Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIV 16*, pages 194–210. Springer, 2020.

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

[39] K. Ren, L. Jiang, T. Lu, M. Yu, L. Xu, Z. Ni, and B. Dai. Octree-gs: Towards consistent real-time rendering with lod-structured 3d gaussians. *arXiv preprint arXiv:2403.17898*, 2024.

[40] X. Ren, J. Huang, X. Zeng, K. Museth, S. Fidler, and F. Williams. Xcube: Large-scale 3d generative modeling using sparse voxel hierarchies. In *CVPR*, 2024.

[41] X. Ren, Z. Qian, and Q. Chen. Video deblurring by fitting to test data. *CoRR*, abs/2012.05228, 2020.

[42] X. Ren and X. Wang. Look outside the room: Synthesizing A consistent long-term 3d scene video from A single image. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 3553–3563. IEEE, 2022.

[43] B. Roessle, N. Müller, L. Porzi, S. R. Bulò, P. Kontschieder, and M. Nießner. Ganerf: Leveraging discriminators to optimize neural radiance fields. *ACM Transactions on Graphics (TOG)*, 42(6):1–14, 2023.

[44] J. L. Schonberger and J.-M. Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4104–4113, 2016.

[45] T. R. Shaham, T. Dekel, and T. Michaeli. Singan: Learning a generative model from a single natural image. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4570–4580, 2019.

[46] T. Shen, J. Munkberg, J. Hasselgren, K. Yin, Z. Wang, W. Chen, Z. Gojcic, S. Fidler, N. Sharp, and J. Gao. Flexible isosurface extraction for gradient-based mesh optimization. *ACM Transactions on Graphics (TOG)*, 42(4):1–16, 2023.

[47] Y. Shi, P. Wang, J. Ye, M. Long, K. Li, and X. Yang. Mvdream: Multi-view diffusion for 3d generation. *arXiv preprint arXiv:2308.16512*, 2023.

[48] J. Shriram, A. Trevithick, L. Liu, and R. Ramamoorthi. Realmdreamer: Text-driven 3d scene generation with inpainting and depth diffusion. *arXiv preprint arXiv:2404.07199*, 2024.

[49] V. Sitzmann, E. Chan, R. Tucker, N. Snavely, and G. Wetzstein. Metasdf: Meta-learning signed distance functions. *Advances in Neural Information Processing Systems*, 33:10136–10147, 2020.

[50] V. Sitzmann, J. Thies, F. Heide, M. Nießner, G. Wetzstein, and M. Zollhofer. Deepvoxels: Learning persistent 3d feature embeddings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2437–2446, 2019.

[51] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser. Semantic scene completion from a single depth image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1746–1754, 2017.

[52] J. Sun, Y. Xie, L. Chen, X. Zhou, and H. Bao. Neuralrecon: Real-time coherent 3d reconstruction from monocular video. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15598–15607, 2021.

[53] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2446–2454, 2020.

[54] H. Tang, Z. Liu, X. Li, Y. Lin, and S. Han. Torchsparse: Efficient point cloud inference engine. *Proceedings of Machine Learning and Systems*, 4:302–315, 2022.

[55] D. Tochilkin, D. Pankratz, Z. Liu, Z. Huang, A. Letts, Y. Li, D. Liang, C. Laforte, V. Jampani, and Y.-P. Cao. Triposr: Fast 3d object reconstruction from a single image. *arXiv preprint arXiv:2403.02151*, 2024.

[56] V. Voleti, C.-H. Yao, M. Boss, A. Letts, D. Pankratz, D. Tochilkin, C. Laforte, R. Rombach, and V. Jampani. Sv3d: Novel multi-view synthesis and 3d generation from a single image using latent video diffusion. *arXiv preprint arXiv:2403.12008*, 2024.

[57] S. Wang, V. Leroy, Y. Cabon, B. Chidlovskii, and J. Revaud. Dust3r: Geometric 3d vision made easy. *arXiv preprint arXiv:2312.14132*, 2023.

[58] W. Wang, Q. Lv, W. Yu, W. Hong, J. Qi, Y. Wang, J. Ji, Z. Yang, L. Zhao, X. Song, et al. Cogvlm: Visual expert for pretrained language models. *arXiv preprint arXiv:2311.03079*, 2023.

[59] 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.

[60] Z. Wang, C. Lu, Y. Wang, F. Bao, C. Li, H. Su, and J. Zhu. Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. *Advances in Neural Information Processing Systems*, 36, 2024.

[61] Y. Wei, L. Zhao, W. Zheng, Z. Zhu, J. Zhou, and J. Lu. Surroundocc: Multi-camera 3d occupancy prediction for autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 21729–21740, 2023.

 https://doi.org/10.52202/079017-3099

[62] F. Williams, J. Huang, J. Swartz, G. Klar, V. Thakkar, M. Cong, X. Ren, R. Li, C. Fuji-Tsang, S. Fidler, E. Sifakis, and K. Museth. fvdb: A deep-learning framework for sparse, large-scale, and high-performance spatial intelligence. *ACM Transactions on Graphics (TOG)*, 43(4):133:1–133:15, 2024.

[63] R. Wu, B. Mildenhall, P. Henzler, K. Park, R. Gao, D. Watson, P. P. Srinivasan, D. Verbin, J. T. Barron, B. Poole, et al. Reconfusion: 3d reconstruction with diffusion priors. *arXiv preprint arXiv:2312.02981*, 2023.

[64] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez, and P. Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. In *Neural Information Processing Systems (NeurIPS)*, 2021.

[65] Y. Xu, H. Tan, F. Luan, S. Bi, P. Wang, J. Li, Z. Shi, K. Sunkavalli, G. Wetzstein, Z. Xu, et al. Dmv3d: Denoising multi-view diffusion using 3d large reconstruction model. *arXiv preprint arXiv:2311.09217*, 2023.

[66] Y. Yan, H. Lin, C. Zhou, W. Wang, H. Sun, K. Zhan, X. Lang, X. Zhou, and S. Peng. Street gaussians for modeling dynamic urban scenes. *arXiv preprint arXiv:2401.01339*, 2024.

[67] Z. Ye, W. Li, S. Liu, P. Qiao, and Y. Dou. Absgs: Recovering fine details for 3d gaussian splatting. *arXiv preprint arXiv:2404.10484*, 2024.

[68] A. Yu, V. Ye, M. Tancik, and A. Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4578–4587, 2021.

[69] X. Yu, Y.-C. Guo, Y. Li, D. Liang, S.-H. Zhang, and X. Qi. Text-to-3d with classifier score distillation. *arXiv preprint arXiv:2310.19415*, 2023.

[70] Z. Yu, S. Peng, M. Niemeyer, T. Sattler, and A. Geiger. Monosdf: Exploring monocular geometric cues for neural implicit surface reconstruction. *Advances in neural information processing systems*, 35:25018–25032, 2022.

[71] Z. Yu, T. Sattler, and A. Geiger. Gaussian opacity fields: Efficient and compact surface reconstruction in unbounded scenes. *arXiv preprint arXiv:2404.10772*, 2024.

[72] F. Zhang, Y. Zhang, Q. Zheng, R. Ma, W. Hua, H. Bao, W. Xu, and C. Zou. 3d-scenedreamer: Text-driven 3d-consistent scene generation. *arXiv preprint arXiv:2403.09439*, 2024.

[73] K. Zhang, S. Bi, H. Tan, Y. Xiangli, N. Zhao, K. Sunkavalli, and Z. Xu. Gs-lrm: Large reconstruction model for 3d gaussian splatting. *arXiv preprint arXiv:2404.19702*, 2024.

[74] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

[75] X.-Y. Zheng, H. Pan, P.-S. Wang, X. Tong, Y. Liu, and H.-Y. Shum. Locally attentional sdf diffusion for controllable 3d shape generation. *ACM Transactions on Graphics (TOG)*, 42(4):1–13, 2023.

[76] X. Zhou, Z. Lin, X. Shan, Y. Wang, D. Sun, and M.-H. Yang. Drivinggaussian: Composite gaussian splatting for surrounding dynamic autonomous driving scenes. *arXiv preprint arXiv:2312.07920*, 2023.

[77] V. Zyrianov, H. Che, Z. Liu, and S. Wang. Lidardm: Generative lidar simulation in a generated world. *arXiv preprint arXiv:2404.02903*, 2024.

– Appendix –

A Implementation Details

Additional Data Processing Details. For each data sample, we crop the point cloud obtained from § 4.1 into a local chunk of $102.4\text{m} \times 102.4\text{m}$. The point cloud is then voxelized into the fine-level and coarse-level grids used in § 3.1 with 1024^3 and 256^3 resolutions respectively (with voxel sizes of 0.1m and 0.4m). Our dataset contains 20243 chunks for training and 5380 chunks for evaluation, out of the 798 training and 202 validation sequences.

Input and Evaluation Details. Waymo dataset provides 5 views for each camera frame, namely `front`, `front-left`, `front-right`, `side-left` and `side-right`. However, not all of the baseline methods we compared with in § 4.2 can handle the unconventional camera intrinsic in the `side-left` and `side-right` views. We hence only use the first three views (with a resolution of 1920×1280) in § 4.2 for both the input and the evaluation metrics. However, in § 4.3 we opt to use all 5 views for the input to both our method and the baseline due to compatibility and maximized performance.

For the baselines, the original PixelSplat [4] method does not have depth supervision. To make the comparison fair, we attempt to add a depth supervision loss to it. However, the experimental result shows that the additional loss hurts the performance as shown in Tab. 4. We thus report the results of vanilla PixelSplat in the main paper.

	$T+5$			$T+10$		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
PixelSplat [4]	**20.11**	**0.70**	**0.60**	18.77	**0.66**	**0.62**
PixelSplat [4] w/ Depth Supervision	19.91	0.58	0.66	**18.87**	0.56	0.67

Table 4: **Comparison of PixelSplat and PixelSplat with Depth Supervision.**

Training Details. The diffusion loss in Eq (2) is defined similar to [16, 40] with a $\boldsymbol{v}$-parametrization as:

$$\mathcal{L}_{\text{Diffusion}} = \mathbb{E}_{t,\mathbf{X},\boldsymbol{\epsilon}\sim\mathcal{N}(0,I)}\left[\left\|\boldsymbol{v}(\sqrt{\bar{\alpha}_t}\mathbf{X} + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) - (\sqrt{\bar{\alpha}_t}\boldsymbol{\epsilon} - \sqrt{1-\bar{\alpha}_t}\mathbf{X})\right\|_2^2\right], \tag{7}$$

where $\boldsymbol{v}(\cdot)$ is the diffusion network, t is the randomly sampled diffusion timestamp, and $\bar{\alpha}_t$ is the scheduling factor for the diffusion process, whose details are referred to in [16].

We train all of our models using the Adam [24] optimizer with $\beta_1 = 0.9$ and $\beta_1 = 0.999$. We use PyTorch Lightning [10] for building our distributed training framework. For the voxel grid reconstruction stage, we train both coarse-level and fine-level voxel latent diffusion models with $64\times$ NVIDIA Tesla A100s for 2 days. For the appearance reconstruction model, we train it using $8\times$ NVIDIA Tesla A100s for 2 days. Empirically, we use $\lambda = 1.0$ for $\mathcal{L}_{\text{Depth}}$ in Eq (2). Additionally, we use $\lambda_1 = 0.9$, $\lambda_2 = 1.0$, $\lambda_{\text{SSIM}} = 0.1$ and $\lambda_{\text{LPIPS}} = 0.6$ in Eq (6). For image condition, we set the feature channel $C = 32$, the number of depth bins $D = 64$, $z_{\text{near}} = 0.1$ and $z_{\text{far}} = 90.0$. We linearly increase the interval of depth bins.

B Network Architecture

Voxel Grid Reconstruction. We follow [40] to implement the Sparse Structure VAE and the Diffusion UNet using the sparse 3D deep learning framework fVDB [62]. Hyperparameters for training them are listed in Tab. 5 and Tab. 6. We pass the images to distilled DINO-v2 [34] ViT-B/14. We use four 2D convolutional layers (channel dims: $[768, 256, 256, 32, 32]$, kernel size: 3, stride: 1) to further process the DINO-v2 output to predict the image feature and the depth distribution.

Appearance Reconstruction. We process the original input images with three 2D convolutional layers (channel dims: $[3, 16, 32, 32]$, kernel size: 3, stride: $[1, 1, 2]$). For the last two convolutional

https://doi.org/10.52202/079017-3099

	Waymo $64^3 \rightarrow 256^3$	Waymo $256^3 \rightarrow 1024^3$
Model Size	14.9M	3.8M
Base Channels	64	32
Channels Multiple	1,2,4	1,2,4
Latent Dim	8	8
Batch Size	32	32
Epochs	50	50
Learning Rate	1e-4	

Table 5: **Hyperparameters for VAE.**

	Waymo - 64^3	Waymo - 256^3
Diffusion Steps	1000	
Noise Schedule	linear	
Model Size	728M	83.0M
Base Channels	192	64
Depth	2	
Channels Multiple	1,2,4,4	1,2,2,4
Heads	8	
Attention Resolution	16	32
Dropout	0.0	0.0
Batch Size	512	256
Iterations	40K	20K
Learning Rate	5e-5	

Table 6: **Hyperparameters for voxel latent diffusion models.**

layers, we set the residual connections. We additionally positionally encode each voxel and then concatenate the positional encoding [32] of each voxel with the corresponding voxel feature after ray casting. We then apply a 3D sparse UNet to output per-Gaussian parameters. We use GT voxels in appearance reconstruction training. Hyperparameters of this 3D sparse UNet are listed Tab. 7.

Model Size	Base Channels	Channels Multiple	Batch Size	Epochs	Learning Rate
4.3M	32	[1, 2, 4]	32	15	1e-4

Table 7: **Hyperparameters for 3D sparse UNet in appearance reconstruction stage.**

Sky Panorama for Background. For the sky panorama model, we set $H_p = 768, W_p = 1536$ in the training stage and increase $H_p = 1024, W_p = 2048$ in the inference time. To decode sampled sky features into the RGB image, we utilize a 2D CNN network reducing the channel from 32 to 16 to 3 with stride 1, keeping the spatial resolution unchanged.

C SCube+ without Per-scene Training

In § 3.3 we introduce a GAN postprocessing module to refine the rendered images, which is finetuned on each scene. To further improve the efficiency of our method, we hereby present a postprocessing module that is jointly trained on the full dataset, without the need of per-scene finetuning. Specifically, we replace the original GAN with a pix2pix-turbo model [35] (which we denote as SCube+*) and train it with image pairs inferred from our model and the ground truths. The results are shown in Fig. 9. This improved model not only reduces the voxel block artifacts, but also resolve the ISP inconsistencies within the image. After enabling this module, the FPS drops from 138 to 20 but can still be visualized interactively.

	Reconstruction (T)			Prediction ($T+5$)			Prediction ($T+10$)		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
SCube	<u>25.90</u>	<u>0.77</u>	0.45	19.90	<u>0.72</u>	0.47	18.78	<u>0.70</u>	0.49
SCube+	**28.01**	**0.81**	**0.25**	**22.32**	**0.74**	**0.34**	**21.09**	**0.72**	**0.38**
SCube+*	22.59	0.68	<u>0.38</u>	<u>20.37</u>	0.66	<u>0.41</u>	<u>19.65</u>	0.65	<u>0.42</u>

Table 8: **Quantitative Comparisons on 3D Reconstruction.** The metrics are computed both at the input frame T and future frames. ↑: higher is better, ↓: lower is better.

SCube SCube+* SCube SCube+* SCube SCube+*

Figure 9: **SCube+*.** Results from the postprocessing network without per-scene optimization. White-balance inconsistencies from different views (marked in red box) can be fixed.

D Additional Results

In this section, we provide more qualitative results on all datasets. We additionally provide a **supplementary video** in the accompanying files to better illustrate our results.

D.1 Geometry Quality

We note that the uncertainty of the scene geometry given our input images is large, and the problem that the model tackles is indeed non-trivial and sometimes even ill-posed. To demonstrate this, we compute the percentage of occluded voxels (that are invisible from the input images) w.r.t. all the ground-truth voxels, and the number is around 80%. To quantitatively evaluate the geometry quality, we compute an additional metric called 'voxel Chamfer distance' that measures the L2-Chamfer distance between the predicted voxels and ground-truth voxels (that are pixel-aligned), divided by the voxel size. This metric reflects the geometric accuracy of our prediction by measuring on average how many voxels is the prediction apart from the ground truth. The results on Waymo Open Dataset are shown in Tab. 9.

Quantile	0.5 *(median)*	0.6	0.7	0.8	0.9
Ours	0.26	0.28	0.32	0.37	0.51

Table 9: **Geometry Quality Comparison.** We show the voxel Chamfer distance comparison between our two-stage model and a single-stage non-diffusion model.

Tab. 9 indicates that on 90% of the test samples, the predicted voxel grid is only half of a voxel off from the ground truth. We note that during our data curation process, there could be errors in the ground-truth voxels (*e.g.*, due to COLMAP failures), accounting for the outliers in the above metric. In the meantime, we visualize the sample with the worst voxel Chamfer distance in Fig. 10. The predicted results are decent even though the ground truth is corrupted due to the lack of motion in the ego car. This demonstrates the robustness of our method.

D.2 Visual Ablation Study

In addition to the quantitative ablation study in Tab. 3, we present a qualitative demonstration in Fig. 11. For the single-stage model, we test the upper bound of it by feeding the ground-truth 1024^3 voxel grids because otherwise the fully-dense high-resolution condition will lead to out-of-memory. The qualitative results match the numbers, showing the importance of using higher-resolution voxel grids and the two-stage model.

https://doi.org/10.52202/079017-3099

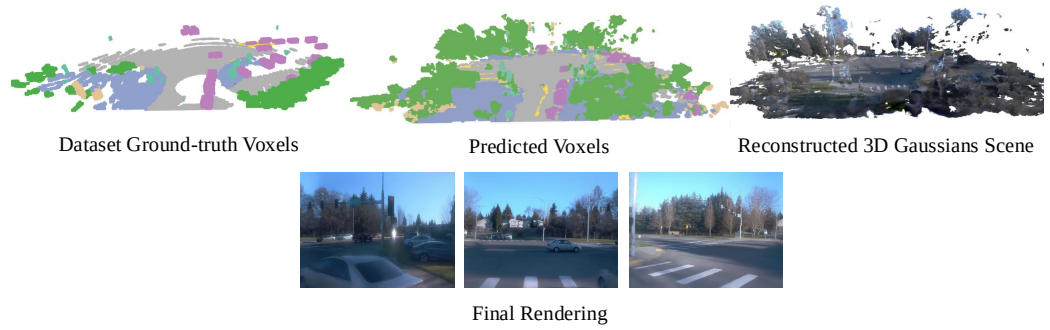


Figure 10: Result on the data sample with the worst voxel Chamfer distance. We show geometry reconstruction and the image renderings.

Figure 11: **Visual Ablation Study.** (a) SCube+ (b) SCube (c) SCube with a 256^3 resolution input grid (d) Single-stage model. Zoom in for a better view.

https://doi.org/10.52202/079017-3099

D.3 Additional Results on Text-2-Scene Generation

We provide additional text-2-scene generation results in Fig. 12 and Fig. 13.

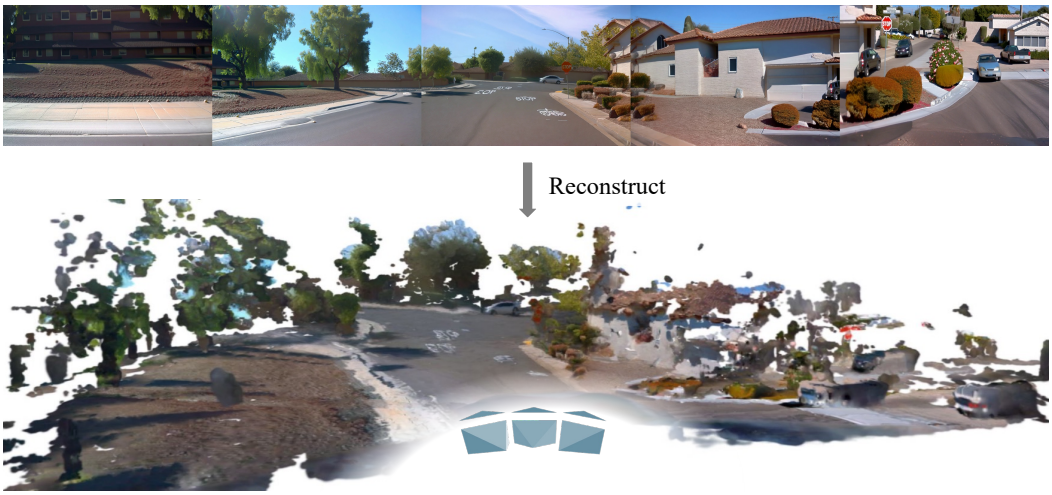
A residential neighborhood features houses with well-maintained gardens, autumn-colored trees, lawns with scattered leaves, parked cars, driveways, and clear blue skies.

A residential area features multiple houses, some with specific decorations and vehicles parked outside, including a white pickup truck and a gray car, along with various greenery and utility elements.

Figure 12: **More Text-2-Scene Generation.** The generated multi-view images may contain flaws, while SCube is still able to reconstruct the 3D scenes.

https://doi.org/10.52202/079017-3099

A suburban neighborhood features two-story houses with reddish-brown roofs and beige walls, marked roads, various parked vehicles, stop signs, and a mixture of gravel, rocks, and trees providing shade on a sunny day.



A suburban neighborhood features a park with green trees, residential houses with red-tiled roofs, streets with bike lane signs and white markings, well-maintained lawns, and sidewalks.

Figure 13: **More Text-2-Scene Generation.**

https://doi.org/10.52202/079017-3099

D.4 Additional Results on Large-scale Scene Reconstruction

We provide additional results on large-scale scene reconstruction from real-world captures in Fig. 14.

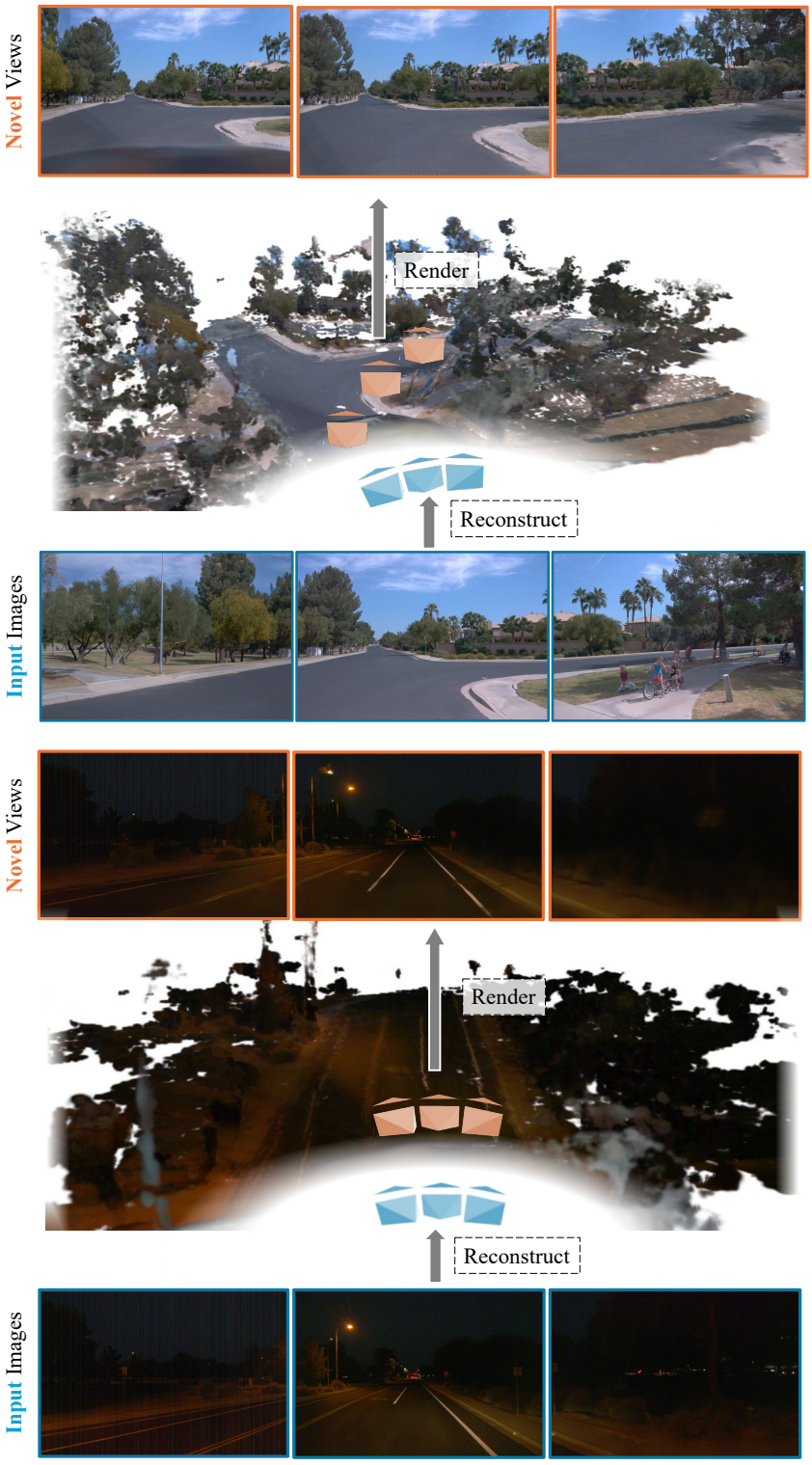


Figure 14: **More Novel View Synthesis.** Our method is able to synthesis extreme novel views.

https://doi.org/10.52202/079017-3099

D.5 Additional Results on LiDAR Simulation

We provide additional LiDAR Simulation results in Fig. 15. We also show the result on a long sequence input in Fig. 16.

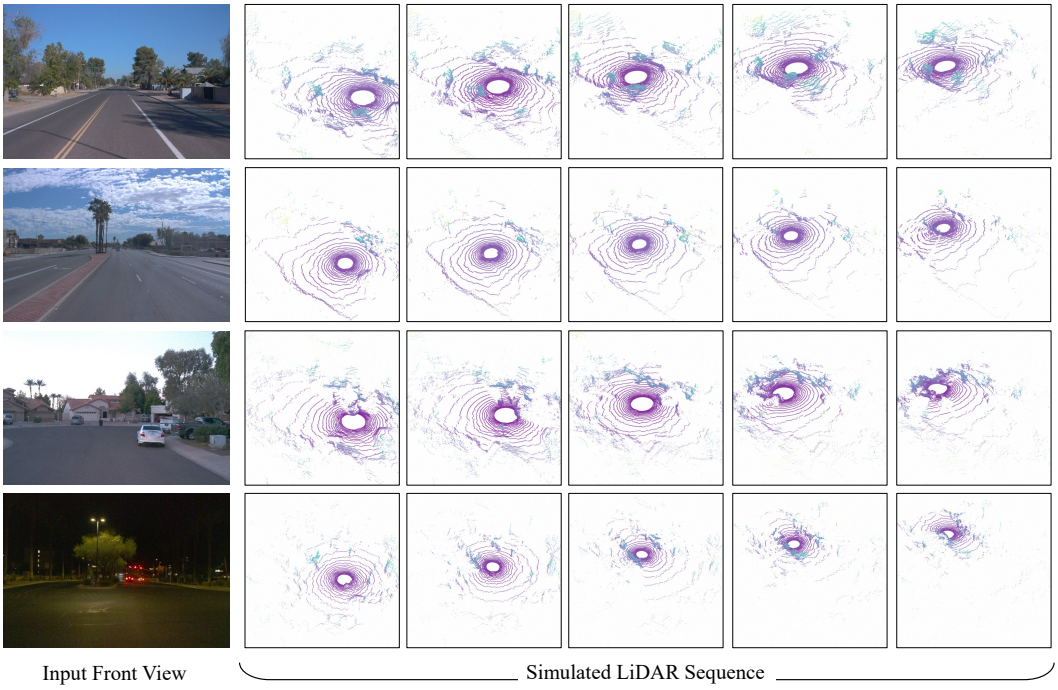


Figure 15: **More LiDAR Simulation results.**

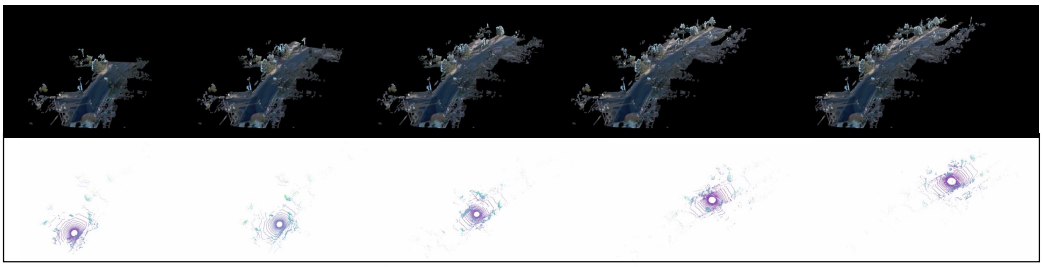


Figure 16: **SCube with Long Sequence Input.** Up: reconstructed scene with appearance. Down: LiDAR simulation result. We chunk the long sequence into clips and apply out method iteratively.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes] .

   Justification: We make three claims: (1) We are the first feed-forward model to directly reconstruct GSplats grounded in a predicted 3D space. To the best of our knowledge, we are not aware of other work in the literature doing this. (2) We obtain a full 3D reconstruction from which we can render novel views in a single forward pass. We show many instances of this in the experiments section (both novel views, semantic reconstruction, and LiDAR resimulation). (3) We are SOTA when compared to baseline methods which perform reconstruction from sparse views. We compare against a wide variety of baselines and demonstrate qualitative and quantitative improvements over these. (4) Our inference pipeline takes under 20 seconds. In the meantime, we simply report this as fact in the paper.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes] .

   Justification: Section 5 includes an explicit limitations section. We remark on difficulties with dynamic scenes and out-of-distribution weather and lighting conditions, challenges reconstructing high-quality appearance in highly occluded regions, and training data limitations. See this section for more details.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.

https://doi.org/10.52202/079017-3099

- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: NA

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We describe our method in great detail and provide enough information that a researcher familiar with diffusion models and neural rendering should be able to reproduce the pipeline. We describe our experiments in detail as well and run them on Waymo Open which is a public dataset accessible to all. The appendix includes detailed training and network architecture information. Together, these should make our results fully reproducible.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.

(b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.

(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: Due to institutional constraints, we are not able to release the code until the paper is fully accepted. Upon acceptance, we will release all code and data required to reproduce this work.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes] .

   Justification: See descrition in Section 4 and appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

https://doi.org/10.52202/079017-3099

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: It is not clear what assumptions should be made on error distributions. Unfortunately, in this particular literature, error bars and confidence intervals are not typically reported.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We report these explicitly in the experiments section

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: We have reviewed the code of ethics and believe our paper fully conforms to it.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.

- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

    Justification: We feel there are no direct negative societal impacts of this work as it is a foundational method for reconstruction. While such a foundational method could eventually be used in pipelines with potential negative consequences (e.g. offensive applications in military settings), we do not believe our method directly enables this. Furthermore, by training on an open driving dataset, we focus simply on reconstructing the 3D world from car footage, which alone cannot be used to negatively impact society.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: We don't believe our model poses an immediate risk of misuse. We believe it will help advance research in 3D reconstruction from images.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

https://doi.org/10.52202/079017-3099

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite relevant related works and datasets. All image and video assets in the paper and supplementary material are our own.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: We do not release new assets alongside the paper.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: We did no crowdsourcing or human subject experiments.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: We did not study on human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

https://doi.org/10.52202/079017-3099