ProtoCar: Learning 3D Vehicle Prototypes from Single-View and Unconstrained Driving Scene Images

Hongyuan Liu, Haochen Yu, Bochao Zou*, Juntao Lyu, Qi Mei, Jiansheng Chen, Huimin Ma*

School of Computer and Communication Engineering, University of Science and Technology Beijing, China
{hongyuanliu,haochen.yu,lyujuntao,qimei}@xs.ustb.edu.cn,{zoubochao,jschen,mhmpub}@ustb.edu.cn

Abstract

Reconstructing 3D models from sensor data is a valuable and promising direction for developing testing and validation environments in applications like autonomous driving. However, existing methods for 3D modeling often rely on extensive multi-view data or controlled conditions, making them difficult and expensive to scale. Furthermore, these methods, particularly those based on neural radiance fields, typically produce implicit models that can be challenging to manipulate and suffer from slow rendering speeds. In this paper, we introduce ProtoCar, a novel approach that overcomes these limitations by learning 3D vehicle prototypes from single-view images with diverse and unconstrained visual conditions. ProtoCar uses real-world driving data from LiDAR and image sensors, and employs 3D Gaussian splatting techniques to represent explicit geometric and texture. Extensive experiments demonstrate that ProtoCar generates high-quality 3D models and adapts well to various vehicle types and challenging visual scenarios, offering a scalable and effective solution for 3D modeling in environments with limited and variable visual information.

Figure 1: Driving scene data is often diverse, characterized by varying vehicle types, lighting conditions, and levels of occlusion. Our method aims to address the limitations of existing approaches by enabling the generation of unconstrained single-image to 3D vehicle models using only low-quality real-world data.

Introduction

Perceiving and understanding the 3D world is foundational for many computer vision applications, such as autonomous driving, augmented reality (AR), and robotics. As a crucial tool, simulation offers safe and controlled environments for testing and validation. However, the construction of 3D models necessary for these simulations often requires significant manual effort and expertise (Dosovitskiy et al. 2017a; Cabon, Murray, and Humenberger 2020; Richter et al. 2016a).

Rapid and convenient acquisition of 3D assets has become a rapidly evolving area of research. Recent advancements have led to the development of techniques such as text-to-3D (Poole et al. 2022; Lin et al. 2023) and image-to-3D (Liu et al. 2023, 2024). These methods promise to automate the creation of 3D models from 2D images or text, significantly reducing the manual workload involved in traditional approaches. These methods leverage different forms of 3D representations, such as point clouds(Nichol et al. 2022), voxels(Rukhovich, Vorontsova, and Konushin 2022), and the widely used neural radiance fields (Mildenhall et al. 2021; Jun and Nichol 2023). However, due to the implicit representation and volumetric rendering process, methods based on NeRF often consume significant computational resources and are challenging for real-time rendering. Recently, 3D Gaussian splatting(Kerbl et al. 2023) has achieved high-quality and efficient 3D explicit representation using anisotropic 3D Gaussians.

Due to the reliance on synthetic datasets, the style of the generated models is often limited to a cartoonish aesthetic, resulting in an unrealistic appearance. For applications that demand a high degree of realism, such as safety-critical autonomous driving, this limitation is particularly problematic. Moreover, many image-to-3D models are trained on extensive, high-quality datasets(Deitke et al. 2024; Downs et al. 2022). These methods mostly require high-resolution multi-view images of the object, while also demanding non-occlusion, carefully designed camera parameters. As shown in Figure 1, real-world scenarios introduce challenges such as occlusions and diverse viewpoints, which are often not adequately represented in these datasets. This gap leads to

*Corresponding author

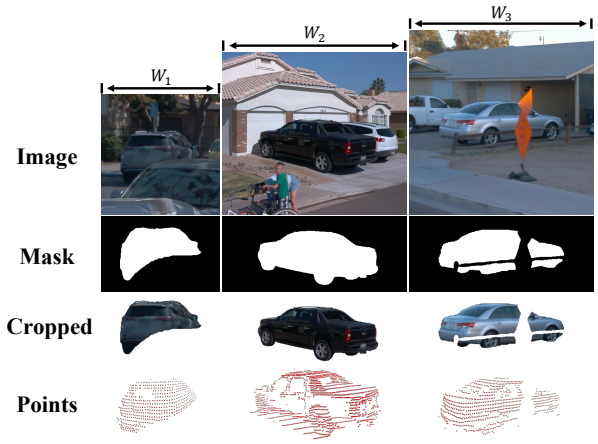

Figure 2: Due to environmental variations and limitations of data acquisition devices, datasets collected from real-world scenes often contain a significant amount of low-quality data, such as occluded images and sparse point clouds. Effectively utilizing these data could potentially address the current methods' reliance on high-quality datasets.

significant performance degradation when models trained in controlled environments are deployed in the wild. Learning to generate realistic 3D models from vast amounts of unconstrained real-world data remains an underexplored yet valuable research direction.

In this work, we propose an unconstrained image-to-3D model, ProtoCar, focusing on learning the prototype of a car using real images captured from driving scenes, as show in Figure 2. This task is more practical and challenging compared to conventional image-to-3D tasks. During training, we do not have access to any surrounding views of a car and face severe variations in lighting, camera parameters, and occlusion levels.

We employ Fibonacci sphere sampling to obtain the initial Gaussian point cloud distribution. By optimizing the initial template, we obtain an explicit representation of the initial geometric shape. Without the need for additional semantic segmentation model supervision, we utilize the imaging relationship between point clouds and images to provide shape supervision. We use cross-attention to fuse 3D explicit features and 2D implicit features. By leveraging efficient 3D spatial representation in tri-plane, we encode complex Gaussian texture features while aligning visible image 2D features with the encode tri-plane 3D features, enhancing the network's ability to generate and complete geometry. Finally, we achieve fast and efficient rendering results using Gaussian splatting. We summarize our contributions as follows:

- We propose a new image-to-3D framework, ProtoCar, which utilizes Gaussian spheres as the representation form to effectively learn from outdoor driving scene data and achieve unconstrained reconstruction.
- We introduce an efficient Gaussian point cloud generation method and achieve rapid, unconstrained image-to-initial 3D Gaussian point cloud generation.
- We develop a self-supervised approach based on 3D projection for shape and feature learning. This method effectively distinguishes object and non-object pixel regions and aligns 3D and 2D features.

Related Work

Single-View 3D Reconstruction

Single-view 3D reconstruction is a problem in computer vision, aimed at reconstructing 3D geometry from a single 2D image. In recent years, deep learning-based single-view 3D reconstruction methods garner widespread attention and made substantial progress. The emergence of large-scale high-quality simulation datasets (Chang et al. 2015; Deitke et al. 2024) and scan datasets (Dai et al. 2017) provide valuable 3D prior knowledge for image-to-3D methods. Earlier approaches explore various 3D representation forms to align with neural networks for improved reconstruction results, such as meshes (Kanazawa et al. 2018; Gkioxari, Malik, and Johnson 2019), Occupancy Networks (Mescheder et al. 2019), signed distance fields (Engelmann, Stückler, and Leibe 2017; Kundu, Li, and Rehg 2018), and implicit models (Mildenhall et al. 2021; Zakharov et al. 2020). With differentiable rendering, 3D reconstruction quality can be supervised using only 2D images.

Benefiting from 2D generation methods like DALL-E (Ramesh et al. 2021) and Stable Diffusion (Rombach et al. 2022), many methods employ score distillation sampling strategies (Poole et al. 2022) to leverage large-scale pre-trained 2D generative models for conditional 3D model generation. Further, Liu et al. (2023, 2024) use view-conditioned pre-trained diffusion models to generate 3D models from single images. Recently, TGS (Zou et al. 2024) achieves single-view reconstruction of 3D Gaussian models. However, these methods either rely on large-scale training of 2D generative models with multiple iterations or depend on extensive high-quality 3D simulation datasets.

Differentiable Rendering

Differentiable rendering is a critical technique in computer vision, which involves generating 2D views from 3D models, computing losses from these views, and enabling gradient backpropagation to optimize the model. Among the most prominent implicit models, NeRF (Mildenhall et al. 2021) gains significant attention and has been widely applied in various domains, such as novel view synthesis (Yu et al. 2021a; Rematas, Martin-Brualla, and Ferrari 2021) and 3D-aware image synthesis (Niemeyer and Geiger 2021; Schwarz et al. 2020). Numerous works achieve NeRF-based 3D generation; for example, EG3D(Chan et al. 2022) introduce a hybrid triplane feature representation to reduce the high computational cost associated with high-resolution images. This simple yet effective representation is widely adopted in subsequent works (Skorokhodov et al. 2023; Shen et al. 2023; Zou et al. 2024). Despite many efforts to accelerate neural rendering of implicit representations (Garbin et al. 2021; Müller et al. 2022b), the computational efficiency remains lower than rasterizable rendering methods due to the volumetric nature.

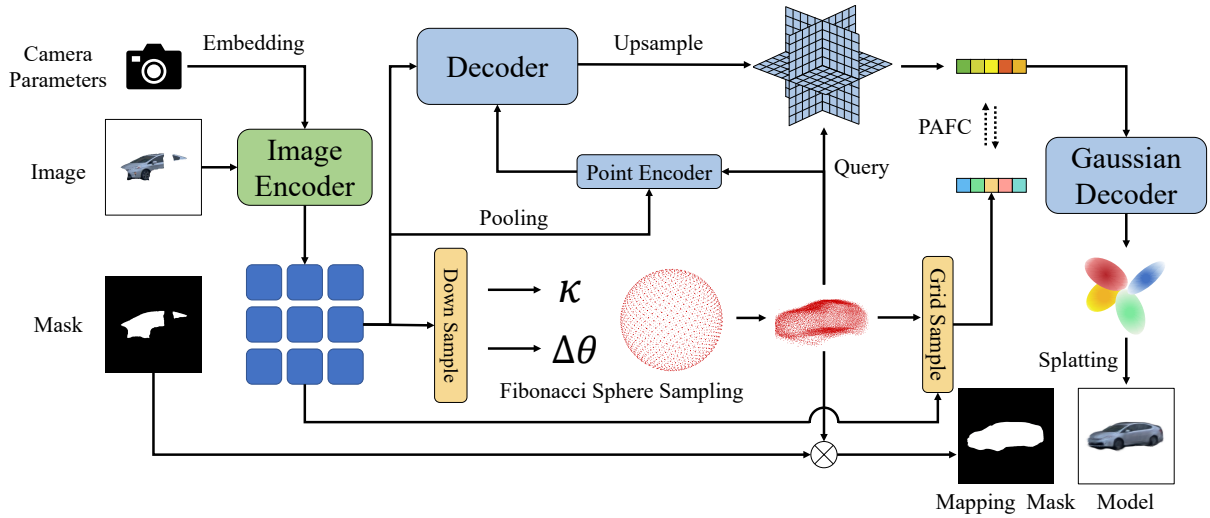

Figure 3: The structural diagram of the ProtoCar network. For the input image, we first encode it to obtain image features. Then, using Fibonacci sphere sampling and a two-degree-of-freedom spherical coordinate system, we decode the initial Gaussian point cloud. Subsequently, we integrate the spatial priors of the Gaussian point cloud and use intermediate feature representations of tri-plane to decode the attributes of the 3D Gaussians. Additionally, we apply Projection-Aware Feature Consistency to constrain the feature space and use Mapping mask to constrain the pixel space.

3D Gaussian Splatting (Kerbl et al. 2023), which represents 3D objects using 3D Gaussian ellipsoids, is the latest advancements in neural rendering. As an explicit representation, it enables high-quality real-time rendering via rasterization. However, unlike neural radiance fields, the unstructured representation format of 3D Gaussians makes it challenging to integrate with various networks (Zou et al. 2024).

Simulation in Autonomous Driving

The safety of autonomous driving algorithms still requires extensive testing within simulated environments for validation. Traditional graphics rendering relies on manual artistic creation to develop all elements within a scene and utilizes a rendering pipeline to present a virtual driving environment (Dosovitskiy et al. 2017b; Richter et al. 2016b). However, manual modeling methods result in high production costs and low realism.

Recently, data-driven simulation approaches have gained widespread attention. For instance, NSG (Ost et al. 2021) combines scene graphs with neural radiance fields and, successfully separates the background from dynamic foreground vehicles. Unisim (Yang et al. 2023) employs additional sensor data, to further improve rendering quality.

Following the introduction of 3D Gaussian splatting, several methods (Yan et al. 2024; Zhou et al. 2024) attempt to adopt a similar approach to NSG (Ost et al. 2021), achieving faster rendering speeds. However, these neural rendering-based simulators often ensure the reconstruction quality of foreground objects only from visible viewpoints, and they struggle to handle complex situations like occlusions, limiting their usability. Although some neural radiance field-based methods (Müller et al. 2022a; Shen et al. 2023) attempt to address this drawback, they suffer from poor rendering quality and slow processing speeds, making them difficult to apply in practice.

Method

Initial Gaussian Point Cloud Generating

Fibonacci Sphere Sampling The quality of the initialized Gaussian point cloud has a significant impact on the subsequent rendering process. In this work, we explore an efficient and rapid method for point cloud initialization and training. We first employ Fibonacci sphere sampling to obtain a point cloud that is as uniformly distributed as possible on a unit sphere. Let n denote the total number of points to be sampled.

For each point i where i ranges from 0 to $n-1$, the Cartesian coordinates (x_i, y_i, z_i) of the point are derived from:

$$\begin{cases} y_i = 1 - \dfrac{2i}{n-1} \\ x_i = \cos(\phi \times i) \times \sqrt{1 - y_i^2} \\ z_i = \sin(\phi \times i) \times \sqrt{1 - y_i^2} \end{cases} \tag{1}$$

The golden angle ϕ is computed as:

$$\phi = \pi \times (3 - \sqrt{5}) \tag{2}$$

By employing these formulas, points are distributed uniformly on the sphere, providing a balanced sampling across the entire surface, as illustrated in Figure 4.

Two Degree of Freedom Optimization Unlike traditional three-degree-of-freedom point clouds, our work employs two-degree-of-freedom point clouds, specifically using k and $\Delta\theta$, to address issues of convergence difficulties due

to excessive degrees of freedom and sparse, uneven distribution due to insufficient degrees of freedom. As shown in Figure 4, starting with a uniformly distributed point cloud on a unit sphere, the scaling factor k controls the collapse ratio of the point cloud towards the origin along the radial direction. The decoder generates different scaling factors for point clouds at different spatial locations, transforming the uniformly distributed points on the unit sphere into a distribution that conforms to the surface of specific model types. To ensure the efficient utilization of the point cloud and to prevent the appearance of internal points that do not contribute to the model's shape or texture, we add a regularization term to the scaling coefficient k:

$$\mathcal{L}_{scale} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{k_i} \tag{3}$$

The fixed spherical initialization method leads to uneven point cloud distribution, with denser point clouds in areas where the k values are smaller. To address this, we allow each point to adjust freely in the azimuthal direction θ, enabling the generated point cloud to be uniformly distributed across the object's surface, which is beneficial for subsequent texture learning.

Finally, the Cartesian coordinates for each point can be represented as:

$$\begin{cases} x_i = k_i \cdot \sin(\theta_i + \Delta\theta_i)\cos\varphi_i \\ y_i = k_i \cdot \sin(\theta_i + \Delta\theta_i)\sin\varphi_i \\ z_i = k_i \cdot \cos(\theta_i + \Delta\theta_i) \end{cases} \tag{4}$$

For the encoded image feature patches, we first apply downsampling to reduce the feature dimensions, retaining only the essential information corresponding to the fundamental aspects of the object. Subsequently, two decoder modules are employed to decode each point's k and $\Delta\theta$, which are then converted into Cartesian coordinates to obtain the spatial position of each point. Thanks to our unique method design, during training, we achieve rapid network convergence by solely using the Chamfer Distance (CD) loss for point cloud supervision, significantly reducing the computational complexity associated with other loss functions, such as Earth Mover's Distance (EMD).

3D Gaussian Feature Learning

Tri-plane Representation Efficiently mapping features to 3D space is a longstanding challenge in the field of computer vision. Recently, triplane representations have garnered significant attention due to their efficiency and simplicity (Chan et al. 2022). In this work, we also employ triplanes as a bridge between feature space and 3D space, serving as an intermediate representation of 3D Gaussian features. Inspired by the works of (Jaegle et al. 2021; Hong et al. 2023; Zou et al. 2024) , we use cross-attention to integrate image features with point cloud features that contain spatial information.

Previous research has shown that projecting point clouds onto image features, thereby providing additional image feature information to the point cloud, can effectively enhance the learning of visible viewpoint textures (Zou et al. 2024). However, in our work, due to the lack of multi-view supervision for an object, applying the same methods would hinder the network's ability to predict and complete unseen textures. In our framework, by using global pooling, we discard redundant information in the image features, aiming to retain only fundamental features such as object category and basic geometric shapes as supplementary information, to enhance the network's generalization capabilities during the generation process.

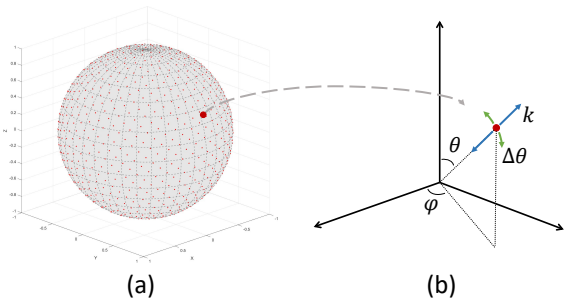

Figure 4: (a) Initializing the point cloud using Fibonacci sphere sampling yields a point cloud uniformly distributed on the sphere. (b) For each point, we maintain its two degrees of freedom in spherical coordinates: the scaling factor k and the polar angle variation $\Delta\theta$.

Let the point cloud be represented as $P = \{p_i\}_{i=1}^{N}$, where each point $p_i = (x_i, y_i, z_i)$ is a vector in R^3. The point cloud is first normalized to a unit space:

$$p_i' = \frac{p_i - p_{\min}}{p_{\max} - p_{\min}} \tag{5}$$

Here, p_{min} represents the minimum value of the point cloud coordinates, and p_{max} is the maximum value. Each normalized point $p_i' = (x_i', y_i', z_i')$ is then projected onto three orthogonal feature planes: the xy-plane, yz-plane, and zx-plane. The corresponding features are interpolated on these 2D planes to obtain the features f_i^{xy}, f_i^{yz}, and f_i^{zx}.

The final feature vector for each point is obtained by concatenating the features from these three projections:

$$f_i = f_i^{xy} \oplus f_i^{yz} \oplus f_i^{zx} \tag{6}$$

Projection Aware Feature Consistency In this work, we reconstruct a 3D model of a car from a single occluded image, which necessitates that the network possesses generative capabilities to complete missing viewpoints and occluded texture features. We propose using Projection-Aware Feature Consistency (PAFC) to guide the network's generative abilities. For a given viewpoint's image features, we project the point cloud onto feature planes to obtain the complete visible point cloud for the current viewpoint. Only a subset of this visible point cloud corresponds to meaningful image features. By utilizing the image mask, we can separate the visible point cloud into occluded and non-occluded point clouds, as illustrated in Figure 5.

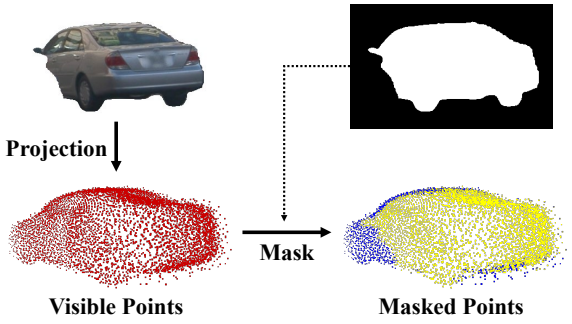

Figure 5: Taking occlusion into account, we can obtain the visible point cloud of the complete model from the current viewpoint, as illustrated by the red point cloud. However, occluded regions provide no useful information for image features. Therefore, we further use a mask to divide the visible point cloud into occluded and non-occluded point clouds, represented by the blue and yellow point clouds, respectively. During the feature constraint process, only the non-occluded point cloud is constrained.

We interpolate the image features for each point in the non-occluded point cloud using the image features. Simultaneously, for all points in the point cloud, features can be generated by the network through sampling on tri-plane. Since the non-occluded point cloud corresponds to all meaningful information in the image, we constrain the network-generated features (i.e., the features obtained by sampling the non-occluded point cloud on the trip-plane) to be close to the image features derived from the projection. By optimizing point clouds at different spatial positions across training iterations, the car's prototype is learned more comprehensively.

In practice, we use the L_2 norm to enforce feature consistency. Specifically, we define the following feature consistency loss function:

$$\mathcal{L}_{PAFC} = \frac{1}{N}\sum_{i=1}^{N}\left\| f_i^{proj} - f_i \right\|_2^2 \tag{7}$$

where N is the number of non-occluded point clouds, and $\|\cdot\|_2$ denotes the L_2 norm. Here, f_i^{proj} represents the image features of the i-th point cloud obtained by projection, while f_i denotes the corresponding features generated by the network.

3D Gaussian Splatting

3D Gaussian Decoder For Gaussian spheres at different positions in space, which should exhibit distinct attribute distributions, we use the positional encoding from NeRF which can help the network capture high-frequency details by mapping the original 3D coordinates to a more expressive feature space. This encoding is concatenated with each point's features as extra information. Specifically, for a point $p_i = (x_i, y_i, z_i)$, its positional encoding is given by:

$$\gamma(p_i) = \left(\sin(2^k\pi p_i), \cos(2^k\pi p_i)\right)_{k=0}^{L-1} \tag{8}$$

where $\gamma(p_i) \in R^{3\times 2\times L}$, with L representing the number of frequency bands used. The sin and cos functions, with frequencies $2^{k-1}\pi$ (for $k = 0, \ldots, L-1$), map the input coordinates into a higher-dimensional space.

For a given point p_i and its corresponding feature f_i, the final complete feature can be expressed as:

$$\hat{f}_i = f_i \oplus \gamma(p_i) \tag{9}$$

For the obtained feature $\hat{f}_i$, we use multiple MLP_{decode} to decode all Gaussian attributes, specifically rotation(R_i), scale (S_i), opacity (O_i), spherical harmonics coefficients (SH_i), and position offset(T_i). This process can be expressed as follows:

$$(R_i, S_i, O_i, SH_i, T_i) = MLP_{decode}(\hat{f}_i) \tag{10}$$

Leveraging differentiable rendering with 3D Gaussian splatting, the rendering results from the input viewpoint can be obtained.

Mapping Mask The input object images may be occluded, meaning we might only have supervision for a subset of the pixels. Addressing how to use partial pixel supervision to guide the network's reconstruction quality is one of the challenges in our task. An intuitive approach is to constrain only the pixels in the visible region, but this leaves other pixels without any supervision. If the reconstructed image is constrained to match the input view entirely, the network will overfit to the input view, losing its ability to complete unseen areas.

Müller et al. (2022a) attempt to address this issue using a pre-trained model combined with manually defined segmentation, Which provides limited shape supervision for unoccluded regions. Moreover, manually designed methods struggle to handle the complexities of real-world scenarios.

Therefore, we propose a simpler but efficient method to address this issue. We utilize point cloud rasterization to project the point cloud onto the pixel plane. If a pixel corresponds to a point, it is marked as 1; otherwise, it is marked as 0, resulting in a coarse projection mask M_{proj}. Since the initialized point cloud may not represent the optimal positions for the object, to retain the information from the input image, we take the union of the projection mask and the input mask M_{input}, and then taking the complement to obtain the final mapping mask $M_{mapping}$:

$$M_{shape} = \neg(M_{proj} \cup M_{input}) \tag{11}$$

The mapping mask $M_{mapping}$ is then used to define the object's shape. Specifically, we apply the mapping mask to both the output image I_{output} and a background image I_{bg} (consisting of the background color, e.g., white). The L_2 loss is used to constrain the unmasked regions of the output image to be close to the corresponding regions of the background image:

$$\mathcal{L}_{shape} = \|M_{shape} \cdot (I_{output} - I_{bg})\|_2^2 \tag{12}$$

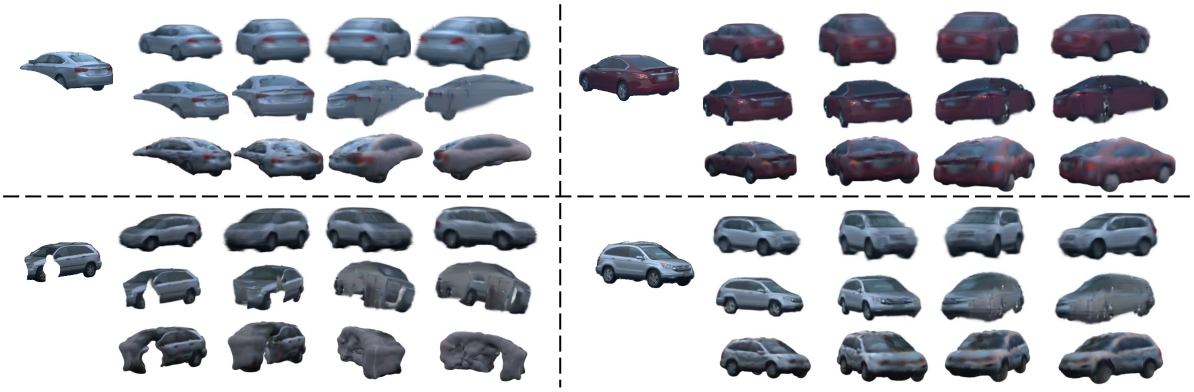

Figure 6: Qualitative comparison of our method with other single-image reconstruction methods based on 3D Gaussians. The top-left image is the input image. The first row shows the novel view results of our method, the second row shows the results of TGS (Zou et al. 2024), and the third row shows the results of Dream Gaussian (Tang et al. 2023). Our method demonstrates superior robustness and generalization compared to the competing methods, both in the presence and absence of occlusions.

Symmetry Prior In this work, we focus on the single-image reconstruction of vehicles in driving scenarios. Given the inherent mirror symmetry of vehicles, we introduce a symmetry prior during the training process. Instead of directly flipping the 3D Gaussian attributes of one side to the other to enforce symmetry in the reconstructed model, we employ an approach akin to data augmentation to encourage the network to actively learn this symmetry. Keeping the camera intrinsics unchanged, we apply mirror symmetry to the input camera poses along the symmetry plane. The image captured with this symmetric pose is thus symmetric to the input image. Similar to the approach applied to the original input images, we also use the mapping mask and imaging results under the symmetric camera poses to constrain the network.

Training

We train our entire model using visible viewpoint images for supervision, along with 3D completed point clouds.

$$\begin{aligned}\mathcal{L} = \lambda_1\mathcal{L}_{CD} + \lambda_2\mathcal{L}_{scale} + \lambda_3\mathcal{L}_{PAFC} + \lambda_4\mathcal{L}_{shape} \\ +\lambda_5\mathcal{L}_{MSE} + \lambda_6\mathcal{L}_{SSIM} + \lambda_7\mathcal{L}_{LPIPS}\end{aligned} \tag{13}$$

In the equation, λ represents the weighting coefficient for each loss. For the initial point cloud generation, we constrain the process using only the $\mathcal{L}_{CD}$ (Chamfer Distance)and a regularization term $\mathcal{L}_{scale}$. In the 3D Gaussian differentiable rendering phase, $\mathcal{L}_{PAFC}$ is employed to supervise the triplane features, $\mathcal{L}_{shape}$ to supervise the generated object shape, and $\mathcal{L}_{MSE}$ and $\mathcal{L}_{SSIM}$ (Wang et al. 2004) to supervise the texture, with $\mathcal{L}_{LPIPS}$ (Zhang et al. 2018)supplementing high-frequency details. It is important to note that for losses constrained by visible pixels, we calculate them exclusively on the visible parts of the rendered result. Since visible pixels may constitute only a small fraction of the total pixels in the input image, which could result in the underestimation of certain losses such as $\mathcal{L}_{MSE}$, we include only visible pixels in the computation to avoid this issue.

Experiments

Datasets

We utilize the Waymo Open Dataset (WOD)(Sun et al. 2020), one of the largest and most comprehensive datasets in the autonomous driving domain. Shen et al. (2023) released a subset derived from the original Waymo dataset, known as the Perception Object Assets Data, which includes a wide range of traffic participants, such as Vehicles, Pedestrians, and Longtail-Vehicles. In this work, we focus on the Vehicle portion of this subset. We aggregated the multi-frame LiDAR point clouds and employed a pre-trained point cloud completion model (Yu et al. 2021b) to complete the point clouds, which served as our ground truth point clouds. Although the completed point clouds contain considerable noise, such as irremovable ground points, our network was still able to generate accurate results. For further details on data processing, please refer to our supplementary material.

Implementation Details

Our image encoder employs the pre-trained Dinov2-small (Oquab et al. 2023) model, which remains frozen during the training process. Images of varying sizes are resized to 252×252 to meet the input requirements of the image encoder; however, within the network, we utilize the original image size for all imaging-related operations. The Fibonacci sphere consists of 16,384 points. Unlike previous methods, we directly obtain all point clouds without the need for additional point cloud upsampling networks. The tri-plane decoder consists of 4 transformer blocks, and the point cloud encoder follows the PointNet (Qi et al. 2017) architecture, with the tri-plane dimensions set to 3×128×64×64. The experiments were conducted on 8 RTX 4090 GPUs, with the batch size set to 32. During training, similar to TGS (Zou et al. 2024), we trained only the point cloud generation part of the network in the first stage, then trained the entire network in the second stage while keeping the point cloud generation network frozen.

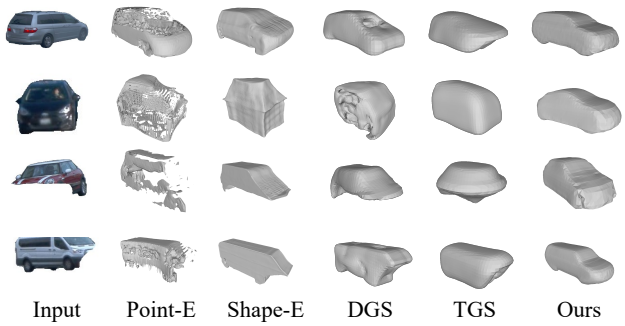

Figure 7: Qualitative comparison of single-image reconstruction geometry. We use the mesh rendering method provided by Shape-E to obtain the mesh. For Point-E, DreamGaussian, TriplaneGaussian and our method, we use the point-to-mesh pre-trained model provided by Point-E to convert the resulting point clouds into meshes.

	$CD_{raw}\downarrow$	$CD_{com}\downarrow$	Time(s) $\downarrow$
Point-E	0.677	0.672	23.64
Shape-E	0.804	1.143	8.200
DGS	0.476	0.687	31.58
TGS	0.567	0.743	0.451
Ours	**0.113**	**0.095**	**0.046**

Table 1: Quantitative comparison of single-image geometry reconstruction. Chamfer Distance is calculated using the raw LiDAR point cloud (raw) and the completed point cloud (complement). Runtime efficiency is measured by the average inference time for a single result.

Baselines

We compare our method with the previous state-of-the-art single-view reconstruction methods, including both qualitative and quantitative comparisons of geometric and texture reconstruction. Point-E (Nichol et al. 2022) generates point clouds from single images using a diffusion model, while Shape-E (Jun and Nichol 2023) generates 3D implicit representations (NeRF). DGS (Tang et al. 2023) proposed a generative 3D Gaussian Splatting model with mesh extraction and texture refinement. TGS (Zou et al. 2024) introduced a hybrid Triplane-Gaussian intermediate representation using two transformer-based networks. AutoRF (Müller et al. 2022a) is the first attempt to achieve vehicle reconstruction from a single image in real-world scenarios.

Initial Gaussian Point Cloud Generation

To validate the geometric reconstruction quality of the initial point cloud generation of our method, we perform both qualitative and quantitative comparisons with Point-E (Nichol et al. 2022), Shape-E (Jun and Nichol 2023), DGS(Tang et al. 2023), and TGS(Zou et al. 2024). As shown in Figure 7, under good viewpoints and complete images, baseline methods may generate meaningful shapes, but they often suffer from significant holes. In cases where the input image has poor viewpoints or occlusions, none of the baseline methods can generate meaningful geometric shapes. However, our method remains robust in all scenarios and produces high-quality results that are consistent with the original vehicle in the input image. As shown in Table 1, we record both Chamfer Distance (CD) and average inference time. We compute the one-sided Chamfer Distance (CD_{raw}) between the raw LiDAR point cloud and the generated point cloud to avoid errors caused by incomplete point cloud shapes in the dataset. Additionally, we calculate the two-sided Chamfer Distance (CD_{com}) using the completed point cloud and the generated point cloud. The data in the table clearly demonstrate that the quality of our point cloud generation significantly outperforms the baseline methods, and is approximately 10 times faster than the fastest baseline method.

	PSNR↑	SSIM↑	LPIPS↓
AutoRF	21.11	0.8835	0.6788
DGS	**25.62**	0.9041	0.2456
TGS	23.87	**0.9162**	**0.2105**
Ours	24.14	0.9058	0.2141

Table 2: Quantitative comparison of visible pixel texture reconstruction results with baseline methods. Although this is not a fair comparison for our method, we still achieve performance comparable to DreamGaussian and TriplanGaussian.

Single View Reconstruction

As shown in Figure 6, under occlusion, both DGS (Tang et al. 2023) and TGS (Zou et al. 2024) fail to produce meaningful reconstruction results, while our method remains robust. In the absence of occlusions, although both methods can maintain consistency with the input image, they often suffer from geometric and texture distortions when viewed from new perspectives. In contrast, our method preserves generalization capability. As shown in Table 2, we present a quantitative comparison with baseline methods. Since we do not have supervision of the complete model, the metrics are calculated only on the visible pixels which is not a fair comparison for our method, as both TGS and DGS apply additional processing to the input viewpoint, while our method focuses on full model generation. The metrics cannot fully evaluate the quality of the generated 3D model, nevertheless, our method still achieves sota results. For more results and details, please refer to our supplementary material.

Conclusion

In this paper, we propose ProtoCar, a novel method for vehicle reconstruction in driving scenarios. We design an efficient initial point cloud generation and Gaussian attribute decoding framework. Experiments demonstrate that ProtoCar can learn vehicle prototype representations and achieve unconstrained single-image to vehicle model reconstruction, even with very poor-quality real-world data. Our approach outperforms existing 3D Gaussian-based methods and NeRF-based baseline methods.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (62227801,62206015,62376024) and by National Science and Technology Major Project (2022ZD0117901).

References

Cabon, Y.; Murray, N.; and Humenberger, M. 2020. Virtual KITTI 2. arXiv:2001.10773.

Chan, E. R.; Lin, C. Z.; Chan, M. A.; Nagano, K.; Pan, B.; De Mello, S.; Gallo, O.; Guibas, L. J.; Tremblay, J.; Khamis, S.; et al. 2022. Efficient geometry-aware 3d generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 16123–16133.

Chang, A. X.; Funkhouser, T.; Guibas, L.; Hanrahan, P.; Huang, Q.; Li, Z.; Savarese, S.; Savva, M.; Song, S.; Su, H.; et al. 2015. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*.

Dai, A.; Chang, A. X.; Savva, M.; Halber, M.; Funkhouser, T.; and Nießner, M. 2017. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 5828–5839.

Deitke, M.; Liu, R.; Wallingford, M.; Ngo, H.; Michel, O.; Kusupati, A.; Fan, A.; Laforte, C.; Voleti, V.; Gadre, S. Y.; et al. 2024. Objaverse-xl: A universe of 10m+ 3d objects. *Advances in Neural Information Processing Systems*, 36.

Dosovitskiy, A.; Ros, G.; Codevilla, F.; Lopez, A.; and Koltun, V. 2017a. CARLA: An open urban driving simulator. In *Conference on robot learning*, 1–16. PMLR.

Dosovitskiy, A.; Ros, G.; Codevilla, F.; Lopez, A.; and Koltun, V. 2017b. CARLA: An open urban driving simulator. In *Conference on robot learning*, 1–16. PMLR.

Downs, L.; Francis, A.; Koenig, N.; Kinman, B.; Hickman, R.; Reymann, K.; McHugh, T. B.; and Vanhoucke, V. 2022. Google scanned objects: A high-quality dataset of 3d scanned household items. In *2022 International Conference on Robotics and Automation (ICRA)*, 2553–2560. IEEE.

Engelmann, F.; Stückler, J.; and Leibe, B. 2017. SAMP: shape and motion priors for 4d vehicle reconstruction. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 400–408. IEEE.

Garbin, S. J.; Kowalski, M.; Johnson, M.; Shotton, J.; and Valentin, J. 2021. Fastnerf: High-fidelity neural rendering at 200fps. In *Proceedings of the IEEE/CVF international conference on computer vision*, 14346–14355.

Gkioxari, G.; Malik, J.; and Johnson, J. 2019. Mesh r-cnn. In *Proceedings of the IEEE/CVF international conference on computer vision*, 9785–9795.

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

Jaegle, A.; Gimeno, F.; Brock, A.; Vinyals, O.; Zisserman, A.; and Carreira, J. 2021. Perceiver: General perception with iterative attention. In *International conference on machine learning*, 4651–4664. PMLR.

Jun, H.; and Nichol, A. 2023. Shap-e: Generating conditional 3d implicit functions. *arXiv preprint arXiv:2305.02463*.

Kanazawa, A.; Tulsiani, S.; Efros, A. A.; and Malik, J. 2018. Learning category-specific mesh reconstruction from image collections. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 371–386.

Kerbl, B.; Kopanas, G.; Leimkühler, T.; and Drettakis, G. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Trans. Graph.*, 42(4): 139–1.

Kundu, A.; Li, Y.; and Rehg, J. M. 2018. 3d-rcnn: Instance-level 3d object reconstruction via render-and-compare. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3559–3568.

Lin, C.-H.; Gao, J.; Tang, L.; Takikawa, T.; Zeng, X.; Huang, X.; Kreis, K.; Fidler, S.; Liu, M.-Y.; and Lin, T.-Y. 2023. Magic3d: High-resolution text-to-3d content creation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 300–309.

Liu, M.; Xu, C.; Jin, H.; Chen, L.; Varma T, M.; Xu, Z.; and Su, H. 2024. One-2-3-45: Any single image to 3d mesh in 45 seconds without per-shape optimization. *Advances in Neural Information Processing Systems*, 36.

Liu, R.; Wu, R.; Van Hoorick, B.; Tokmakov, P.; Zakharov, S.; and Vondrick, C. 2023. Zero-1-to-3: Zero-shot one image to 3d object. In *Proceedings of the IEEE/CVF international conference on computer vision*, 9298–9309.

Mescheder, L.; Oechsle, M.; Niemeyer, M.; Nowozin, S.; and Geiger, A. 2019. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 4460–4470.

Mildenhall, B.; Srinivasan, P. P.; Tancik, M.; Barron, J. T.; Ramamoorthi, R.; and Ng, R. 2021. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1): 99–106.

Müller, N.; Simonelli, A.; Porzi, L.; Bulò, S. R.; Nießner, M.; and Kontschieder, P. 2022a. Autorf: Learning 3d object radiance fields from single view observations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 3971–3980.

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

Nichol, A.; Jun, H.; Dhariwal, P.; Mishkin, P.; and Chen, M. 2022. Point-e: A system for generating 3d point clouds from complex prompts. *arXiv preprint arXiv:2212.08751*.

Niemeyer, M.; and Geiger, A. 2021. Giraffe: Representing scenes as compositional generative neural feature fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11453–11464.

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

Ost, J.; Mannan, F.; Thuerey, N.; Knodt, J.; and Heide, F. 2021. Neural scene graphs for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2856–2865.

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

Qi, C. R.; Su, H.; Mo, K.; and Guibas, L. J. 2017. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 652–660.

Ramesh, A.; Pavlov, M.; Goh, G.; Gray, S.; Voss, C.; Radford, A.; Chen, M.; and Sutskever, I. 2021. Zero-shot text-to-image generation. In *International conference on machine learning*, 8821–8831. Pmlr.

Rematas, K.; Martin-Brualla, R.; and Ferrari, V. 2021. Sharf: Shape-conditioned radiance fields from a single view. *arXiv preprint arXiv:2102.08860*.

Richter, S. R.; Vineet, V.; Roth, S.; and Koltun, V. 2016a. Playing for data: Ground truth from computer games. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*, 102–118. Springer.

Richter, S. R.; Vineet, V.; Roth, S.; and Koltun, V. 2016b. Playing for data: Ground truth from computer games. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*, 102–118. Springer.

Rombach, R.; Blattmann, A.; Lorenz, D.; Esser, P.; and Ommer, B. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 10684–10695.

Rukhovich, D.; Vorontsova, A.; and Konushin, A. 2022. Imvoxelnet: Image to voxels projection for monocular and multi-view general-purpose 3d object detection. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2397–2406.

Schwarz, K.; Liao, Y.; Niemeyer, M.; and Geiger, A. 2020. Graf: Generative radiance fields for 3d-aware image synthesis. *Advances in Neural Information Processing Systems*, 33: 20154–20166.

Shen, B.; Yan, X.; Qi, C. R.; Najibi, M.; Deng, B.; Guibas, L.; Zhou, Y.; and Anguelov, D. 2023. Gina-3d: Learning to generate implicit neural assets in the wild. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 4913–4926.

Skorokhodov, I.; Siarohin, A.; Xu, Y.; Ren, J.; Lee, H.-Y.; Wonka, P.; and Tulyakov, S. 2023. 3d generation on imagenet. *arXiv preprint arXiv:2303.01416*.

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

Tang, J.; Ren, J.; Zhou, H.; Liu, Z.; and Zeng, G. 2023. Dreamgaussian: Generative gaussian splatting for efficient 3d content creation. *arXiv preprint arXiv:2309.16653*.

Wang, Z.; Bovik, A. C.; Sheikh, H. R.; and Simoncelli, E. P. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4): 600–612.

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

Yang, Z.; Chen, Y.; Wang, J.; Manivasagam, S.; Ma, W.-C.; Yang, A. J.; and Urtasun, R. 2023. Unisim: A neural closed-loop sensor simulator. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1389–1399.

Yu, A.; Ye, V.; Tancik, M.; and Kanazawa, A. 2021a. pixelnerf: Neural radiance fields from one or few images. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 4578–4587.

Yu, X.; Rao, Y.; Wang, Z.; Liu, Z.; Lu, J.; and Zhou, J. 2021b. Pointr: Diverse point cloud completion with geometry-aware transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, 12498–12507.

Zakharov, S.; Kehl, W.; Bhargava, A.; and Gaidon, A. 2020. Autolabeling 3d objects with differentiable rendering of sdf shape priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 12224–12233.

Zhang, R.; Isola, P.; Efros, A. A.; Shechtman, E.; and Wang, O. 2018. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 586–595.

Zhou, H.; Shao, J.; Xu, L.; Bai, D.; Qiu, W.; Liu, B.; Wang, Y.; Geiger, A.; and Liao, Y. 2024. Hugs: Holistic urban 3d scene understanding via gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 21336–21345.

Zou, Z.-X.; Yu, Z.; Guo, Y.-C.; Li, Y.; Liang, D.; Cao, Y.-P.; and Zhang, S.-H. 2024. Triplane meets gaussian splatting: Fast and generalizable single-view 3d reconstruction with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10324–10335.