UrbanCAD: Towards Highly Controllable and Photorealistic 3D Vehicles for Urban Scene Simulation

Yichong Lu[1*] Yichi Cai[1*] Shangzhan Zhang[1] Hongyu Zhou[1] Haoji Hu[1]
Huimin Yu[1] Andreas Geiger[2,3] Yiyi Liao[1†]
[1]Zhejiang University [2]University of Tübingen [3]Tübingen AI Center
https://xdimlab.github.io/UrbanCAD/

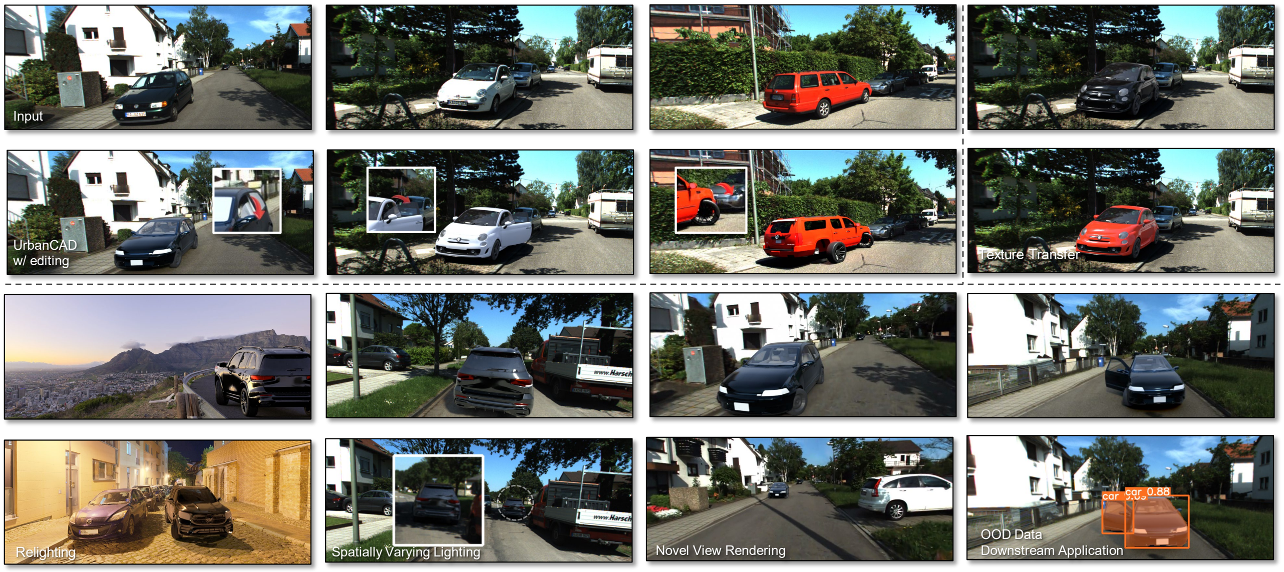


Figure 1. **UrbanCAD** automatically builds photorealistic and highly controllable digital twins from a single urban image and a large collection of 3D CAD models and handcrafted materials, supporting various editing operations (top). The produced CAD models can be photorealistically inserted into various background scenes and rendered in novel views, synthesizing challenging out-of-distribution (OOD) scenarios with high fidelity for important downstream applications (bottom).

Abstract

Photorealistic 3D vehicle models with high controllability are essential for autonomous driving simulation and data augmentation. While handcrafted CAD models provide flexible controllability, free CAD libraries often lack the high-quality materials necessary for photorealistic rendering. Conversely, reconstructed 3D models offer high-fidelity rendering but lack controllability. In this work, we introduce UrbanCAD, a framework that generates highly controllable and photorealistic 3D vehicle digital twins from a single urban image, leveraging a large collection of free 3D CAD models and handcrafted materials. To achieve this, we propose a novel pipeline that follows a retrieval-optimization manner, adapting to observational data while preserving fine-grained expert-designed priors for both geometry and material. This enables vehicles' realistic 360° rendering, background insertion, material transfer, relighting, and component manipulation. Furthermore, given multi-view background perspective and fisheye images, we approximate environment lighting using fisheye images and reconstruct the background with 3DGS, enabling the photorealistic insertion of optimized CAD models into rendered novel view backgrounds. Experimental results demonstrate that UrbanCAD outperforms baselines in terms of photorealism. Additionally, we show that various perception models maintain their accuracy when evaluated on UrbanCAD with in-distribution configurations but degrade when applied to realistic out-of-distribution data generated by our method. This suggests that UrbanCAD is a significant advancement in creating photorealistic, safety-critical driving scenarios for downstream applications.

[*] Equal contribution. [†] Corresponding author.

1. Introduction

Photorealistic driving simulators have gained great attention for providing a safe and cost-effective way to evaluate driving algorithms [62, 68, 69]. Digital twins of vehicles, representing key traffic participants, are essential for these simulators. Since simulators must assess driving algorithms in both common and rare, long-tailed scenarios, the vehicles within these simulators must exhibit both *photorealism* and *controllability*.

Classical driving simulators based on game engines, such as CARLA [16], use handcrafted CAD models to represent vehicles. While offering high controllability, they suffer from a significant domain gap compared to the real world. Leveraging easily accessible real-world urban images, photorealistic simulation offers a scalable solution to bridge this gap across diverse scenarios. This direction has gained wide attention with advances in neural rendering techniques [44, 51, 56, 57, 65]. While these methods close the domain gap and enable photorealistic rendering, they lack fine-grained control over the reconstructed vehicles. [62] employs CAD models of cars as shape priors and reconstructs vehicles with controllable wheels, appearance, and scene lighting using differentiable rendering. However, it still provides limited control over other vehicle components and yields suboptimal geometry. In addition, vehicles are usually observed from limited viewpoints in urban scenes, impeding the photorealistic rendering of occluded regions. Although prior knowledge can help reconstruct unobserved parts [42, 46], the quality of these regions remains unsatisfactory. Furthermore, complex material physical properties such as opacity and metallic reflectance, pose significant challenges for differentiable rendering, especially in our single-view observation setting. In contrast, handcrafted material libraries like Adobe Material Library [1] provide materials with complex physical properties. Motivated by these observations, we seek to push the frontier of the photorealism-controllability trade-off.

We move towards this objective by introducing a novel framework that automatically produces 3D vehicles with photorealistic appearance, fine-grained geometric details, and high controllability, including part-level control, from a single urban image and a collection of free 3D CAD models and handcrafted materials. This framework operates within a retrieval-optimization paradigm, performing both CAD retrieval and retrieval-based material optimization. The key idea is to retain the existing detailed expert-designed priors including part-level controllability and complex material physical properties, while refining the appearance to fit the observational data. Specifically, we first retrieve vehicles' geometries represented by handcrafted CAD models, which offer high controllability due to disentangled designs, particularly part-disentangled geometry for component editing. Existing reconstruction-based methods [26, 42, 62] that use CAD models for training focus only on their appearance or overall geometry, neglecting their disentangled geometry design, which results in a loss of part controllability. Then, we retrieve vehicles' materials represented by optimizable, manually designed procedural graphs, which offer photorealistic material properties—such as opacity and roughness—that are difficult to optimize directly. However, for vehicles that have multiple types of materials, accurately retrieving and assigning the part-aware material priors are not trivial. Previous works [55, 71] typically retrieve material priors based on visual similarity, which is inaccurate for materials with complex physical properties, and primarily focus on objects with a single material. To address this, we retrieve part-aware materials based on semantic meanings via foundation models and assign materials through a ControlNet-based recognition method informed by the retrieved material designs. Finally, we perform part-aware material optimization using physics-based differentiable rendering to align the appearance with the input image. In addition, given multi-view fisheye and perspective images of the background scene, we propose a fisheye-based spatially varying lighting estimation method to realistically render the optimized CAD model and reconstruct the background using 3D Gaussian Splatting [82] for high-fidelity novel view background synthesis. The integration of these renderings results in photorealistic novel view synthesis and versatile controllability over foreground vehicles.

Using this comprehensive pipeline, we systematically evaluate several perception models on our synthesized images. Our experimental results demonstrate that pre-trained perception models retain their performance when replacing real cars with our CAD model renderings for in-distribution data generation. However, they show a clear performance drop when UrbanCAD is used for generating out-of-distribution scenarios, such as cars with opened doors. These results indicate that UrbanCAD produces photorealistic and controllable 3D assets, enabling the creation of rare scenarios for autonomous driving that are not achievable with reconstruction- or retrieval-based methods.

Our main contributions are as follows: 1) We propose a novel pipeline based on the retrieval-optimization paradigm that automatically constructs photorealistic and highly controllable 3D vehicle digital twins with detailed geometry. These digital twins closely align with a single input image and allow for control even over part-level components. 2) Our system allows for inserting the optimized 3D digital twins back into various urban scenes, and achieving novel view synthesis of the full scene when multi-view images are provided for background reconstruction. 3) We evaluate various vehicle models in terms of fidelity and downstream task accuracy. Our results indicate that our CAD retrieval, material optimization, and lighting estimation modules are all crucial for generating photorealistic out-of-distribution

(OOD) scenarios, such as door opening, which are vital for testing the robustness of autonomous perception systems.

2. Related Work

Simulation for Autonomous Driving: There are two major approaches to sensor simulation for autonomous driving: graphics-based methods [16, 18, 54] and data-driven methods [37, 41, 53, 61, 82]. Graphics-based simulators, such as CARLA [16] and AirSim [54], are fast and highly controllable but produce unrealistic simulation results due to substantial manual effort, leading to a significant domain gap for autonomous systems. Recently, data-driven methods [22, 30, 36, 38, 43, 44, 48, 51, 57, 63, 65, 68, 80] have made significant progress in realistic novel view synthesis using neural fields. However, most of these methods have limited editing capabilities and yield suboptimal results when viewing from a large range of angles due to limited observation data. Some approaches like [62] represent vehicles with mesh and model the wheels separately, allowing for wheel rotation during simulation. However, optimizing the geometry, material, and lighting together in an end-to-end manner is challenging and this design still lacks full controllability over other vehicle components, e.g., windows and doors. In contrast, our CAD model retrieval and optimization-based approach yields a good trade-off between photorealism and controllability.

CAD Model as Scene Representations: CAD model retrieval has been investigated in many existing approaches [5, 18, 19, 21, 31, 59]. While obtaining good geometry details, the appearance of retrieved CAD models is often unsatisfactory because of the lack of optimization. Another line of works [17, 60, 62] utilizes the CAD models as priors and performs geometry optimization afterward. While the optimized geometry is closer to the observation, the CAD models are converted to other scene representations, e.g., implicit surfaces, to allow for optimization, hence losing controllability over vehicle components. In contrast, we retain the detailed geometry and high controllability of CAD models while achieving photorealistic appearance. Concurrently, ACDC [14] obtains the digital cousins via CAD retrieval, but it doesn't perform material optimization and lighting estimation for photorealistic rendering to further reduce the domain gap.

Material Transfer from Images: Recently, image-based mesh texturing methods [7, 45, 72, 75] using generative models have demonstrated strong performance. However, these methods typically rely on per-vertex texture maps for material representation, which can lead to slow optimization processes. Conversely, we employ procedural graphs to represent materials, resulting in higher quality and faster optimization speeds. Besides, these methods often suffer from multi-face or blurry problems whereas our approach achieves fine-grained and photorealistic materials through effective retrieval and optimization techniques. Another line of work [66, 71] using optimizable procedural graphs mainly focuses on objects with a single material, such as furniture. In contrast, our method extends its capabilities to objects with complex materials, such as vehicles, by part-aware material retrieval.

3. Vehicle CAD Retrieval and Optimization

Our method begins with CAD retrieval and optimization, using a single urban image and a large collection of free CAD models and material graphs as input. Our aim is to create digital twins that match the reference vehicles in the real-world images, both in geometry and appearance. While these free CAD models are handcrafted with animatable components and detailed geometry, they often lack the high-quality materials required for photorealistic rendering.

The process consists of three stages, as illustrated in Fig. 2. First, we perform image-based CAD model retrieval given a single view input image (Section 3.1). Next, we perform part-aware material prior retrieval using vision foundation models (Section 3.2) and refine the material quality through part-aware optimization (Section 3.3).

3.1. CAD Model Retrieval

Given a single urban image $\mathbf{I}_{input}$ and one user-selected 2D point inside a target vehicle, we segment the reference vehicle image $\mathbf{I}_{ref}$ from the input scene using SAM [29]. Note that vehicle segmentation can also be automatically obtained via foundation models like GroundedSAM [52]. Next, each segmented vehicle image $\mathbf{I}_{ref}$ is encoded into latent code $\mathbf{L}_{ref}$ using a pre-trained image encoder $\mathcal{E}_{clip}$ [49]: $\mathbf{L}_{ref} = \mathcal{E}_{clip}(\mathbf{I}_{ref})$. Then, we compute latent codes $\left\{\mathbf{L}^i_{cad}\right\}_{i=1}^N$ for all CAD models $\{M^i_{\text{cad}}\}_{i=1}^N$ in our library using a pre-trained multi-modality aligned 3D encoder $\mathcal{E}_{3d}$ [39] : $\left\{\mathbf{L}^i_{cad}\right\}_{i=1}^N = \mathcal{E}_{3d}(\{M^i_{\text{cad}}\}_{i=1}^N)$. Here, $\mathcal{E}_{3d}$ maps both images and 3D shapes to a shared latent space, where a closer distance in this latent space indicates greater semantic similarity. These latent codes can be pre-cached for efficiency. Finally, we compare the latent codes of the input vehicle images with those of the CAD models using cosine similarity. We identify the CAD model with the highest cosine similarity to the input vehicle image by solving: $\arg\max_i \left(\text{sim}(\mathbf{L}_{\text{ref}}, \{\mathbf{L}^i_{\text{cad}}\}_{i=1}^N)\right)$. Note that this kind of retrieval can obtain CAD models with the highest semantic similarities aligned with the input image while preserving the handcrafted priors including flexible controllability, detailed geometry, and symmetric material design.

3.2. Material Prior Retrieval

Adobe Material Library [1] offers a rich collection of high-quality handcrafted procedural material graphs, which can

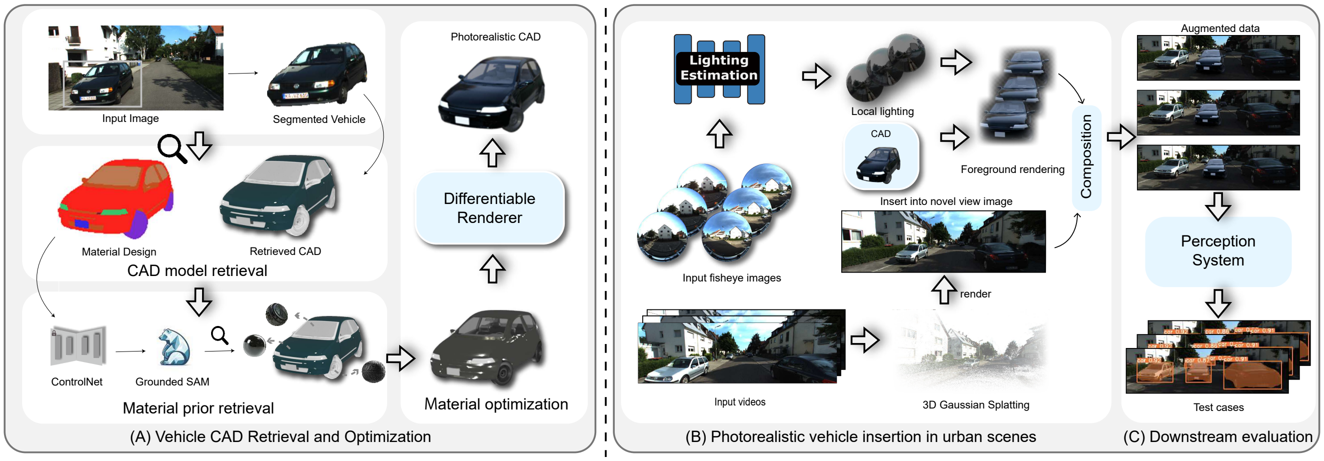


Figure 2. **Overview of UrbanCAD.** Given a single view input image, we first perform CAD model retrieval and retrieval-based material optimization to create photorealistic and highly controllable vehicle digital twins (left). Given multi-view background images, we then perform realistic vehicle insertion to create various synthetic data for self-driving system testing (right).

serve as effective material priors. Consequently, we begin with material prior retrieval before proceeding to material optimization. Previous works [66, 71] focused on objects with single materials and retrieved material categories based on visual similarity. However, this type of retrieval can be inaccurate, particularly for objects such as vehicles, which are composed of various materials, including glass, where color representation can be ambiguous. To address this problem, we propose retrieving part-aware material priors based on the semantic characteristics of CAD model parts, e.g., windows, wheels, and car bodies.

Material Prior: Optimizable Procedural Node Graphs. Procedural node graphs $\mathbf{G}$ provide an expressive material representation in graphics. Unlike per-pixel material parameter maps, these graphs can compactly represent various materials using a small amount of parameters. MATch [55] proposes converting such node graphs into differentiable programs, utilizing differentiable rendering to optimize continuous node parameters in an end-to-end manner through rendering loss. The discrete parameters and graph structure, designed by artists, remain fixed. In this work, we first collect handcrafted procedural material graphs from Adobe Material Library and rename them to the corresponding CAD model part names. For instance, when retrieving material priors for vehicles, we collect three artist-designed base node graphs — glass, rubber, and reflective metal — and rename them to windows, wheels, and car bodies. Note that this process typically needs to be conducted only once for most objects within a given category, as they often share a common set of materials. Then, we follow MATch to translate the handcrafted graphs into optimizable ones to fit the observation data efficiently.

Semantic-based Part Material Prior Retrieval. Considering that a single vehicle comprises various materials, we need to assign different parts of the CAD models with specific base procedural graphs. Importantly, we have obtained symmetric material design during our CAD retrieval, where disconnected components with the same semantic meaning (e.g., left and right windows) are assigned the same initial material index (refer to the supplementary Fig. 9). The semantic meanings of these material indexes are unknown. Consequently, we only need to recognize the semantic meaning of the indexes in the material design for effective material retrieval. However, directly interpreting the semantic meanings from material designs or retrieved CAD model renderings can be inaccurate, as they often present unrealistic appearances (see Fig. 3). To solve this, we use ControlNet [78] to produce photorealistic images based on the retrieved material design and use Grounded SAM [52], a foundational vision model that combines a large language model with image segmentation, to identify the part-level meanings of the retrieved material design (see the supplementary Fig. 10 for illustration). To enhance the robustness further, we also implement multi-view recognition. For the car body, it is challenging to recognize it directly using text prompts. Therefore, after identifying the other components, we treat the largest area of the remaining part as the car body. We then retrieve base procedural graphs ($\mathbf{G}_{init}$) based on the names of the recognized components. This method allows us to assign material priors to the entire vehicle robustly without the need for accurate part-segmentation.

3.3. Material Optimization

Material Graph Differentiable Rendering. We follow DiffMat v2 [32] to convert the material node graph into texture elements like the albedo map $\mathbf{A}_{uv}$, normal map $\mathbf{N}_{uv}$, and roughness map $\mathbf{R}_{uv}$. This produces a physically-based

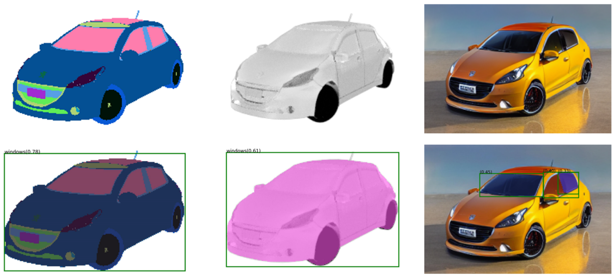


Figure 3. **Window recognition results** on colored material design, retrieved CAD rendering, and augmented data by ControlNet [78].

microfacet BRDF [27] model. To obtain the per-pixel material parameters $\mathbf{A}$, $\mathbf{N}$, and $\mathbf{R}$, we use the UV sampling function $\mathbf{Sample}$ to sample the material textures $\mathbf{A}_{uv}$, $\mathbf{N}_{uv}$, and $\mathbf{R}_{uv}$ from the per-pixel texture (UV) coordinates $\mathbf{UV}$ of the UV map rendered in the matched pose (see supplementary Section 7.2). Combined with the estimated incoming lighting $\mathbf{L}$ [33], we perform differentiable rendering to achieve the estimated rendering results shown below:

$$\mathbf{A}_{uv}, \mathbf{N}_{uv}, \mathbf{R}_{uv} = \mathbf{DiffMat}(\mathbf{G}) \tag{1}$$

$$\mathbf{A}, \mathbf{N}, \mathbf{R} = \mathbf{Sample}(\mathbf{A}_{uv}, \mathbf{N}_{uv}, \mathbf{R}_{uv}, \mathbf{UV}) \tag{2}$$

$$\mathbf{I}_{render} = \mathbf{Render}(\mathbf{A}, \mathbf{N}, \mathbf{R}, \mathbf{L}) \tag{3}$$

where $\mathbf{Render}$ is the differentiable renderer adopted from InvRenderNet [33].

Part-Aware Material Optimization. We also segment the components in the reference view using Grounded SAM and optimize the corresponding material of the CAD model to align with the reference view. Since there is no exact correspondence between rendered and reference pixels, we use a part-level loss ℓ_{stat} by minimizing the difference between the mean and variance of the corresponding parts following [71]. To match the patterns of the reference view, we use a masked VGG loss ℓ_{vgg} using Gram matrices [20] to enhance visual similarity. To further align the color of the reference vehicles, we incorporate a masked RGB loss ℓ_{rgb} on the overlap region between components in the reference view and CAD model rendering. Please see the supplementary Section 7.5 for details. The total loss optimizes the parameters of the material graphs with backpropagation. Note that optimizing complex material physical properties from single view images, such as opacity and roughness through differentiable rendering is not ideal. Instead, we only optimize the albedo of the retrieved metal and rubber materials to align with the input image and directly assign the retrieved glass material to the car window without further optimization, which can produce satisfactory results, as demonstrated in our experiments.

4. Photorealistic Insertion in Urban Scenes

To construct photorealistic and controllable urban scenes with our optimized 3D CAD models, we need to seamlessly integrate them into the provided urban backgrounds. Given multi-view background perspective and fisheye images, we begin by rendering the vehicles with lighting that matches the estimated environmental conditions (Section 4.1). Next, we compose these rendered vehicles with the scene's background created through reconstruction methods (Section 4.2).

4.1. Environment Lighting Estimation

To render our vehicle models realistically, accurately estimating the scene's environment lighting map is essential. While per-pixel incoming lighting estimation (as discussed in Section 3.3) is one option, it lacks global consistency and can cause artifacts when the vehicle is moved. To address this, we propose using multi-view fisheye images mounted on both sides of a car [34] to estimate the lighting environment. A pair of fisheye images provides a 360° field of view, enabling the construction of a globally consistent environment map for each pair. We first convert each fisheye pair into a panorama image. Given that the images captured by the fisheye cameras are in low dynamic range (LDR) format, we transform the upper half of the LDR panorama image into high dynamic range (HDR) to accurately represent the lighting of the skydome using a pre-trained network [64]. Next, to incorporate other objects in the scene, we segment the non-sky region of the LDR panorama using FastSAM [81] and then compose it with the HDR skydome after aligning the value ranges. This approach allows us to consider the surrounding lighting and shadow caused by foreground objects when rendering. To further achieve the spatially varying effect, we select the environment map closest to the insertion position based on the distances between the insertion position and the fisheye camera locations. Although this is a rough approximation compared to time-consuming ray-tracing techniques [48], it results in a reasonable and robust performance.

4.2. Background Reconstruction and Compostion

Since autonomous driving simulators require free navigation within the scene, we integrate our method with novel view synthesis (NVS) to enable this functionality. Given input background videos, we use the 3D Gaussian Splatting method [82] to reconstruct the environment and render background images from novel views. Subsequently, we render the foreground vehicle using Blender [2] and blend it with the background images via alpha composition. Specifically, we position the vehicle models generated in Section 3 in Blender according to a target 3D position obtained from annotations or trajectory generation methods [77]. A virtual plane is utilized to account for shadow effects based on the estimated ground plane. Using the previously estimated HDR environment map, we render the vehicles and composite the rendered vehicles with the background images.

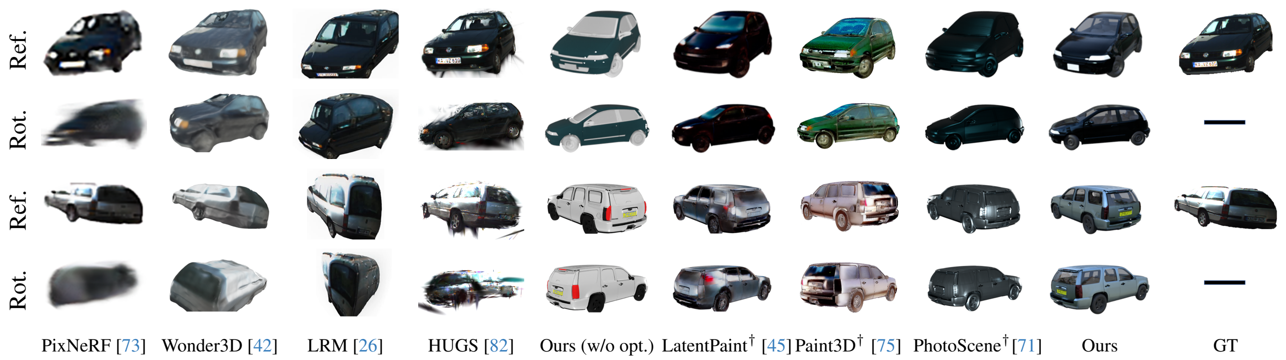


Figure 4. **Qualitative results** on KITTI-360 for novel view synthesis from reference (Ref.) and rotated (rot.) viewpoints. UrbanCAD produces more robust and realistic results at the novel viewpoint compared to the baselines.

Please refer to the supplementary Section 7.7 for details.

5. Experiment

5.1. Experiment Setup

Datasets. We evaluate our method on various urban datasets. We mainly conduct experiments on the KITTI-360 dataset [34] which contains high-quality fisheye images. However, we also present CAD model optimization results on the Multi-View Marketplace Cars (MVMC) [76] dataset since our CAD retrieval and optimization module can function without fisheye images. For the CAD models, we utilize the Objaverse library [15], a free 3D asset repository containing 26k+ car and vehicle models. For the base procedural material graphs, we collect them from the Adobe 3D Asset Library [1] containing 13k+ materials.

Baselines. We compare our approach with various types of methods: (1) Single-view reconstruction method using the conditional implicit function: PixelNeRF [73]. (2) Single-view generation method using diffusion prior: Wonder3D [42]. (3) Single-view reconstruction method using large reconstruction model: LRM [26]. (4) Multi-view reconstruction method using 3DGS: HUGS [82]. (5) Mesh texturing methods using the generative model: LatentPaint† [45] and Paint3D† [75]. (6) Mesh texturing method using optimizable procedural graph: PhotoScene† [71]. Note that we use our CAD retrieval module (Section 3.1) to retrieve CAD models before mesh texturing with LatentPaint [45], Paint3D [75] and PhotoScene [71] (marked as "LatentPaint†", "Paint3D†" and "PhotoScene†"). We also investigate the performance when directly using our CAD retrieval module without material retrieval and optimization (marked as UrbanCAD (w/o opt.)).

Metrics. Given our focus on the controllable aspects of digital twins, we manipulate the vehicles to be rendered from different viewpoints and evaluate the Fréchet Inception Distance (FID) [25] and Kernel Inception Distance (KID) [6] between these renderings and real-world car datasets [67]. This assesses the photorealism in terms of free viewpoint controllability. To evaluate the reconstruction quality, we also calculate the Learned Perceptual Image Patch Similarity (LPIPS) [79] between the input reference vehicle images and the renderings under matched poses (detailed in the supplementary). Additionally, we evaluate the performance of self-driving perception methods on our generated synthetic data using Intersection over Union (IOU) and Panoptic Quality (PQ) [28] metrics for all vehicles in the scenes. We also assess corner cases using both category-level IOU for all vehicles and instance-level IOU for a specific vehicle.

Reconstruction-based	Retrieval-based	Method	FID↓	KID↓	LPIPS↓
✓		PixelNeRF [73]	264.61	0.2415	-
✓		Wonder3D [42]	246.43	0.2292	-
✓		LRM [26]	220.77	0.2050	-
✓		HUGS [82]	240.92	0.2417	-
	✓	UrbanCAD (w/o opt.)	81.05	0.0567	0.6174
	✓	LatentPaint† [45]	85.62	0.0604	0.5525
	✓	Paint3D† [75]	67.52	**0.0417**	0.5652
	✓	PhotoScene† [71]	170.21	0.1561	0.5422
	✓	UrbanCAD (Ours)	**62.80**	0.0479	**0.5242**

Table 1. **Quantitative Comparison** on the photorealism.

5.2. Photorealism Quality

Comparison to baselines. As shown in Fig. 1, UrbanCAD successfully reconstructs photorealistic vehicles within the provided urban images. We compare our method with baseline approaches both qualitatively (Fig. 4) and quantitatively (Table 1). Given that self-driving simulation systems require vehicles to move freely within the scene, we focus on novel view rendering results across 360°. We report the FID and KID metrics for the 360° renderings of the vehicles in Table 1. UrbanCAD demonstrates superior performance on FID and KID compared to most baselines, indicating that our vehicle models are more realistic. Interestingly, even only using our CAD retrieval module (UrbanCAD (w/o opt.)) outperforms the other reconstruction-based baselines in terms of FID and KID. This can be attributed to the fact that the reconstruction-based baselines may only provide reasonable reconstruction near the reference viewpoint (PixelNeRF, HUGS, Wonder3D, LRM). We also report the LPIPS metrics between reference images and

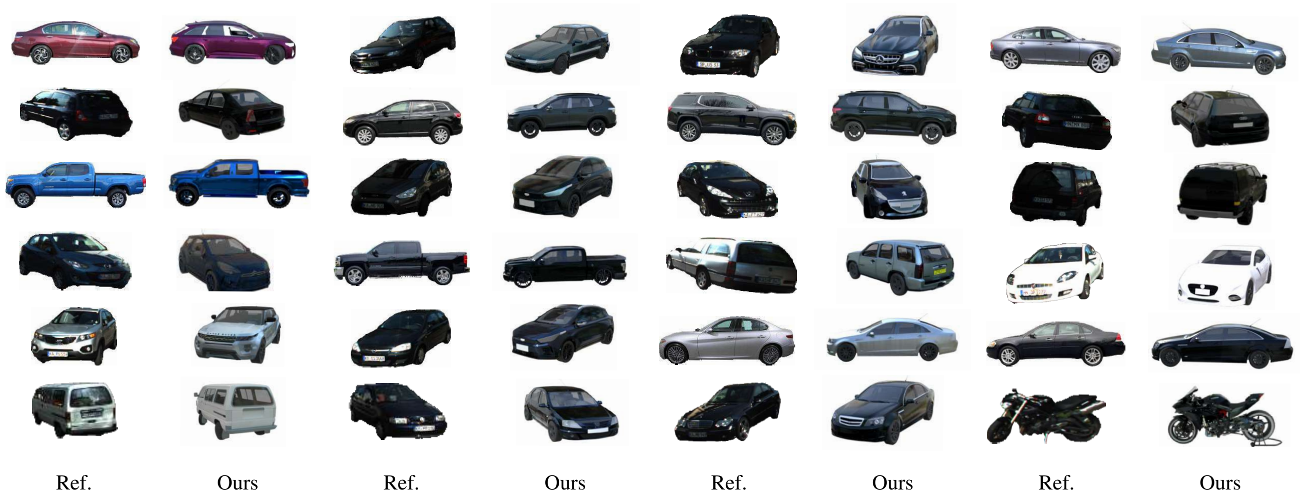


Figure 5. **More pairs** of 3D vehicles after CAD model retrieval and material optimization (right) alongside the input single-view segmented vehicles (left). UrbanCAD produces photorealistic 3D vehicles with different categories given single-view inputs.

CAD model renderings under matched poses for retrieval-based methods in Table 1. Compared to the baselines, UrbanCAD has better performance on LPIPS, suggesting that our vehicle models are more similar to the vehicles in the reference image. This is probably because the baselines lack high-frequency details and accurate material estimation (LatentPaint†, PhotoScene†, Paint3D†). Our qualitative results in Fig. 4 and Fig. 7 further demonstrate that UrbanCAD produces superior outcomes, especially at large rotation angles. Besides, we provide a quantitative comparison with a 3D reconstruction method [76] based on a surface implicit model (see the supplementary Table 5). We also notice that Paint3D† performs better on KID compared to our method. This is probably because Paint3D tends to generate unrealistic yet rich textures as shown in Fig. 4 and Fig. 7.

Generalization ability. Thanks to the large scale of the CAD model library and our material optimization module, our method exhibits strong generalization ability on various kinds of vehicles including cars, trucks, vans, and motorcycles as shown in Fig. 5.

Lighting estimation results. We compare our fisheye-based lighting estimation module with baselines qualitatively (see the supplementary Section 10.1). The results demonstrate that our method can produce more accurate lighting for photorealistic vehicle insertion in urban scenes.

Ablation study. The comparison against UrbanCAD (w/o opt.) and PhotoScene† in Fig. 4 and Table 1 highlights the significance of our retrieval-based material optimization and part-aware material prior retrieval modules. Results in Table 2 and supplementary Section 10.1 demonstrate our lighting estimation module is crucial for constructing photorealistic scenarios.

Robustness to occlusion. We present more results based on partial observations in Fig. 6. We find SAM is robust to occlusion, likely due to its training on occluded images. Our CAD retrieval module demonstrates resilience to occlusion as well, as it utilizes a 3D encoder aligned with CLIP's latent space to encode CAD models and retrieve them based on semantic instead of visual similarity. In addition, our material optimization module produces reliable results by leveraging part-aware regularization between corresponding semantic regions, eliminating the need for precise pixel-level alignment.

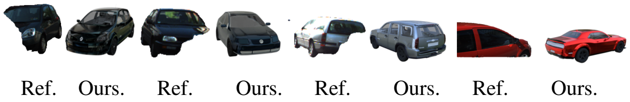


Figure 6. Recovery from partial observation.

5.3. Downstream Application.

To further explore whether our optimized CAD models facilitate the rendering of photorealistic images that can enhance downstream applications, we evaluate several pre-trained segmentation models on our augmented data, which consists of optimized CAD models blended with ground truth (GT) images. Specifically, we test the pre-trained YOLO V8 instance segmentation model, a widely used real-time instance segmentation method that combines the YOLO V8 detection model [50] with YOLACT [8]. Additionally, we assess the performance of the instance segmentation model Mask2Former [12], utilizing different backbones, on both our synthetic data and real-world reference data.

In-Distribution Driving Scenarios. Firstly, we construct normal in-distribution driving scenarios using the vehicles generated by Wonder3D, UrbanCAD without retrieval-based material optimization, UrbanCAD without lighting estimation, and UrbanCAD in full setting, to evaluate the perception model's performance. Please see the supplementary Section 8.2 for details. We also select 100 frames of

Data		YOLO-Seg		Mask2Former (R50)		Mask2Former (R101)		Mask2Former (SwinL)	
		IOU↑	PQ↑	IOU↑	PQ↑	IOU↑	PQ↑	IOU↑	PQ ↑
Real-world data		68.27	62.91	66.15	38.20	73.04	32.05	86.89	48.99
Wonder3D	Trajectory 1	70.66	**62.60**	61.85	28.40	71.32	41.50	73.15	47.11
	Trajectory 2	71.06	**63.12**	62.58	28.48	71.66	40.41	73.14	46.87
	Trajectory 3	71.08	**62.78**	63.60	29.34	72.04	40.62	73.08	47.74
UrbanCAD (w/o material optimization)	Trajectory 1	69.21	60.06	67.28	32.61	72.89	40.13	**84.33**	**52.66**
	Trajectory 2	68.69	59.88	67.10	32.01	73.14	39.95	**84.89**	**52.54**
	Trajectory 3	67.06	59.07	67.53	32.10	73.42	40.62	**84.63**	**52.67**
UrbanCAD (w/o lighting estimation)	Trajectory 1	73.11	61.19	65.45	33.17	74.09	41.10	82.57	51.34
	Trajectory 2	73.33	61.30	65.46	32.99	74.65	41.13	83.34	51.27
	Trajectory 3	73.22	60.67	65.57	32.89	73.84	41.78	83.28	51.33
UrbanCAD (Ours)	Trajectory 1	**74.08**	61.60	**70.28**	**34.71**	**76.40**	**43.47**	83.83	52.27
	Trajectory 2	**74.15**	62.16	**70.07**	**34.10**	**76.91**	**43.56**	84.45	52.12
	Trajectory 3	**74.02**	61.64	**70.42**	**34.45**	**76.56**	**43.55**	84.23	52.39

Table 2. **Quantitative Comparison** of perception methods on different data. Underline denotes second best.

real-world images with similar vehicle distribution and positions compared to our synthetic scenes. Table 2 shows that the YOLO-seg model and Mask2Former with ResNet [23] backbones achieve better IOU results on synthetic scenarios constructed with the UrbanCAD (ours) vehicle models. Additionally, Mask2Former with ResNet backbones reports a higher PQ value on scenarios constructed by UrbanCAD (ours). These results demonstrate that UrbanCAD (ours) can produce high-quality synthetic data for perception tasks. Interestingly, we find that both material optimization and lighting estimation are essential for constructing synthetic scenarios with a small domain gap. The performance of perception models degrades when these modules are removed. The small drop in PQ value compared to Wonder3D is due to the different geometries between Wonder3D vehicles and UrbanCAD (ours) vehicles, which leads to different GT values. We observe that the Mask2Former model using the large Swin Transformer backbone [40] exhibits strong generalization ability. However, inference speed is crucial for self-driving applications, and the Mask2Former model with the SwinL backbone is limited by low inference speed. Furthermore, performance on real-world data is notably lower because exact real-world data corresponding to our synthetic scenarios are not obtainable, as we modify the ground truth of the background images when inserting the vehicles.

Data	Yolo-Seg		Mask2Former (R101)		Mask2Former (SwinL)	
	IOU↑	iIOU↑	IOU↑	iIOU↑	IOU↑	iIOU ↑
Reference data	75.76	**81.06**	88.89	**96.47**	91.36	**95.67**
OOD data (Ours)	75.08	72.20	87.15	83.46	90.93	85.44

Table 3. **Quantitative Comparison** on reference data and out-of-distribution data generated by UrbanCAD.

Out-of-Distribution Driving Scenarios. Thanks to the high controllability and photorealism of our generated vehicle models, we demonstrate that our method can generate photorealistic corner cases, as shown in Fig. 1 and supplementary Fig. 16, to challenge existing perception models. Since measuring perception results based on windows and tires in safety-critical scenarios is difficult, we focus on the performance of the perception system in door-opening settings. Specifically, we construct five door opening and closing scenarios with a total of 150 frames and test the perception system on these corner-case scenarios and reference images where vehicle doors remain closed. Given that opening and closing vehicle doors lead to small changes in the overall scene, we also report the instance-level IOU. As shown in Table 3, the performance of the self-driving perception system declines rapidly in such out-of-distribution scenarios, despite the same perception model performing well in the in-distribution setting as shown in Table 2. This underscores the importance of constructing urban scenarios with our highly controllable vehicles to test self-driving perception systems.

6. Conclusion and Limitations

In this paper, we aim to create photorealistic and highly controllable 3D vehicle digital twins for constructing challenging and realistic scenarios. Towards this goal, we introduce UrbanCAD, a framework that generates 3D vehicle digital twins with photorealistic appearances and high controllability through CAD model retrieval and optimization. Additionally, by reconstructing the background and environmental lighting, UrbanCAD facilitates the realistic insertion of our generated vehicle models into urban scenes. We demonstrate UrbanCAD's capabilities in producing photorealistic and highly controllable 3D vehicle digital twins, as well as in creating realistic, challenging, and safety-critical scenarios to test the robustness of self-driving perception systems. However, due to the semantically aligned CAD retrieval, the geometries of our created CAD models are not the same as the vehicles in the input image. Besides, our estimated spatially varying environment lighting may be not accurate if the insertion position is far from our fisheye cameras.

References

[1] Adobe stock. https://stock.adobe.com/3d-assets. 2, 3, 6

[2] Blender. https://www.blender.org/. 5

[3] OpenAI Josh Achiam, Steven Adler, Sandhini Agarwal, and etc. Gpt-4 technical report. 2023. 4

[4] Shirzad Amir, Yossi Gandelsman, Shai Bagon, and Tali Dekel. Deep vit features as dense visual descriptors. *ArXiv*, abs/2112.05814, 2021. 1

[5] Armen Avetisyan, Manuel Dahnert, Angela Dai, Manolis Savva, Angel X. Chang, and Matthias Nießner. Scan2cad: Learning cad model alignment in rgb-d scans. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2609–2618, 2018. 3

[6] Mikołaj Bińkowski, Danica J Sutherland, Michael Arbel, and Arthur Gretton. Demystifying mmd gans. *arXiv preprint arXiv:1801.01401*, 2018. 6

[7] Alexey Bokhovkin, Shubham Tulsiani, and Angela Dai. Mesh2tex: Generating mesh textures from image queries. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 8884–8894, 2023. 3

[8] Daniel Bolya, Chong Zhou, Fanyi Xiao, and Yong Jae Lee. Yolact: Real-time instance segmentation. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9156–9165, 2019. 7

[9] Shubhankar Borse, Ying Wang, Yizhe Zhang, and Fatih Porikli. Inverseform: A loss function for structured boundary-aware segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5901–5911, 2021. 4

[10] Angel X. Chang, Thomas A. Funkhouser, Leonidas J. Guibas, Pat Hanrahan, Qi-Xing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, L. Yi, and Fisher Yu. Shapenet: An information-rich 3d model repository. *ArXiv*, abs/1512.03012, 2015. 4

[11] Xinya Chen, Hanlei Guo, Yanrui Bin, Shangzhan Zhang, Yuanbo Yang, Yue Wang, Yujun Shen, and Yiyi Liao. Learning 3d-aware gans from unposed images with template feature field. *ArXiv*, abs/2404.05705, 2024. 1

[12] Bowen Cheng, Ishan Misra, Alexander G. Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1280–1289, 2021. 7

[13] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3213–3223, 2016. 5

[14] Tianyuan Dai, J. Wong, Yunfan Jiang, Chen Wang, Cem Gokmen, Ruohan Zhang, Jiajun Wu, and Fei-Fei Li. Acdc: Automated creation of digital cousins for robust policy learning. 2024. 3, 1

[15] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13142–13153, 2023. 6, 4

[16] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR, 2017. 2, 3

[17] Francis Engelmann, J. Stückler, and B. Leibe. Samp: Shape and motion priors for 4d vehicle reconstruction. *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 400–408, 2017. 3

[18] Adrien Gaidon, Qiao Wang, Yohann Cabon, and Eleonora Vig. Virtual worlds as proxy for multi-object tracking analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4340–4349, 2016. 3

[19] Daoyi Gao, Dávid Rozenberszki, Stefan Leutenegger, and Angela Dai. Diffcad: Weakly-supervised probabilistic cad model retrieval and alignment from an rgb image. *ArXiv*, abs/2311.18610, 2023. 3

[20] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2414–2423, 2016. 5, 2

[21] Can Gümeli, Angela Dai, and Matthias Nießner. Roca: Robust cad model retrieval and alignment from a single image. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4012–4021, 2021. 3

[22] Jianfei Guo, Nianchen Deng, Xinyang Li, Yeqi Bai, Botian Shi, Chiyu Wang, Chenjing Ding, Dongliang Wang, and Yikang Li. Streetsurf: Extending multi-view implicit surface reconstruction to street views. *ArXiv*, abs/2306.04988, 2023. 3

[23] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2015. 8

[24] Zexin He and Tengfei Wang. Openlrm: Open-source large reconstruction models. https://github.com/3DTopia/OpenLRM, 2023. 4

[25] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017. 6

[26] Yicong Hong, Kai Zhang, Jiuxiang Gu, Sai Bi, Yang Zhou, Difan Liu, Feng Liu, Kalyan Sunkavalli, Trung Bui, and Hao Tan. Lrm: Large reconstruction model for single image to 3d. *arXiv preprint arXiv:2311.04400*, 2023. 2, 6, 4

[27] Brian Karis and Epic Games. Real shading in unreal engine 4. *Proc. Physically Based Shading Theory Practice*, 4(3):1, 2013. 5

[28] Alexander Kirillov, Kaiming He, Ross B. Girshick, Carsten Rother, and Piotr Dollár. Panoptic segmentation. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9396–9405, 2018. 6

[29] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and

Ross B. Girshick. Segment anything. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3992–4003, 2023. 3

[30] Abhijit Kundu, Kyle Genova, Xiaoqi Yin, Alireza Fathi, Caroline Pantofaru, Leonidas J. Guibas, Andrea Tagliasacchi, Frank Dellaert, and Thomas A. Funkhouser. Panoptic neural fields: A semantic object-aware neural scene representation. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12861–12871, 2022. 3

[31] Weicheng Kuo, Anelia Angelova, Tsung-Yi Lin, and Angela Dai. Patch2cad: Patchwise embedding learning for in-the-wild shape retrieval from a single image. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 12569–12579, 2021. 3

[32] Beichen Li, Liang Shi, and Wojciech Matusik. End-to-end procedural material capture with proxy-free mixed-integer optimization. *ACM Transactions on Graphics (TOG)*, 42: 1 – 15, 2023. 4

[33] Zhengqin Li, Mohammad Shafiei, Ravi Ramamoorthi, Kalyan Sunkavalli, and Manmohan Chandraker. Inverse rendering for complex indoor scenes: Shape, spatially-varying lighting and svbrdf from a single image. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2472–2481, 2019. 5

[34] Yiyi Liao, Jun Xie, and Andreas Geiger. Kitti-360: A novel dataset and benchmarks for urban scene understanding in 2d and 3d. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3):3292–3310, 2022. 5, 6

[35] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, 2014. 5

[36] Zhi Lin, Bohan Liu, Yi-Ting Chen, David A. Forsyth, Jia-Bin Huang, Anand Bhattad, and Shenlong Wang. Urbanir: Large-scale urban scene inverse rendering from a single video. *ArXiv*, abs/2306.09349, 2023. 3

[37] Carl Lindstrom, Georg Hess, Adam Lilja, Maryam Fatemi, Lars Hammarstrand, Christoffer Petersson, and Lennart Svensson. Are nerfs ready for autonomous driving? towards closing the real-to-simulation gap. *ArXiv*, abs/2403.16092, 2024. 3

[38] Jeffrey Yunfan Liu, Yun Chen, Ze Yang, Jingkang Wang, Sivabalan Manivasagam, and Raquel Urtasun. Real-time neural rasterization for large scenes. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 8382–8393, 2023. 3

[39] Minghua Liu, Ruoxi Shi, Kaiming Kuang, Yinhao Zhu, Xuanlin Li, Shizhong Han, Hong Cai, Fatih Porikli, and Hao Su. Openshape: Scaling up 3d shape representation towards open-world understanding. *Advances in Neural Information Processing Systems*, 36, 2024. 3, 4

[40] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9992–10002, 2021. 8

[41] William Ljungbergh, Adam Tonderski, Joakim Johnander, Holger Caesar, Kalle Åström, Michael Felsberg, and Christoffer Petersson. Neuroncap: Photorealistic closed-loop safety testing for autonomous driving. *ArXiv*, abs/2404.07762, 2024. 3

[42] Xiaoxiao Long, Yuanchen Guo, Cheng Lin, Yuan Liu, Zhiyang Dou, Lingjie Liu, Yuexin Ma, Song-Hai Zhang, Marc Habermann, Christian Theobalt, and Wenping Wang. Wonder3d: Single image to 3d using cross-domain diffusion. *ArXiv*, abs/2310.15008, 2023. 2, 6, 4

[43] Fan Lu, Yan Xu, Guang Chen, Hongsheng Li, Kwan-Yee Lin, and Changjun Jiang. Urban radiance field representation with deformable neural mesh primitives. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 465–476, 2023. 3

[44] Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7210–7219, 2021. 2, 3

[45] Gal Metzer, Elad Richardson, Or Patashnik, Raja Giryes, and Daniel Cohen-Or. Latent-nerf for shape-guided generation of 3d shapes and textures. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12663–12673, 2022. 3, 6, 2, 4

[46] Norman Müller, Andrea Simonelli, Lorenzo Porzi, Samuel Rota Bulò, Matthias Nießner, and Peter Kontschieder. Autorf: Learning 3d object radiance fields from single view observations. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3961–3970, 2022. 2

[47] Pakkapon Phongthawee, Worameth Chinchuthakun, Nontaphat Sinsunthithet, Amit Raj, Varun Jampani, Pramook Khungurn, and Supasorn Suwajanakorn. Diffusionlight: Light probes for free by painting a chrome ball. *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 98–108, 2023. 7, 8

[48] Ava Pun, Gary Sun, Jingkang Wang, Yun Chen, Ze Yang, Sivabalan Manivasagam, Wei-Chiu Ma, and Raquel Urtasun. Lightsim: Neural lighting simulation for urban scenes. *ArXiv*, abs/2312.06654, 2023. 3, 5

[49] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *ArXiv*, abs/2204.06125, 2022. 3

[50] Dillon Reis, Jordan Kupec, Jacqueline Hong, and Ahmad Daoudi. Real-time flying object detection with yolov8. *ArXiv*, abs/2305.09972, 2023. 7

[51] Konstantinos Rematas, Andrew Liu, Pratul P Srinivasan, Jonathan T Barron, Andrea Tagliasacchi, Thomas Funkhouser, and Vittorio Ferrari. Urban radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12932–12942, 2022. 2, 3

[52] Tianhe Ren, Shilong Liu, Ailing Zeng, Jing Lin, Kunchang Li, He Cao, Jiayu Chen, Xinyu Huang, Yukang Chen, Feng Yan, Zhaoyang Zeng, Hao Zhang, Feng Li, Jie Yang,

Hongyang Li, Qing Jiang, and Lei Zhang. Grounded sam: Assembling open-world models for diverse visual tasks. *ArXiv*, abs/2401.14159, 2024. 3, 4

[53] Jay Sarva, Jingkang Wang, James Tu, Yuwen Xiong, Sivabalan Manivasagam, and Raquel Urtasun. Adv3d: Generating safety-critical 3d objects through closed-loop simulation. In *Conference on Robot Learning*, 2023. 3

[54] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and Service Robotics: Results of the 11th International Conference*, pages 621–635. Springer, 2018. 3

[55] Liang Shi, Beichen Li, Miloš Hašan, Kalyan Sunkavalli, Tamy Boubekeur, Radomir Mech, and Wojciech Matusik. Match: Differentiable material graphs for procedural material capture. *ACM Transactions on Graphics (TOG)*, 39(6): 1–15, 2020. 2, 4

[56] Jiali Sun, Tong Wu, and Lin Gao. Recent advances in implicit representation-based 3d shape generation. *Visual Intelligence*, 2:1–13, 2024. 2

[57] Matthew Tancik, Vincent Casser, Xinchen Yan, Sabeek Pradhan, Ben Mildenhall, Pratul P Srinivasan, Jonathan T Barron, and Henrik Kretzschmar. Block-nerf: Scalable large scene neural view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8248–8258, 2022. 2, 3

[58] Jiajun Tang, Yongjie Zhu, Haoyu Wang, Jun Hoong Chan, Si Li, and Boxin Shi. Estimating spatially-varying lighting in urban scenes with disentangled representation. In *European Conference on Computer Vision*, 2022. 7, 8

[59] Maxim Tatarchenko, Stephan R. Richter, René Ranftl, Zhuwen Li, Vladlen Koltun, and Thomas Brox. What do single-view 3d reconstruction networks learn? *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3400–3409, 2019. 3

[60] Mikaela Angelina Uy, Jingwei Huang, Minhyuk Sung, Tolga Birdal, and Leonidas J. Guibas. Deformation-aware 3d model embedding and retrieval. In *European Conference on Computer Vision*, 2020. 3

[61] Jingkang Wang, Ava Pun, James Tu, Sivabalan Manivasagam, Abbas Sadat, Sergio Casas, Mengye Ren, and Raquel Urtasun. Advsim: Generating safety-critical scenarios for self-driving vehicles. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9904–9913, 2021. 3

[62] Jingkang Wang, Sivabalan Manivasagam, Yun Chen, Ze Yang, Ioan Andrei Bârsan, Anqi Joyce Yang, Wei-Chiu Ma, and Raquel Urtasun. Cadsim: Robust and scalable in-the-wild 3d reconstruction for controllable sensor simulation. In *Conference on Robot Learning*, 2023. 2, 3

[63] Zian Wang, Tianchang Shen, Jun Gao, Sheng Yu Huang, Jacob Munkberg, Jon Hasselgren, Zan Gojcic, Wenzheng Chen, and Sanja Fidler. Neural fields meet explicit geometric representations for inverse rendering of urban scenes. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8370–8380, 2023. 3

[64] Yuxi Wei, Zi Wang, Yifan Lu, Chenxin Xu, Changxing Liu, Hao Zhao, Siheng Chen, and Yanfeng Wang. Editable scene simulation for autonomous driving via collaborative llm-agents. *ArXiv*, abs/2402.05746, 2024. 5, 3, 7, 8

[65] Felix Wimbauer, Nan Yang, Christian Rupprecht, and Daniel Cremers. Behind the scenes: Density fields for single view reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9076–9086, 2023. 2, 3

[66] Kai Yan, Fujun Luan, Milovs. Havs.an, Thibault Groueix, Valentin Deschaintre, and Shuang Zhao. Psdr-room: Single photo to scene using differentiable rendering. *SIGGRAPH Asia 2023 Conference Papers*, 2023. 3, 4

[67] L. Yang, Ping Luo, Chen Change Loy, and Xiaoou Tang. A large-scale car dataset for fine-grained categorization and verification. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3973–3981, 2015. 6, 4

[68] Ze Yang, Yun Chen, Jingkang Wang, Sivabalan Manivasagam, Wei-Chiu Ma, Anqi Joyce Yang, and Raquel Urtasun. Unisim: A neural closed-loop sensor simulator. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1389–1399, 2023. 2, 3

[69] Ze Yang, Sivabalan Manivasagam, Yun Chen, Jingkang Wang, Rui Hu, and Raquel Urtasun. Reconstructing objects in-the-wild for realistic sensor simulation. *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11661–11668, 2023. 2

[70] Hu Ye, Jun Zhang, Siyi Liu, Xiao Han, and Wei Yang. Ip-adapter: Text compatible image prompt adapter for text-to-image diffusion models. *ArXiv*, abs/2308.06721, 2023. 4

[71] Yu-Ying Yeh, Zhengqin Li, Yannick Hold-Geoffroy, Rui Zhu, Zexiang Xu, Miloš Hašan, Kalyan Sunkavalli, and Manmohan Chandraker. Photoscene: Photorealistic material and lighting transfer for indoor scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18562–18571, 2022. 2, 3, 4, 5, 6

[72] Yu-Ying Yeh, Jia-Bin Huang, Changil Kim, Lei Xiao, Thu Nguyen-Phuoc, Numair Khan, Cheng Zhang, Manmohan Krishna Chandraker, Carl Marshall, Zhao Dong, and Zhengqin Li. Texturedreamer: Image-guided texture synthesis through geometry-aware diffusion. *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4304–4314, 2024. 3

[73] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4576–4585, 2020. 6, 2, 4

[74] Tao Yu, Runsen Feng, Ruoyu Feng, Jinming Liu, Xin Jin, Wenjun Zeng, and Zhibo Chen. Inpaint anything: Segment anything meets image inpainting. *ArXiv*, abs/2304.06790, 2023. 4

[75] Xianfang Zeng, Xin Chen, Zhongqi Qi, Wen Liu, Zibo Zhao, Zhibin Wang, Bin Fu, Yong Liu, and Gang Yu. Paint3d: Paint anything 3d with lighting-less texture diffusion models. *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4252–4262, 2023. 3, 6, 2, 4

[76] Jason Y. Zhang, Gengshan Yang, Shubham Tulsiani, and Deva Ramanan. Ners: Neural reflectance surfaces for sparse-

view 3d reconstruction in the wild. In *Neural Information Processing Systems*, 2021. 6, 7, 8

[77] Linrui Zhang, Zhenghao Peng, Quanyi Li, and Bolei Zhou. Cat: Closed-loop adversarial training for safe end-to-end driving. *ArXiv*, abs/2310.12432, 2023. 5

[78] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3813–3824, 2023. 4, 5, 3

[79] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 586–595, 2018. 6

[80] Xiaoshuai Zhang, Abhijit Kundu, Thomas Funkhouser, Leonidas Guibas, Hao Su, and Kyle Genova. Nerflets: Local radiance fields for efficient structure-aware 3d scene representation from 2d supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8274–8284, 2023. 3

[81] Xu Zhao, Wen-Yan Ding, Yongqi An, Yinglong Du, Tao Yu, Min Li, Ming Tang, and Jinqiao Wang. Fast segment anything. *ArXiv*, abs/2306.12156, 2023. 5, 3

[82] Hongyu Zhou, Jiahao Shao, Lu Xu, Dongfeng Bai, Weichao Qiu, Bingbing Liu, Yue Wang, Andreas Geiger, and Yiyi Liao. Hugs: Holistic urban 3d scene understanding via gaussian splatting. *arXiv preprint arXiv:2403.12722*, 2024. 2, 3, 5, 6

UrbanCAD: Towards Highly Controllable and Photorealistic 3D Vehicles for Urban Scene Simulation

Supplementary Material

This appendix details our method, implementation, experimental designs, additional experiment results, utilized resources, and broader implications. We first detail how to retrieve and optimize the CAD models in Section 7.1, Section 7.2, Section 7.3, Section 7.4, and Section 7.4, and then we show the process of urban lighting estimation in Section 7.6 and background reconstruction in Section 7.7. In Section 8, we provide details on experiment designs including baselines implementation (Section 8.1), synthetic data generation (Section 8.2), and perception systems implementation (Section 8.3). We also show more results and implementation details of our functionality in (Section 9). Finally, we report additional experiments and analysis in (Section 10).

7. UrbanCAD Implementation Details

7.1. CAD Model Filtering

Our method requires the CAD models to have correct material index assignment to support automatic coloring. However, we observe that in free CAD model libraries, there are small parts of handcrafted CAD models without proper material index designs. To this end, we design a script to filter the unqualified CAD models automatically or with a small amount of user interface based on the material design.

7.2. Pose Matching

Following [11, 14], we choose the CAD model rendering poses based on the DINO [4] feature similarity with the reference image. First, we crop the vehicles from both the reference image $\mathbf{I}_{ref}$ and 360° retrieved CAD model renderings $\{\mathbf{I}_{cad}^k\}_{k=1}^M$, where $M = 360/A$ is the number of rendering views, and resize them to the same resolution. Then, we compute the DINO feature maps [4] for both vehicle image in reference view and CAD models rendering results using the DINO-ViT encoder $\mathcal{E}_{\text{DINO}}$: $\mathbf{F}_{ref} = \mathcal{E}_{\text{DINO}}(\mathbf{I}_{ref})$, $\{\mathbf{F}_{cad}^k\}_{k=1}^M = \mathcal{E}_{\text{DINO}}(\{\mathbf{I}_{cad}^k\}_{k=1}^M)$. Finally, we compute the L2 distances between the $\mathbf{F}_{ref}$ and the $\{\mathbf{F}_{cad}^k\}_{k=1}^M$ and select the rendering that has the minimum L2 distance with the vehicle in the reference view. In our experiment, we find this approach can achieve accurate pose-matching results regardless of appearance and geometry differences between the retrieved CAD models and reference vehicles. The quality results of our pose-matching method are shown in Fig. 8.

7.3. Part-level Material Prior Retrieval

Since the retrieved CAD models usually have an unsatisfactory appearance, simply using Grounded SAM to segment the CAD model renderings will lead to many failure cases. ControlNet can translate primitives like edges into realistic pictures. Therefore, we propose to use ControlNet to augment the CAD model renderings and ensure the accurate segmentation of Grounded SAM. Specifically, we first extract edges from the material index maps rendered in 360°. Then, we input edges into a canny-based pre-trained ControlNet model and obtain the augmented multi-view images. Note that this canny-based ControlNet translation does not affect the position of the components. After that, we use Grounded SAM to segment the 360° augmented images with component text prompts like windows and wheels. Once we get the multi-view segmentation results, we first select the rendering with the highest mean mask confidence. Then, we calculate the material index masks that have an intersection with the segmented mask. We define them as active materials $\mathbf{Mat}_{act}$. We calculate the masks of each active material $\mathbf{Mat}_{act}$ in material index map $\mathbf{M}_{ind}$ and in the Grounded SAM segmentation map $\mathbf{M}_{seg}$. We then compute the IOU between $\mathbf{M}_{ind}$ and $\mathbf{M}_{seg}$. If the IOU is larger than the IOU threshold (we set the IOU threshold as 0.5), the material will be classified into the corresponding component. We illustrate our method in Fig. 10.

7.4. Material Design Merging Using DINO Feature

Since the bodies of some vehicles are composed of many small components in the CAD models, only retrieving and optimizing materials for the largest part will lead to unsatisfactory results. However, Grounded SAM sometimes can't recognize tiny components like vehicle lights. Simply regarding all remaining parts after component recognition as car bodies will also lead to inaccurate material assignment. To this end, we utilize the DINO corresponding points proposed in [4] to merge the small components in the CAD models. Specifically, we first segment the known components in the input image using Grounded SAM. Then, we calculate the corresponding points between the remaining parts in the input images and the CAD model renderings. Since the remaining parts in the input images are the car body, the corresponding parts in the CAD model renderings are the car body as well. Besides, with a suitable setting of corresponding points' numbers, tiny components not belonging to car bodies will not be wrongly merged. During our experiment, this kind of merging produces good mate-

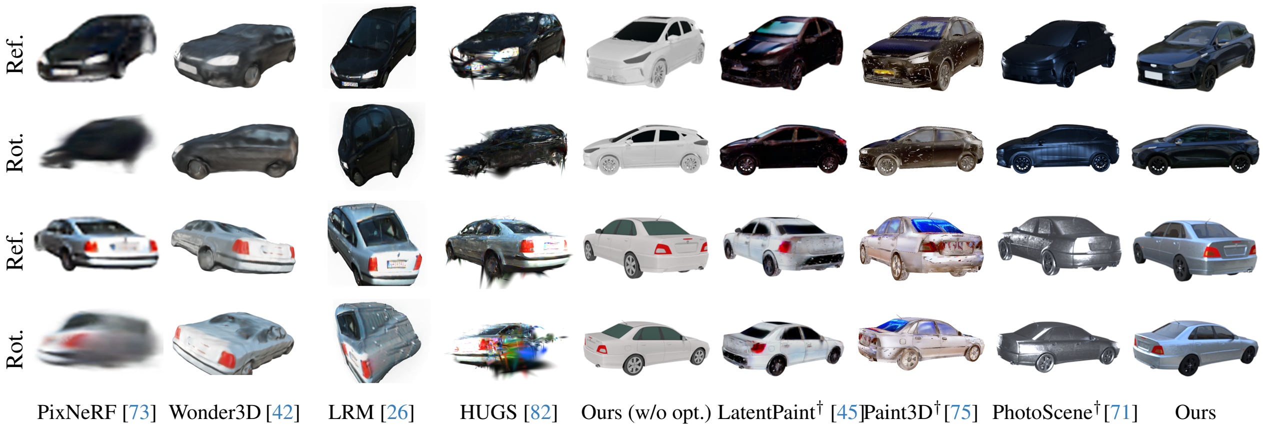


Figure 7. **More qualitative results** on KITTI-360 for novel view synthesis from reference (Ref.) and rotated (rot.) viewpoints.

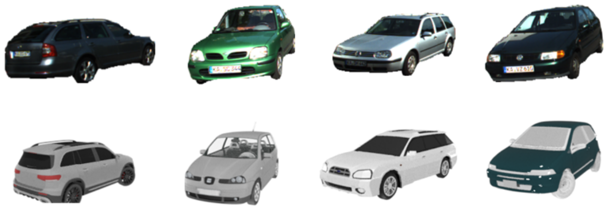


Figure 8. Pose matching results. It shows that the pose of retrieved CAD models (second row) can match accurately with the pose of the input vehicle images (first row) despite the large difference between appearance and geometry.

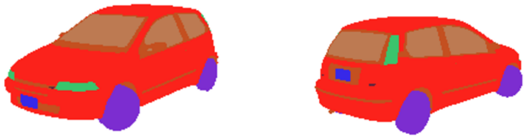


Figure 9. **Symmetric material design**. Different colors represent different material indexes.

rial assignment results on tiny components of CAD models.

7.5. Material Optimization

Since there is no exact correspondence between rendered and reference pixels, we use a part-level loss ℓ_{stat} by minimizing the difference between the mean and variance of the corresponding parts following [71]:

$$\ell_{mean} = |\mu(\mathbf{I}_{ref} \cdot \mathbf{S}_{ref}[\mathbf{c}]) - \mu(\widehat{\mathbf{I}}_{render} \cdot \mathbf{S}_{cad}[\mathbf{c}])| \quad (4)$$

$$\ell_{var} = |\sigma^2(\mathbf{I}_{ref} \cdot \mathbf{S}_{ref}[\mathbf{c}]) - \sigma^2(\widehat{\mathbf{I}}_{render} \cdot \mathbf{S}_{cad}[\mathbf{c}])| \quad (5)$$

$$\ell_{stat} = \ell_{mean} + \ell_{var} \quad (6)$$

where $\widehat{\mathbf{I}}_{render}$ is the CAD model rendering after pose matching, $\mathbf{S}_{ref}[\mathbf{c}]$ and $\mathbf{S}_{cad}[\mathbf{c}]$ are the segmentation masks of the component $\mathbf{c}$ in the reference view and CAD model rendering.

To match the patterns of the reference view, we use a masked VGG loss ℓ_{VGG} using Gram matrices [20] to enhance visual similarity:

$$\ell_{vgg} = |Gram(\mathbf{I}_{ref}, \mathbf{S}_{ref}[\mathbf{c}]) - Gram(\widehat{\mathbf{I}}_{render}, \mathbf{S}_{cad}[\mathbf{c}])| \quad (7)$$

To further match the color of the reference vehicles, we add a masked RGB loss ℓ_{rgb} on the overlap region between components in the reference view and CAD model rendering:

$$\ell_{rgb} = |\mathbf{I}_{ref} \cdot \mathbf{S}_{overlap} - \mathbf{I}_{cad} \cdot \mathbf{S}_{overlap}| \quad (8)$$

The total loss function is shown as below:

$$\ell_{\mathbf{total}} = \lambda_{\mathbf{stat}}\ell_{\mathbf{stat}} + \lambda_{\mathbf{vgg}}\ell_{\mathbf{vgg}} + \lambda_{\mathbf{rgb}}\ell_{\mathbf{rgb}} \quad (9)$$

In our experiment, we set the λ_{stat} to 0.1, the λ_{vgg} to 1, the λ_{rgb} to 1. Note that spatially varying roughness parameters are difficult to optimize from single-view images due to limited highlight observations. Handcrafted procedural material graphs provide photorealistic spatially varying effects, so the roughness parameters of the retrieved material prior are fixed during optimization, as in [71]. Besides, we observe two types of materials with distinct spatially varying effects in car bodies depending on whether the vehicles are painted or not, as shown in Fig. 13. To best fit the observation, we recommend selecting the corresponding car body material prior via the user interface.

7.6. Spatially Varying Lighting Estimation Based on Fisheye Images

As shown in Fig. 14, to obtain spatially varying lighting, we first stitch 2 fisheye images into an LDR panorama. Then

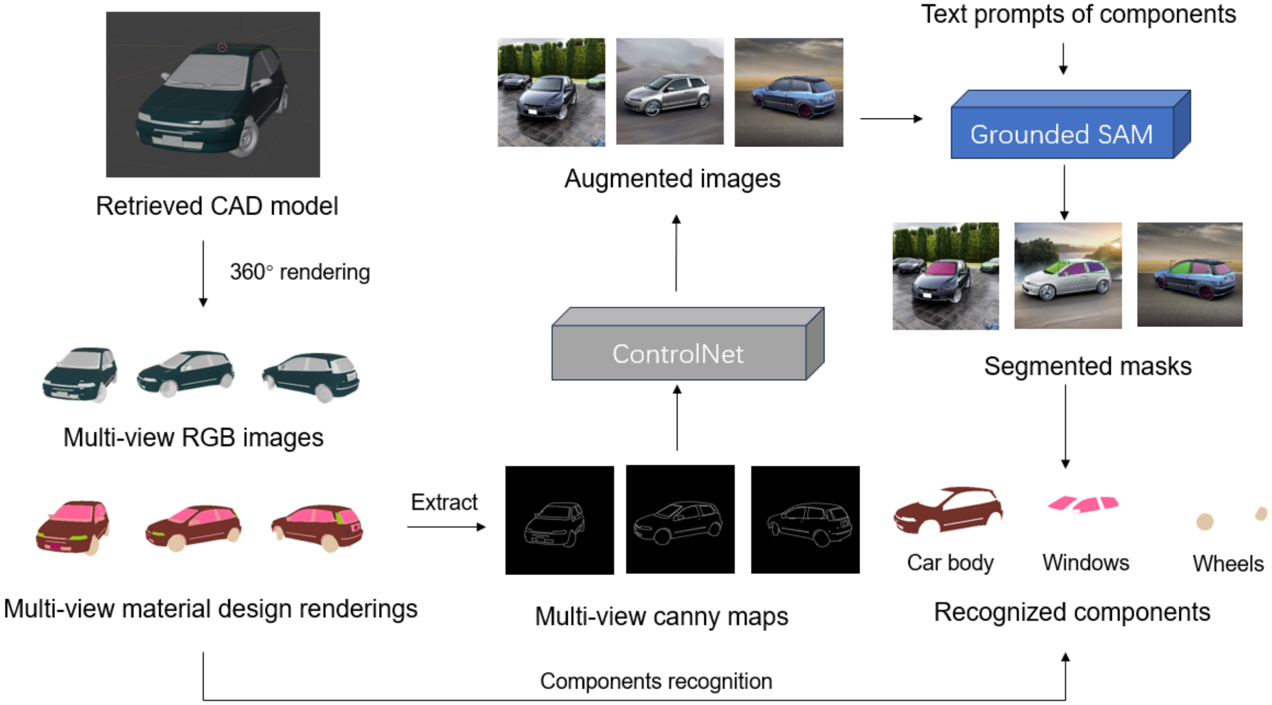


Figure 10. Illustration of Semantic-based Part-aware Material Prior Retrieval Module. To accurately recognize the semantic meaning of the retrieved CAD model for material prior retrieval, we first render the multi-view material designs and convert them to the canny maps. Subsequently, we use the canny-based ControlNet [78] to produce multi-view augmented images. Note that the components' locations in augmented images are aligned with the corresponding material design renderings. After that, we use Grounded SAM [52] and components' names (e.g. windows) to segment the components in the augmented images and obtain multi-view segmented masks with corresponding components' meanings. Finally, we utilize these segmented masks to recognize the material indexes of corresponding components in the material designs.

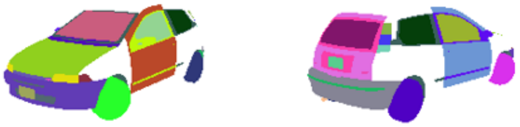


Figure 11. **Disentangled geometry of handcrafted CAD model**. Different colors represent different disentangled geometry.

we crop the upper part of the panorama representing the skydome and feed it into the ChatSim [64] LDR to HDR prediction network. After obtaining the HDR panorama of the sky part, we use FastSAM [81] with text prompts to obtain the ground part. FastSAM selectively ignores detailed pixels, enabling us to separate the clean sky, which could be beneficial to subsequent usage. After performing numerical correction on the LDR image and concatenating it with the previously obtained HDR panorama, we can obtain the local lighting of the current position where the fisheye image is captured.

7.7. Background Reconstruction using 3DGS

We employ the HUGS [82] to reconstruct the background of urban scenes. This process involves utilizing multi-view ap-

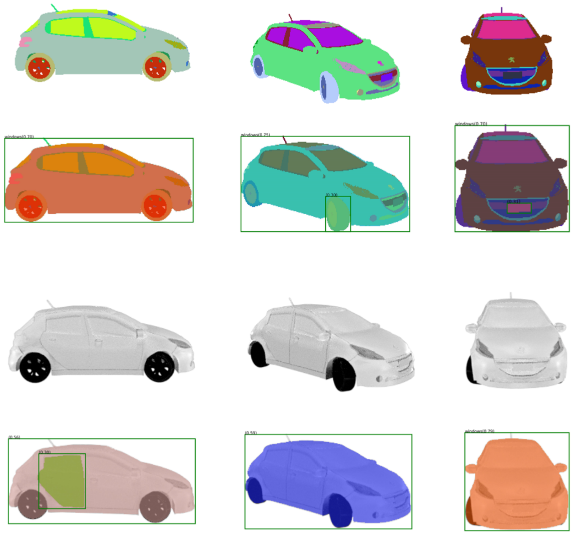


Figure 12. **Quality results** of part-recognition based on randomly colored material design (top) and retrieved CAD renderings without ControlNet augmentation (bottom) with the text prompt of "windows".

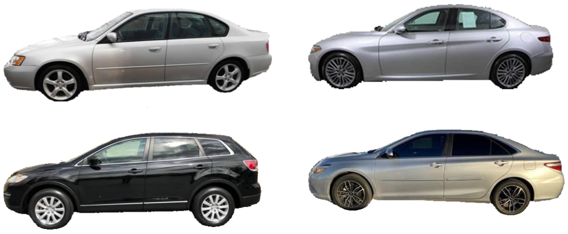


Figure 13. Two types of car body materials with different roughness. Vehicles in the left column are painted and vehicles in the right column are not painted.

pearance observations and pseudo-semantic labels obtained from InverseForm [9]. The HUGS is trained for a total of 30,000 iterations, using two front-perspective cameras and two side-look fisheye cameras in each sequence. Each sequence encompasses 40 frames both prior to and following the target frame. For this reconstruction process, we adhere to the configurations defined by the HUGS. Notably, when converting a static car into our optimized CAD model, we can use inpainting methods [74] for background inpainting to animate the car without leaving holes in the ground.

8. Implementation Details of Experiments

8.1. Baselines Implementation

During appearance comparison, we evaluate 1800 images of 30 models rendered from 360° views and report FID/KID scores comparing with 1800 reference images collected from [67]. Besides, we report the LIPIS scores by comparing the difference between input reference vehicle images and CAD renderings under matched poses.

PixelNeRF. PixelNeRF [73] is an image-based reconstruction method using a conditional implicit function. It supports single-view reconstruction tasks on real-world images. We use the official model pre-trained on ShapeNet [10] to evaluate the performance. We input our single-view images to the PixelNeRF and rendered the reconstructed neural radiance field in 360° with 180 frames.

Wonder3D. Wonder3D [42] is a image-based single view 3D generation method using diffusion priors. We use the official pretrained model to evaluate its single-view generation quality on our input images.

LRM. LRM [26] is a conditional implicit function based single view 3D reconstruction method with large scale training. Since the official LRM implementation hasn't been open-sourced, we use the open-sourced implementation OpenLRM [24]. When inferring on single view image, we simply use its open-sourced pre-trained model.

HUGS. As described in 7.7, we employ HUGS to reconstruct the urban scene, including the target vehicle. The extraction of the target vehicle requires identifying the specific Gaussians that constitute the vehicle. Fortunately, our approach achieved the 3D semantic reconstruction facilitated by HUGS, where every 3D Gaussian possesses a semantic label. This allows for extracting the target vehicle by selecting 3D Gaussians that lie within the bounding box and carry car semantic labels. By manipulating the position and orientation of the 3D Gaussians with a transformation matrix, we can easily manipulate the vehicle representation.

Labor Cost	Method	FID↓	KID↓	LPIPS↓
High	OpenShape [39]	73.10	**0.0453**	0.5761
Middle	UrbanCAD (w/o opt.)	81.05	0.0567	0.6174
Low	OpenShape* [39]	116.36	0.0990	0.6676
Middle	UrbanCAD (Ours)	**62.80**	0.0479	**0.5242**

Table 4. **Quantitative Comparison** on the photorealism of retrieved CAD models with different kinds of materials.

UrbanCAD (w/o opt.). UrbanCAD (w/o opt.) is implemented by directly using the official pre-trained checkpoint of Openshape [39], a multi-modality joint representation method, to retrieve the CAD models from Objaverse [15] dataset according to the input single-view images. Note that while Objaverse includes vehicle CAD models with high-quality texture maps, these require significant manual labor and cannot be optimized to fit observation data. In contrast, our method only requires CAD models with base colors as input, reducing the need for human effort. We further evaluate the quality of these labor-intensive handcrafted textures in Table 4. OpenShape [39] refers to the retrieved CAD models with external handcrafted texture maps, while UrbanCAD (w/o opt.) refers to the CAD models with base colors. OpenShape* [39] denotes the retrieved CAD models without any materials. UrbanCAD (Ours) refers to CAD models with our optimized materials. The results show that our method generates materials that better fit the observations, achieving comparable or superior quality to the labor-intensive handcrafted texture maps.

LatentPaint. LatentPaint [45] is a mesh texturing method using a generative model. When implementing LatentPaint, we found its open-sourced code doesn't support textual inversion. Therefore, we use ChatGPT4 [3] to implement textual inversion by asking ChatGPT4 to estimate the colors of the input vehicles. After we get the colors described in the text, we use the official implementation of LatentPaint to accomplish the mesh texturing task.

Paint3D. Paint3D [75] is a SOTA mesh texturing method using diffusion model. It generates high-resolution textures in a coarse-to-fine manner and supports texutre transfer from a single view image using IP-Adapter [70]. In our implementation, we directly use its open-source code and checkpoints to do the inference.

PhotoScene. Since the procedural graph library used in PhotoScene is different from our method, which may lead to unfairness, we implement PhotoScene on our pre-defined procedural graph library. Specifically, we directly assign the metal material used in our method and further optimize the

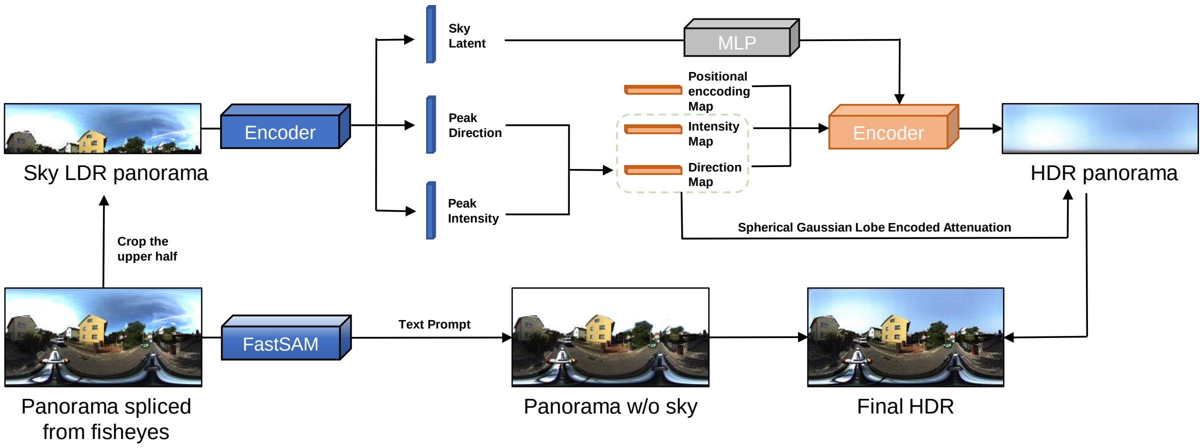


Figure 14. LDR to HDR reconstruction pipeline. The upper half obtains the HDR panorama from the LDR input. The other half stitches the origin panorama with the predicted HDR sky to get the spatially varying lighting

material, since retrieving materials based on visual similarity proposed in Photoscene will lead to severe degradation of appearance.

8.2. Synthetic Data Generation

We utilize a series of 3d bounding boxes to control the movement of vehicles. We construct our synthetic data for self-driving perception system testing in 3 different trajectories as illustrated in Fig. 15. Specifically, trajectory 1 involves vehicles moving normally on the road. Trajectory 2 includes scenarios of vehicles rotating 360°. Trajectory 3 involves vehicles moving in near and partially obscured views, which are typically more challenging for perception models. For each group of synthetic data, there are 60 images for Trajectory 1, 90 images for Trajectory 2, and 120 images for Trajectory 3. When constructing scenarios using UrbanCAD without lighting estimation, we position six uniform point lights along the positive and negative x, y, and z axes.

8.3. Perception Systems Implementation

For YOLOv8 instance segmentation method, we use the official model yolov8n pre-trained on COCO dataset [35]. For the Mask2Former instance segmentation method, we use the official pretrianed models with different backbones on the cityscapes dataset [13].

8.4. Computing Resource

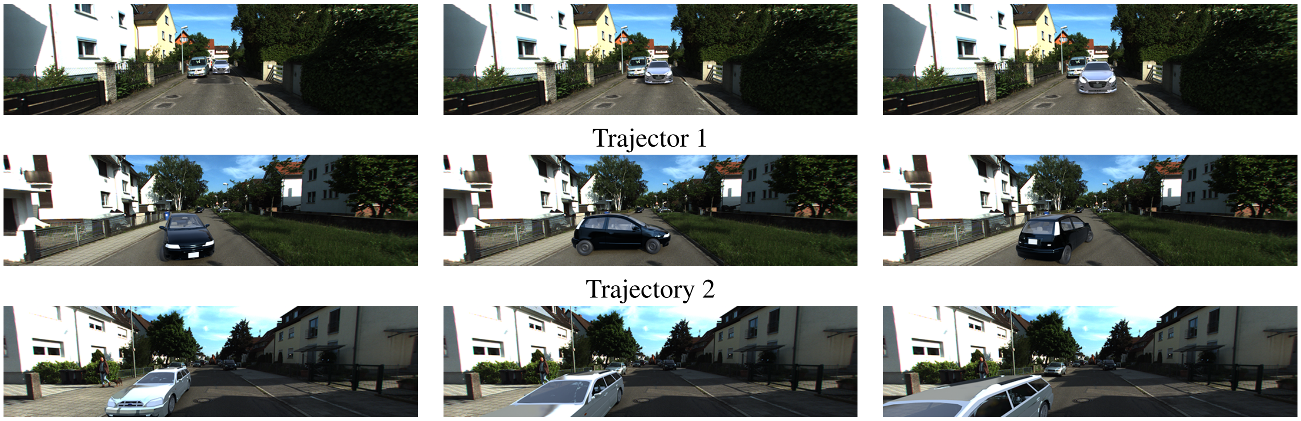
We use a single RTX3090 GPU to perform material optimization. Optimizing a material takes about 35 seconds for 300 optimization epochs.

9. Functionality

Since our created vehicle models are fully controllable, we showcase more editing results including component editing, relighting, material transfer, 360° rotation, and novel view rendering.

9.1. Component Editing

Our produced 3D vehicle models support easy component editing mainly due to the handcrafted disentangled geometry as shown in Fig. 11. Note that complete component editing requires human effort for animation, such as setting joint types and parameters in Blender. Additionally, some retrieved handcrafted CAD models may have merged geometry, for example, the four wheels are merged in one mesh. For these cases, simply hiding other vehicle components and entering the edit mode to separate the wheels by selection in the Blender can solve the problem with small manual efforts. However, we notice that some vehicle CAD models have been post-processed by geometry merging, which means the loss of part controllability. Fortunately, most handcrafted vehicle CAD models in the Objaverse still preserve part controllability without being post-processed, and many post-processed CAD models still have disconnected geometry, which can be manually separated by Blender "Separate Selection" operation after selecting connected geometry ("Select Linked" function in "Select" menu). Besides, more corner case results are displayed in Figure 16 thanks to the representation of CAD models. In addition to the editing results mentioned earlier, we can generate more scenes, using the powerful physical simulation effects in Blender. By assigning physics properties to the vehicle model, we can create collision scenes or even simulate car accidents in Blender.

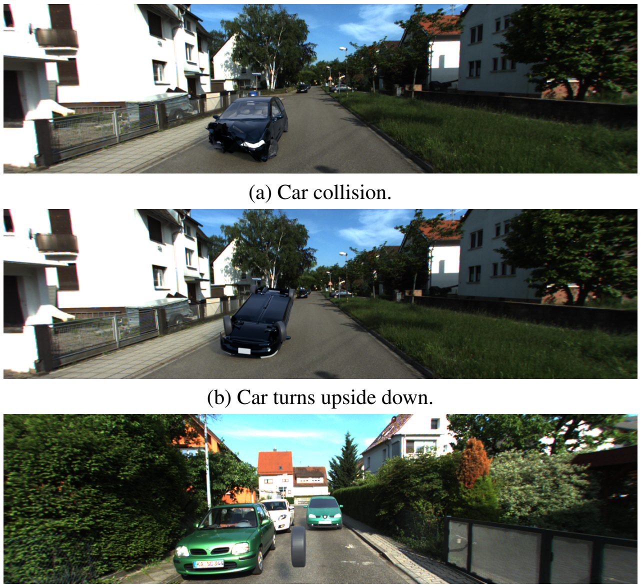


Trajector 1

Trajectory 2

Trajectory 3

Figure 15. Illustration of our synthetic data during self-driving system testing.



(a) Car collision.

(b) Car turns upside down.

(c) Tire rolling.

Figure 16. More corner cases.

9.2. Relighting

Realistic insertion results are shown in Figure 17. We utilize the LDR and HDR pairs from online databases to perform the relighting.

9.3. Material transfer

Material transfer results are shown in Figure 18. Since we have obtained the semantic meaning of CAD model material designs, we can easily transfer the part-aware material from one to another.

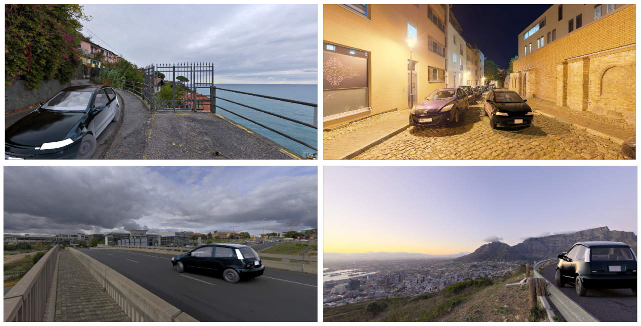
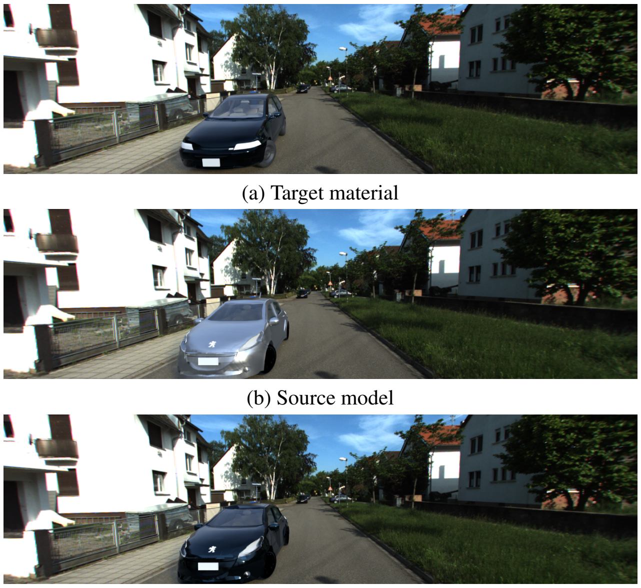


Figure 17. Realistic Insertion.



(a) Target material

(b) Source model

(c) Material transferred model

Figure 18. Material Transfer.

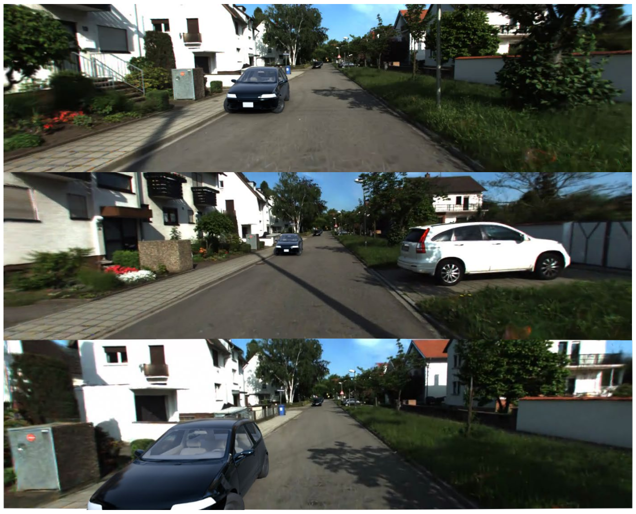


Figure 19. Novel View Synthesis

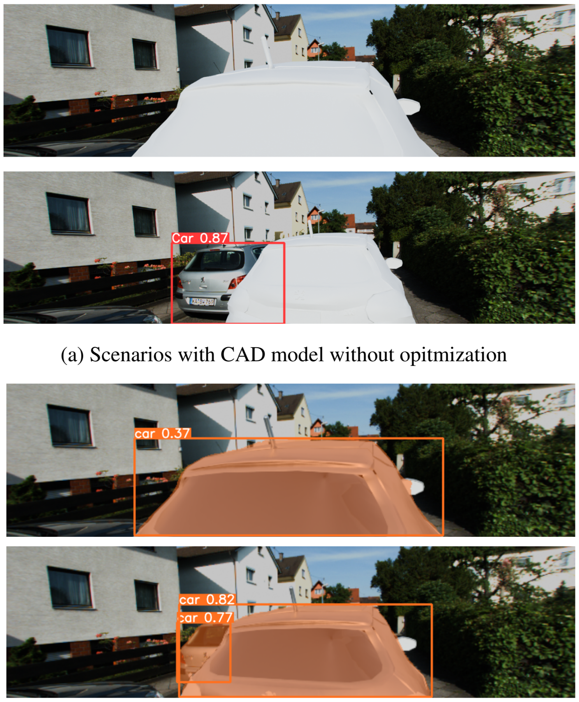
9.4. Novel view rendering

We showcase our novel view rendering results after reconstructing the background using the implicit function and inserting our produced vehicle model, as shown in Fig. 19. Our method can produce high-fidelity rendering results of both background scenes and foreground vehicles under novel viewpoints.

10. Additional Experiments and Analysis

10.1. Lighting Estimation Comparison

We conduct lighting estimation comparison experiments with three baselines as shown in Fig. 22. (1) lighting estimation method using the generative model: DiffusionLight [47]. (2) lighting estimation method with the auto-regressive network: SOLD-Net [58]. (3) lighting estimation method using ray-tracing: ChatSim [64]. For the DiffusionLight, we use the open-sourced official code and checkpoints and take the single-view perspective image as input. For the SOLD-Net, we manually select two points on the ground to mark the area where the network estimates the lighting. After obtaining the output results, we selected the HDR image that closely matched the lighting of the real scene for testing. For the ChatSim [64], we used the view directly ahead of the vehicle as the network input. We also present the quality results of UrbanCAD without lighting estimation in Table 2, where we use six point lights positioned in the positive and negative x, y, and z axis. As demonstrated in the Fig. 22, our method performs better than the baselines, especially in sunny weather where the sun is absent from the perspective images. This is because our fisheye-based method has a 360° view of the environment to accurately capture the location and existence of the sun. However, our method may have limitations in estimating the lighting for objects in the shadow. This is due to the presence of overexposed areas in the fisheye camera's captured image. When these overexposed areas are combined into a panorama, they are given higher brightness, resulting in artifacts when lightening the vehicles in shadow in the final rendering.



(a) Scenarios with CAD model without opitmization

(b) Scenarios with CAD model with opitmization

Figure 20. Quality results on self-driving perception system.

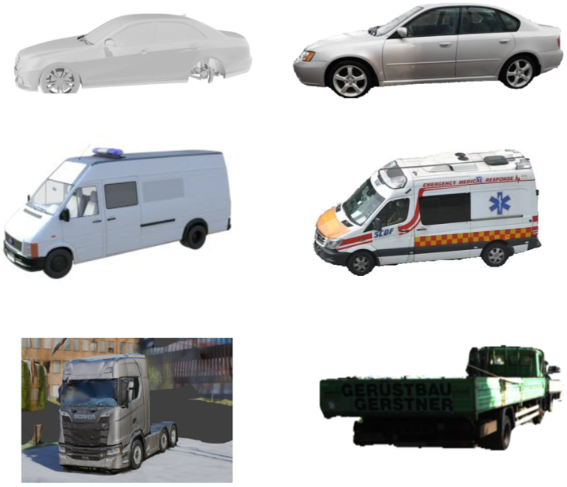


Figure 21. Failure Cases

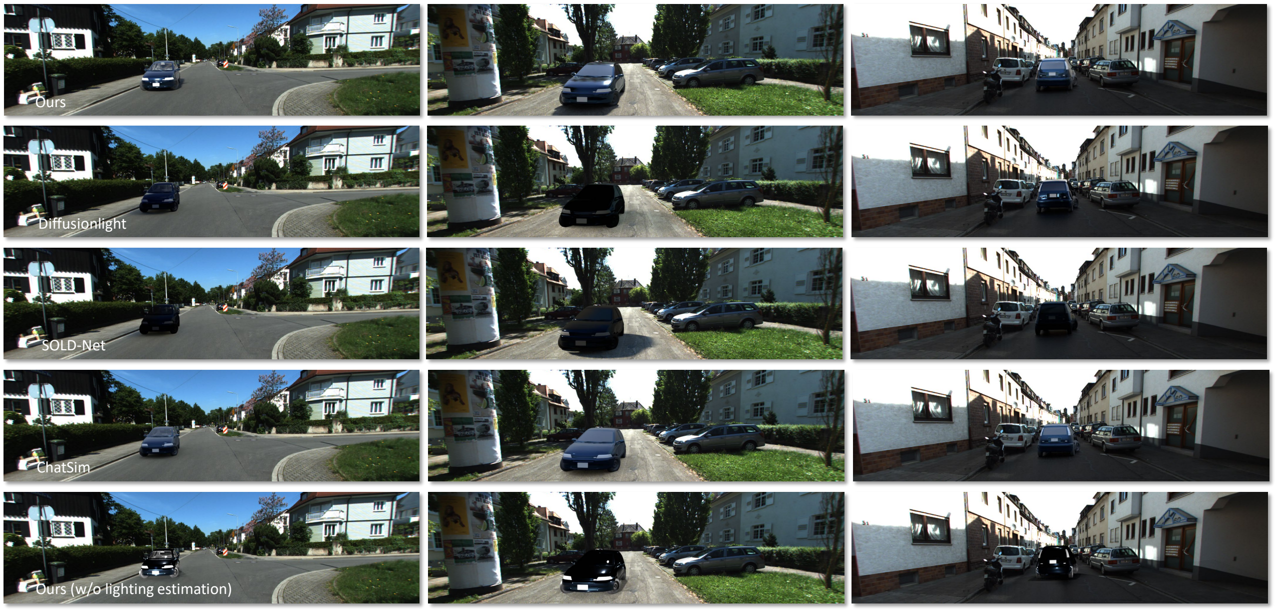


Figure 22. Lighting estimation comparison between ours, DiffusionLight [47], ChatSim [64], SOLD-Net [58], and ours without lighting estimation. In the setup of ours (w/o lighting estimation), the vehicles are illuminated by six point lights positioned along the positive and negative x, y, and z axes. The results show that our method estimates environmental lighting more accurately, particularly in sunny weather.

Method	FID↓	KID↓
NeRS [76] (Surrounding)	110.55	0.0780
NeRS [76] (Partial)	206.46	0.1685
UrbanCAD (Ours)	79.50	0.0530

Table 5. **Quantitative comparison** to NeRS on MVMC dataset in both surrounding and partial observation. Note that our method uses only a single-view image as input.

10.2. Quality results of perception system

In Fig. 20, we show the quality result of different perception results on synthetic data created by UrbanCAD (Ours) and UrbanCAD without material optimization. We find the perception system may fail to work in the synthetic data constructed with the vehicle models with unrealistic materials.

10.3. Failure Cases

we provide failure cases in Fig. 21. Our method may provide unsatisfactory results when the retrieved CAD model is defective (e.g., missing wheels), when the reference vehicle has multiple colors in one component (e.g., ambulance), or when the vehicles in the reference view are rarely seen (e.g., heavy-duty truck).

	Chamfer Dist.↓	Volume IOU↑
Ours	**0.052**	**0.636**
Wonder3D	0.058	0.588

Table 6. Geometry quality

10.4. Geometry quality.

We randomly select 30 vehicles from the ShapeNet dataset, encompassing various types, and retrieve their corresponding models from the Objaverse dataset. We report the Chamfer Distance and Volume IOU in Table 6. Our retrieved models' geometry quality surpasses the reconstruction baseline, Wonder3D [42], as our retrieved CAD models often exhibit better geometry quality in unobservable regions.

11. Broader Impact

UrbanCAD may help the development of self-driving simulation technology, which can further encourage the development of the self-driving industry. However, our method may be used to create some false urban scenes, leading to some social problems.