GeoSim: Realistic Video Simulation via Geometry-Aware Composition for Self-Driving

Yun Chen[1*] Frieda Rong[1,3*] Shivam Duggal[1*] Shenlong Wang[1,2] Xinchen Yan[1]
Sivabalan Manivasagam[1,2] Shangjie Xue[1,4] Ersin Yumer[1] Raquel Urtasun[1,2]
[1]Uber Advanced Technologies Group [2]University of Toronto
[3] Stanford University [4]Massachusetts Institute of Techonology
{chenyuntc, shivamduggal.9507, skywalkeryxc, meyumer}@gmail.com,
rongf@cs.stanford.edu, {slwang, manivasagam, urtasun}@cs.toronto.edu, sjxue@mit.edu

Abstract

Scalable sensor simulation is an important yet challenging open problem for safety-critical domains such as self-driving. Current works in image simulation either fail to be photorealistic or do not model the 3D environment and the dynamic objects within, losing high-level control and physical realism. In this paper, we present GeoSim, a geometry-aware image composition process which synthesizes novel urban driving scenarios by augmenting existing images with dynamic objects extracted from other scenes and rendered at novel poses. Towards this goal, we first build a diverse bank of 3D objects with both realistic geometry and appearance from sensor data. During simulation, we perform a novel geometry-aware simulation-by-composition procedure which 1) proposes plausible and realistic object placements into a given scene, 2) renders novel views of dynamic objects from the asset bank, and 3) composes and blends the rendered image segments. The resulting synthetic images are realistic, traffic-aware, and geometrically consistent, allowing our approach to scale to complex use cases. We demonstrate two such important applications: long-range realistic video simulation across multiple camera sensors, and synthetic data generation for data augmentation on downstream segmentation tasks. Please check https://tmux.top/publication/geosim/ for high-resolution video results.

1. Introduction

Walking along an empty pavement on a silent Sunday morning, one can easily fantasize how busy it could look during rush hour on a weekday, or how a parked car might look when driving on a different street. Humans are capable of recreating the experience of visually perceiving objects and scenes to generate new visual data in their minds. Such an ability allows us to formulate novel scenarios and synthesize events in our heads without experiencing it directly.

Researchers have devoted significant effort towards enhancing computers with the capability of creating pictures by replicating visual content [68]. This brings immense value to many industries, such as film making, robot simulation, augmented reality, and teleconferencing. In the literature, two main paradigms exist: *computer graphics* approaches and *image editing* methods. Computer graphics models the image generation process with physics, by first creating a virtual 3D world and then mimicking how light is transmitted within the world to produce a realistic scene rendering.

To produce visually appealing results, physics-based rendering requires a significant amount of computing resources, costly manual asset creation, and physical modeling [1]. Images produced by existing real-time rendering engines [22, 53, 19], still have a significant realism gap, reducing their impact in robot simulation and data augmentation for training. Data-driven image editing methods such as image composition [38, 43, 16, 20, 5] and generative image synthesis [72, 75, 71, 11, 35, 56] have received significant attention over the past few years. They focus on pushing realism through generative models trained from large-scale visual data. However, most of the efforts do not correspond to an underlying realistic 3D world, and as a consequence, the generated 2D contents are not directly useful for applications such as 3D gaming and robot simulation.

In this paper, we propose *GeoSim*, a realistic image manipulation framework that inserts dynamic objects into existing videos. GeoSim exploits a combination of both data-driven approaches and computer graphics to generate assets inexpensively while maintaining high visual quality through physically grounded simulation. In particular, by leveraging low-cost bounding box annotations and sensor data captured

*Equal Contribution

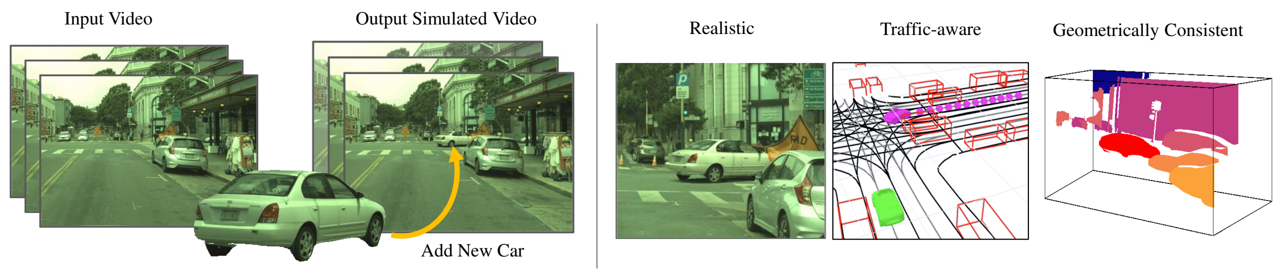

Figure 1: **Realistic video simulation via geometry-aware composition for self-driving.** We proposed a novel data-driven image manipulation approach that inserts dynamic objects into existing videos. Our resulting synthetic video footages are highly realistic, layout-aware, and geometrically consistent, allowing image simulation to scale to complex use cases.

by a self-driving fleet driving around multiple U.S. cities, GeoSim builds a fully-textured large-scale 3D assets bank. While self-driving data is widely available [25, 9, 7, 64], it is non-trivial to automatically build these assets due to the sparsity of the 3D observations, occlusions, shadows, limited viewpoints, and lighting changes. Our asset reconstruction is robust to these challenges, as we ensure consistency across multiple observations in time and learn a strong shape prior to regularize our assets. GeoSim then exploits the 3D scene layout (from high-definition (HD) maps and LiDAR data) to add vehicles in plausible locations and make them behave realistically by considering the full scene. Finally, using this new 3D scene, GeoSim performs image-based rendering to properly handle occlusions, and neural network-based image in-painting to ensure the inserted object seamlessly blends in by filling holes, adjusting color inconsistencies due to lighting changes, and removing sharp boundaries.

Using GeoSim, our resulting synthetic images and video footages are realistic, dynamically plausible, and geometrically consistent. We showcase two important applications: long-range realistic video simulation across multiple camera sensors and synthetic labeled data generation for training self-driving perception algorithms. Our approach outperforms prior work on both qualitative and quantitative realism metrics. We also see significant gains on perception performance when leveraging GeoSim images. These experiments suggest the potential of GeoSim for a plethora of applications, such as realistic safety verification, data augmentation, Sim2Real, augmented reality, and automatic video editing.

2. Related Work

Simulation for Robot Learning: Sensor simulation has received wide attention in the literature [60, 19, 59, 2, 61, 62, 6, 17, 74, 41, 62, 47] for its applications in training and evaluating robotic agents. Sensor simulation systems should be efficient and scalable in order to enable such applications. Many automatic approaches [61, 62, 17, 74, 41] have been proposed to generate indoor environments. Unconstrained outdoor scenes such as the urban driving setting tackled here bring additional challenges due to the scale of the scene, weather, lighting, presence of fast moving objects, and large viewpoint changes arising from sensor motion. In the context of autonomous driving, simulation engines [19, 60, 59] based on rendered 3D models allow the combinatorial generation of scenarios with varying configurations of vehicle attributes, traffic, and weather conditions. However, these methods often have limited diversity in scene content due to the manual design of 3D assets and still have a Real2Sim gap. Data-driven sensor simulation offers a scalable alternative that can capture the complexity of the real world. Many methods [46, 23, 67, 76, 4, 52] have been proposed to directly leverage real-world data for sensor simulation in the autonomous driving domain, typically by augmenting existing recorded data to generate corresponding sensor measurements for novel scene configurations. However, previous works either focus on LiDAR [52, 23, 67], rely on CAD model registration, constraining the set of dynamic objects that can be simulated [46], or require additional effort to scale to high-resolution images [76]. In contrast, we combine data-driven simulation techniques with the image-based rendering techniques in simulation engines. This enables us to construct a scalable, geometrically consistent, and realistic camera simulation system.

Image Synthesis and Manipulation: Image synthesis and manipulation methods offer another route to sensor simulation. Existing work mainly focused on generating 2D images from intermediate representations including scene graphs [35, 30], surface normal maps [72], semantic segmentations [32, 82, 11, 71, 58, 56, 54], and images with different styles [37]. These methods create high-resolution images but with noticeable artifacts in texture and object shape. Rather than generating the full image in one shot, [49, 29, 44] utilize a conditional image generator for scene manipulation. In particular, [49] proposed a spatial-transformer GAN that overlays the target objects on top of existing scene layouts by iteratively adjusting 2D affine transformations. [29] introduced a hierarchical image generation pipeline that is capable of inserting and removing one object at a time. This

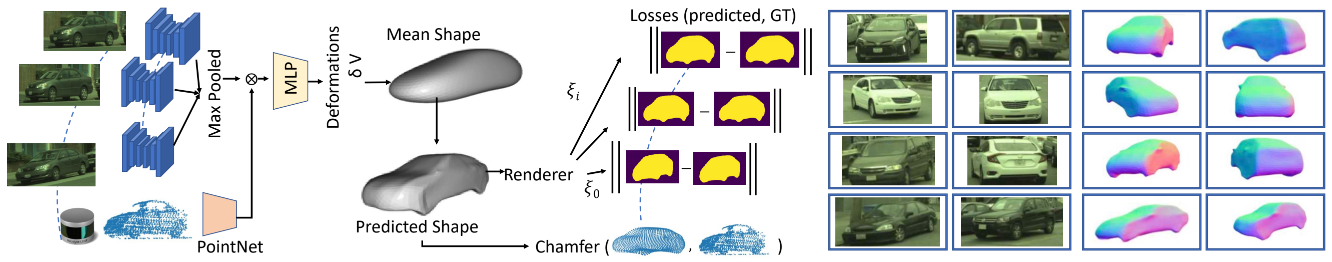

Figure 2: **Realistic 3D assets creation.** Left: multi-view multi-sensor reconstruction network; Right: 3D asset samples. For each sample we show one of the source images and the 3D mesh.

improves realism, but using purely a network-based image synthesis approach has difficulty handling complex physics such as lighting changes. [5] attempts to combine data driven approaches with graphics knowledge, using an image-based neural renderer and image decomposition to improve the synthetic result. Our work builds on this direction of leveraging graphics with real world data. We perform image-based rendering and neural in-painting to adjust for differences between the original image and the image texture of the inserted actor. Furthermore, GeoSim is 3D-layout-aware, allowing for controllable and realistic scene modification.

Video Synthesis and Manipulation: Image simulation alone is insufficient for generating new scenarios with realistic video. One way prior works have extended image synthesis approaches to video generation is by including the past and adding regularization to ensure temporal consistency for realistic object motion. Conditional video generation methods [70, 51, 8, 24] take sequential semantic masks, depth maps or trajectory pose data as inputs, which can then be semantically modified to generate the current video frame. [21] performs 2D-aware image composition via generative modeling of objects and learned dynamics. Automatic video manipulation approaches [45, 31] insert foreground objects or object videos into existing videos in a seamless manner. Unlike most prior work, our image-composition approach is 3D-layout-aware and handles occlusions. Thus, by combining our image composition with automatic trajectory generation methods [63, 66], we easily extend to automatic and scalable controllable video simulation with high realism.

3D Reconstruction and View Synthesis: Our neural network-based 3D asset creation step reconstructs 3D shape from camera imagery and LiDAR in order to synthesize novel views of dynamic objects. View synthesis and 3D reconstruction are well-studied open problems [68], with varying approaches on the relationship between geometry and appearance and possible geometric representations. Image-based rendering methods [27] focus on combining 2D textures to directly render novel views. Appearance flow-based approaches [81, 55, 65] seek to learn unconstrained pixel-level displacements, whereas [13, 83] encode geometric information in latent representations and [36] takes advantage of strong shape priors. Recently, advancements in differentiable rendering [50, 39] and open-source libraries have enabled classical graphics rendering to serve as an optimizable module, allowing for better learning of 3D and visual representations [12, 79].

3. Realistic 3D Assets Creation

In this paper we propose a novel image manipulation approach that inserts dynamic objects into an existing video footage and generates a high-quality video of the augmented scene that is geometrically and semantically consistent with the scene. Key to the success of such an approach is the availability of realistic 3D assets that contain accurate pose, shape and texture. Here we argue that rather than using artists to create these assets, we can leverage data captured by self-driving vehicles to reconstruct the objects around us. This is a much more scalable approach, as many self-driving datasets are available [9, 64, 25], each containing many thousands of unique assets that could potentially be reconstructed. In Sec. 3.1 we first describe how we leverage both LiDAR and camera sensor data from multiple viewpoints to generate realistic 3D vehicle assets using an asset reconstruction network. Sec. 3.2 describes our self-supervised learning procedure to train the network.

3.1. Multi-Sensor 3D Asset Reconstruction

Reconstructing 3D dynamic objects in the wild is challenging: dynamic objects are often partially observed due to the sparsity of the sensor observations and occlusions, they are seen from a limited set of viewpoints, and they appear distorted due to lighting and shadows. To tackle these challenges, we propose a novel, learning-based, multi-view, multi-sensor reconstruction approach for 3D dynamic objects that does not require any ground-truth 3D-shape for training. Instead, we exploit weak annotations in the form of 3D bounding boxes, which are readily available in most self-driving datasets.

More formally, let $\{\mathbf{B}_{i,j}\}_{\forall j}$ be the set of 3D bounding boxes where the i-th object is visible over j views in the recorded snippet. Let $\{\mathbf{I}_{i,j}\}_{\forall j}$ be the associated set of cropped images, and $\{\mathbf{X}_{i,j}\}_{\forall j}$ be the associated set of li-

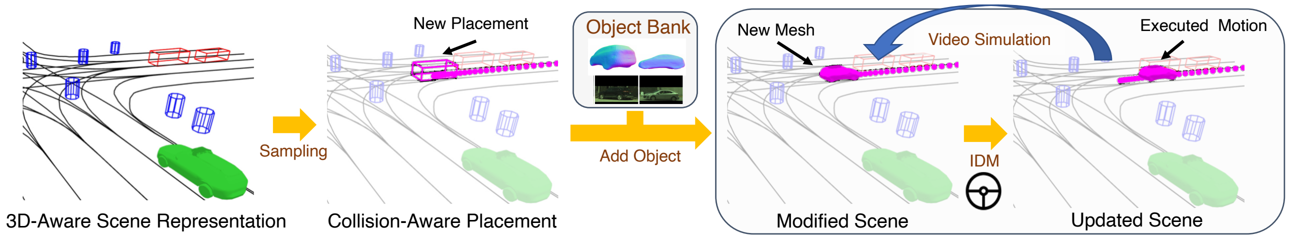

Figure 3: **3D-aware object placement, segment retrieval, and temporal simulation.**

dar points recorded inside $\{\mathbf{B}_{i,j}\}$, transformed to a single canonical frame and let $\mathbf{X}_i$ be the set of aggregated LiDAR points across all views. Our 3D reconstruction network then processes the LiDAR points and image inputs in two streams that are later fused to produce the shape of the object. We refer the reader to Fig. 2 for an illustration. We represent the shape as a 3D mesh $\mathbf{M}_i = \{\mathbf{V}_i, \mathbf{F}_i\}$ where $\mathbf{V}_i$ and $\mathbf{F}_i$ are the faces and vertices of the mesh, respectively. In addition, we also store $\{\mathbf{I}_{i,j}, \mathbf{S}_{i,j}\}_{\forall j}$ to encode object appearance, where $\mathbf{S}_{i,j}$ is the extracted object's silhouette obtained from a pre-trained instance segmentation model [40]. We use this later on to perform novel-view warping.

Network Architecture: Our backbone consists of two submodules. A convolutional network takes each cropped camera image as input and outputs a corresponding feature map. The feature maps from multiple cameras are then aggregated into a one-dimensional latent representation using max-pooling. A similar latent representation is extracted from the input LiDAR point cloud using a PointNet network [57]. The LiDAR and camera features are then passed through an MLP to output the final shape. Instead of employing a learned PCA shape space from CAD models to predict the shape of cars [42], we take inspiration from [36] and parameterize the 3D shape as a category-specific canonical mean shape with per-vertex deformations. This parameterization encodes a categorical prior and ensures the completeness of the shape under partial observations.

3.2. Self-Supervised Learning

Note that we do not have supervision about the shape. We thus train our approach end-to-end in an self-supervised manner to obtain the parameters of the reconstruction network and the mean shape. Our training objective encodes the agreement between the 3D shape and the camera and LiDAR observations, as well as a regularization term.

$$\ell_{\text{total}} = \sum_i \{\ell_{\text{sil}}(\mathbf{M}_i; \mathbf{P}_i, \mathbf{I}_i) \; + \; \ell_{\text{lidar}}(\mathbf{M}_i; \mathbf{X}_i) \; + \; \ell_{\text{reg}}(\mathbf{M}_i)\}$$

where i ranges over all the training objects. The *silhouette loss* measures the consistency between the 2D silhouette (automatically generated using the state-of-the-art object segmentation method PointRend [40]) and the silhouette of the rendered 3D shape.

$$\ell_{\text{sil}}(\mathbf{M}_i; \mathbf{P}_i, \mathbf{I}_i) = \sum_j \|\mathbf{S}_{i,j} - \tau(\mathbf{M}_{\mathbf{i},\mathbf{j}}, \mathbf{P}_{\mathbf{i},\mathbf{j}})\|_2^2$$

where $\mathbf{S}_{\mathbf{i},\mathbf{j}} \in \mathbb{R}^{H\times W}$ is the 2D silhouette, j ranges over multiple views, and $\tau(\mathbf{M}, \mathbf{P})$ is a differentiable neural rendering operator [12] that renders a differentiable mask on the camera image given a projection matrix $\mathbf{P}$. The *LiDAR loss* represents the consistency between the accumulated LiDAR point cloud and the mesh vertices, defined as the asymmetric Chamfer distance

$$\ell_{\text{lidar}}(\mathbf{M}_\mathbf{i}, \mathbf{X}_\mathbf{i}) = \sum_{\mathbf{x}\in\mathbf{X}_i} \min_{\mathbf{v}\in\mathbf{V}_i} \|\mathbf{x} - \mathbf{v}\|_2^2$$

In addition, we also minimize a set of *regularizers* to enforce prior knowledge over the predicted 3D shape, namely local smoothness on the vertices as well as normals. This includes 1) a Laplacian regularization which preserves local geometry and prevents intersecting faces; 2) mesh normal regularization which enforces smoothness of local surfaces; 3) edge regularization which penalizes long edges. Please refer to supp. material for details.

4. Geometry-Aware Image Simulation

Here we describe our image simulation by composition approach that places novel objects into an existing 3D scene and generates a high-quality video sequence of the composition. Our approach takes as input camera video footage, LiDAR point clouds, and an HD map in the form of a lane graph and automatically outputs a video with novel objects inserted into the scene. Note that the input sensory data and HD maps we employ are readily available in most self-driving platforms, which are the application domain we tackle in this paper. Importantly, our simulation takes into account both geometric occlusions by other actors and the background, plausibility of the locations and motions as well as the interactions with other dynamic agents and thus avoids collision for the newly inserted objects.

Towards this goal, we first infer the location of all objects in the scene by performing 3D object detection and tracking [48]. For each new object to be inserted we select where to place it as well as which asset to use based on the HD map and the existing detected traffic. We then utilize an intelligent traffic model for our newly placed object such that its motion is realistic, takes into account the interactions with other actors and avoids collision. The output of this process defines the new scenario to be rendered. We then use a novel-view rendering with 3D occlusion reasoning w.r.t. all elements in the scene, to create the appearance of the novel objects in the new image. Finally, we utilize a neural network to fill in the boundary of the inserted objects, create any missing texture and handle inconsistent lighting. Fig. 1 outlines our approach.

4.1. Scenario Generation

We want to place new objects in existing images such that they are plausible in terms of their scale, location, orientation and motion. Towards this goal, we design a 3D sampling procedure, which takes advantage of the priors we have about how vehicles behave in our cities. Note that it is difficult to achieve similar levels of realism with 2D object insertion. We thus exploit HD maps that contain the location of the lanes in bird's eye view (BEV), and parameterize the object placement as a tuple (x, y, θ) defining the object center and orientation in BEV, which we later convert to a 6DoF pose using the local ground elevation.

Note that our object samples should have realistic physical interactions with existing objects, respect the flow of traffic, and be visible in the camera's field of view. To achieve this, we randomly sample a placement (x, y) from the lane regions lying within the camera's field of view and retrieve the orientation from the lane. We reject all samples that result in collision with other actors or background objects. The aforementioned process provides the placement of the object in the initial frame. To simulate plausible placements over time for video simulation, we use the Intelligent Driver Model (IDM) [63, 66] fitted to a kinematic model following [26], to update the simulated object's state for realistic interactions with surrounding traffic. Fig. 3 depicts the full procedure of placement and kinematics simulation.

So far we have selected where to place an object and how is going to move, but we still need to select which object to place. In order to minimize the artifacts when rendering our assets, we propose to retrieve objects from the asset bank that were viewed with similar viewpoints and distance to the camera in the original footage. The former avoids the need to deal with large unseen object regions while the latter avoids utilizing assets that have been captured at lower resolution. Please refer to the supp. for the specific scoring criteria. Objects are then sampled (as opposed to a hard max) according to a categorical distribution weighted by their inverse score. Once a segment is retrieved for a desired placement, we perform collision checking using the retrieved object shape to ensure that the placement is valid.

4.2. Occlusion-Aware Neural Rendering

Now that we have selected a source object and its corresponding camera image based on the segment retrieval mechanism defined above, we proceed to render this source object into the target scene. Since the object's target pose might differ from the original observed poses, we cannot simply paste the image segment from the source to the target. Thus we proposed to utilize the asset's 3D shape to warp the source to the novel target view.

Novel-view Warping: Let $\mathbf{M}$ be the selected object's 3D mesh, $\mathbf{I}_{\text{s}}$ be the source object's camera image, and $\mathbf{P}_{\text{s}}/\mathbf{P}_{\text{t}}$ be the source/ target camera matrices. We first render $\mathbf{M}_{\text{s}}$ at the selected target viewpoint to generate the corresponding target depth map, $\mathbf{D}_{\text{t}}$. Then using the rendered depth map and source camera image $\mathbf{I}_{\text{s}}$, we generate the object's 2D texture map using the inverse warping operation [69, 33] as:

$$\mathbf{I}_{\text{t}} = \mathbf{I}_{\text{s}}(\pi(\pi^{-1}(\mathbf{D}_{\text{t}}, \mathbf{P}_{\text{t}}), \mathbf{P}_{\text{s}})) \text{ , where } \quad \mathbf{D}_{\text{t}} = \psi(\mathbf{M}, \mathbf{P}_{\text{t}}),$$

ψ is a differentiable neural renderer [12] that produces a depth image given the 3D mesh $\mathbf{M}$ and camera matrix $\mathbf{P}$; π is the perspective projection and π^{-1} is the inverse projection that takes the depth image and camera matrix as input and outputs the 3D points.

Shadow Generation: Inserting an object into a scene will not only change the pixels where the object is present, but can also change the rest of the scene (i.e. shadows and ambient occlusion). We improve the perceptual quality of the image by approximating these effects with image based rendering. While recent works [73, 78] learn shadow synthesis from scene context with a neural network, we render shadows with a graphics engine as geometry is available. To estimate the shadow casted by each inserted object, we construct a virtual scene with the inserted object and a ground plane and exploit image-based rendering [18], where the environment light comes from a real-world HDRI. We render the scene with and without the inserted objects, and add the shadow by blending in the background image intensities with the ratio of the two rendered images intensities. As lighting estimation by manually waving a shadow-casting stick [15] is not applicable, we select a cloudy HDRI to cast shadows. In practice, we find this produces reasonable results. Please refer to the supp. for illustration.

Occlusion Reasoning: An inserted object must respect occlusions from the existing scene elements. Vegetation, fences, and other dynamic objects, for example, may have irregular or thin boundaries, complicating occlusion reasoning. We employ a simple strategy to determine occlusions of the inserted objects and their shadow in the target scene by comparing their depth w.r.t the depth map of the existing 3D scene (see Fig. 4). To achieve this, we first estimate the target image's dense depth map through a depth completion

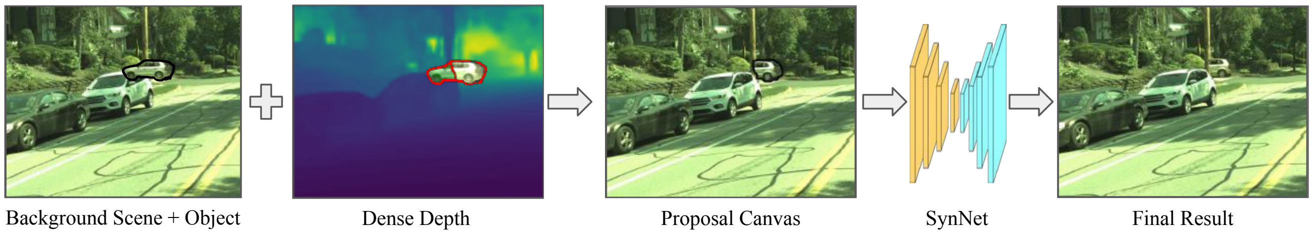

Figure 4: **Geometry-aware composition with occlusion reasoning followed by an image synthesis module.**

network [14]. The input is the RGB image and a sparse depth map acquired by projecting the LiDAR sweep onto the image. Using the rendered depth of the object, the occlusion mask is then computed by evaluating for each object pixel if the target image's depth is smaller than the corresponding object pixel's depth.

Post-Composition Synthesis: After occlusion reasoning, the rendered image may still look unrealistic as the inserted segment may have inconsistent illumination and color-balancing w.r.t the target scene, discrepancies at the boundaries, and missing regions that were not available in the source view. To solve these issues, we leverage an image synthesis network to naturally blend the source segment to the target scene (see Fig. 4). Our network takes the target background image $\mathbf{B}_t$, rendered target object $\mathbf{I}_t$ as well as the object binary silhouette $\mathbf{S}_t$ as input, and outputs the final image that naturally composites the background and rendered object. Our synthesis network architecture is similar to [77], which is a generative image in-painting network except that we take the rendered object mask as additional input. Our network is trained using images with instance segmentation masks inferred by [40] in the target scene. Data augmentation, including random occlusion, color jittering, random contrast and saturation is applied to mimic the differences among real-world images. Two loss functions are adopted, namely perceptual loss [34] to ensure the generated output's fidelity, as well as GAN loss to boost the realism of the in-painted region as well as the lighting consistency. Please refer to supp. for more details.

5. Experimental Evaluation

In this section we first introduce our experimental setting. We then compare GeoSim against a comprehensive set of image simulation baselines in visual realism through perceptual quality scores and human A/B tests, and in downstream tasks such as semantic segmentation. We also showcase our method generating multi-camera and temporally consistent video simulation. While our method can be adapted to handle most rigid objects, in our experiments we showcase vehicles, the most relevant objects in self driving.

5.1. Experimental Details

We utilize two large-scale self-driving datasets (UrbanData and Argoverse [9]) to showcase GeoSim.

UrbanData: We collected a real-world dataset by having a fleet of self-driving cars drive in two major cities in North America. We labeled 16,500 snippets, where each snippet contains 25 seconds (~250 frames, sampled at 10Hz) of video with 7 cameras, a 64-beam LiDAR sensor, and HD maps. We use 4500 for reconstruction and synthesis network training, 7000 for depth completion training, and 5000 for perceptual quality and downstream evaluation. Please see supp. for the full breakdown.

Argoverse: We also evaluate on the Argoverse training split which contains 65 snippets from 2 different cities. We use the provided HD maps for vehicle placement. We directly adopt the 3D assets built from UrbanData, as Argoverse is too small for diverse asset creation. We train our image synthesis network on Argoverse, where 80k frames are sampled for training and 16k are sampled for evaluation.

Asset Bank Creation: We created automatically a large object bank of ~ 8000 vehicles, from cameras, LiDAR data and 3D bounding boxes using our 3D reconstruction network on UrbanData. Each successfully reconstructed object is registered in our 3D asset bank, with its 1) 3D mesh; 2) images; and 3) object pose in ego-vehicle-centric coordinates. We use a pre-trained instance segmentation to get the inferred instance mask [40], a LiDAR detector [48] to acquire other actors' bounding boxes for collision avoidance (Sec. 4.1), and a depth completion network [14] to get dense depth for occlusion reasoning (Sec. 4.2).

Baselines: We compare our method against several deep learning based end-to-end 2D image synthesis and augmentation baselines [44, 71, 29, 56]. Unlike GeoSim, these methods cannot perform placement directly and require an input mask based on the object's shape and pose that denotes the area to synthesize. We therefore use [44] to insert object instances at the semantic level in a background semantic image. We then generate high resolution images from this augmented scene representation with three different approaches: (1) Holistic image generation (**"SPADE"**): we use the state-of-the-art conditional image generation model SPADE [56] to generate the entire image given the semantic mask. (2) Retrieval-based generation (**"Cut-Paste"**): given the new object's 2D mask, we retrieve the most similar example from a bank of 2D object images. The similarity is defined using

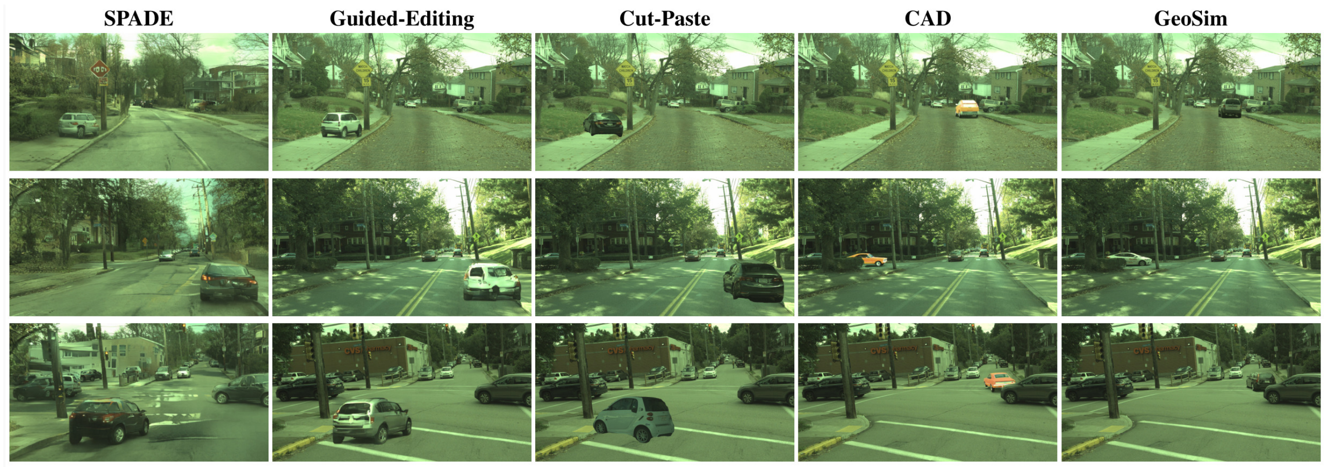

Figure 5: **Qualitative comparison of image simulation approaches.**

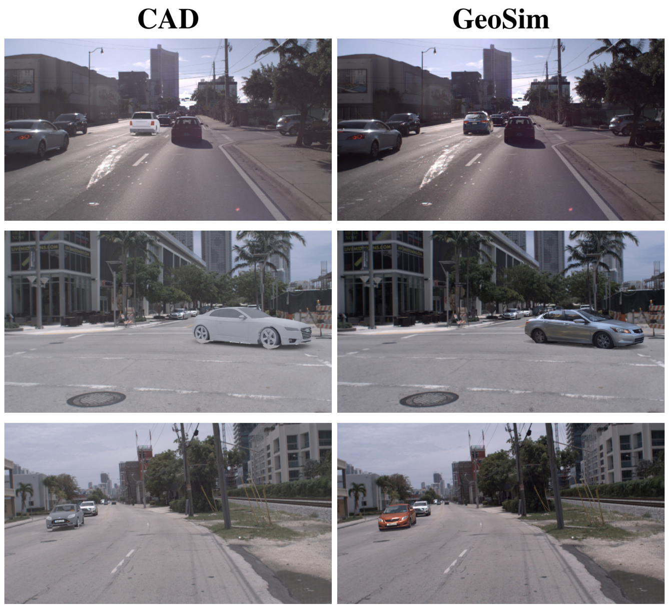

Figure 6: **Qualitative comparison of image simulation approaches on Argoverse dataset.**

semantic mask IOU. The rest of the background comes from the corresponding real image; and (3) Guided semantic image editing (**"Guided-Editing"**): we use [29] to in-paint the tight bounding box region of the added object. Additionally, we compare against a graphics-based CAD model insertion baseline (**"CAD"**), in spirit of [3] with the following differences: 1) we use our 3D placement in order to produce more realistic layout-aware insertion; 2) unlike the original work, we do not have environment lighting maps and instead use a HDRI captured on a cloudy day.

5.2. Perceptual Quality Evaluation

Human Study: To verify the realism of our approach, we conduct a human A/B test, where we show a pair of images generated from different approaches on the same background image, one from GeoSim and another one from a competing algorithm. We then ask the human judges to click the one they believe is more realistic. In total, 13 human judges participated and labeled ~ 1500 image pairs. Tab. 1 shows the human preference score for each algorithm, which measures the percentage of participants who prefer our GeoSim results over each baseline method. Results on Argoverse are presented in Tab. 3. The A/B test confirms that our method produces drastically more realistic images than the baselines. The minimum p-value in the A/B test is 1.64e-18, demonstrating statistical significance. Please see the detailed A/B test interface and instructions in supp.

Perceptual Quality Score: We further use the Fréchet Inception Distance (FID) [28] between the synthesized images and the ground-truth images as an automatic measure of image quality. We report the FID on the full image for GeoSim and the baselines in Tab. 1. Our method significantly outperforms all competing methods on FID.

Qualitative Comparison: Fig. 5 compares simulated images. Note that GeoSim is significantly more realistic than the baselines. While one can easily and quickly detect the added object in other methods due to unrealistic generation with smeared cars ("SPADE", "Guided-Editing"), or geometrically invalid results ("Cut-Paste"), or unrealistic appearance ("CAD"), one must look closely at GeoSim images to distinguish the added objects from the real ones. In Fig. 6, we show qualitative examples on Argoverse, where GeoSim obtains very high visual quality. This demonstrates GeoSim's potential to generalize across datasets.

Effect of Rendering Approach: We evaluate the importance of using a hybrid rendering module proposed in our method, compared to using solely physics-based rendering or 2D synthesis with 3D placement constant across all approaches). As shown in Tab. 2, our proposed geometry-aware

Method	Human Score (%)	FID
SPADE [56]	99.3	43.2
Guided Editing [29]	94.3	20.3
Cut-Paste [20]	98.5	22.1
CAD [2]	94.3	17.3
GeoSim	**-**	**14.3**

Table 1: **Perceptual quality evaluation.** Human score: % of participants who prefer our GeoSim results over baseline.

Approach	Shadow	Human Score (%)	FID
Physics	Yes	94.2	17.3
2D Synthesis	-	75.7	**13.7**
Geo Synthesis	No	71.9	**13.7**
Geo Synthesis	Yes	**-**	14.3

Table 2: **Ablation on rendering options for GeoSim.** Human score: % of participants who prefer our GeoSim results over baseline.

Method	Human Score (%)	FID
CAD	84.0	28.3
GeoSim	-	**24.5**

Table 3: **Results on Argoverse**. Human score: % of participants who prefer our GeoSim results over baseline.

synthesis significantly outperforms all other approaches on human scores. Additionally, enhancing hybrid-rendering with shadows significantly boosts the realism for humans, but such improvements are not reflected in FID score. This suggests there still exists a gap between computational perceptual quality measurements and humans' criteria. Please see supp. for ablation of other GeoSim components.

Video Simulation: We showcase in the supp. video GeoSim's ability to simulate highly realistic and temporally consistent video for multiple cameras.

Failure Cases: While most GeoSim-simulated images are high-quality, there is room for improvement. We find four major failure cases: (1) incorrect occlusion relationships in a complex scene, (2) irregular reconstructed mesh, (3) inaccurate object pose, usually caused by map error and (4) illumination failure due to illumination differences between rendered segment and target scene. Besides, we also notice blank pixel artifacts in long range video simulation, which are caused by inverse warping textures from source viewpoints which are far from the target viewpoint. Please refer to the supp. for qualitative examples.

5.3. Downstream Perception Task

We now investigate data augmentation, where we use labeled real data combined with GeoSim to get performance gains, without the cost of large scale annotations (as seen in Fig. 7). We first train a segmentation model on labeled real data with around 2000 images. We then use GeoSim to augment these images with inserted vehicles, obtaining 9879 additional training examples in total. We re-train the segmentation model on both real and augmented data for the same number of iterations. We evaluate the performance on real data and report the results on Tab. 4. With these additional training images, we can further boost perception performance by 3.4% (or 0.4% on DeepLabv3 [10]) for car category and 1.8% (or 0.2% on DeepLabv3 [10]) for overall mIOU on PSPNet [80]. Importantly we can show consistent improvements across two segmentation models.

Method	PSPNet [80]		DeepLabv3 [10]	
	mIOU	carIOU	mIOU	carIOU
Real	93.5	87.8	94.0	88.7
Real+GeoSim	**95.3**	**91.2**	**94.2**	**89.2**

Table 4: **Sim2Real on semantic segmentation.**

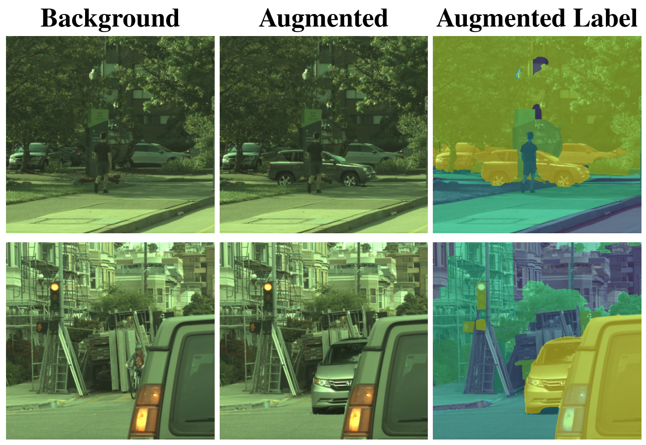

Figure 7: **GeoSim for data augmentation.** Left: image before augmentation. Middle: image after augmentation. Right: augmented semantic annotation.

6. Conclusion

In this work we presented a novel geometry-guided simulation procedure for flexible generation and rendering of synthetic scenes. Not only is our approach the first of its kind to fully take into consideration physical realism for dynamic object placement into images, it also bypasses the need for manual 3D asset creation and achieves greater visual realism than competing alternatives. Moreover, we demonstrated improvements in downstream tasks through applications of our technique to semantic segmentation. There are many exciting follow-up directions opened up by this work such as sim2real, autonomous system evaluation, video editing, etc. and we look forward to future extensions of GeoSim.

References

[1] Corona renderer, 2020. 1

[2] Hassan Abu Alhaija, Siva Karthik Mustikovela, Lars Mescheder, Andreas Geiger, and Carsten Rother. Augmented Reality Meets Computer Vision : Efficient Data Generation for Urban Driving Scenes. *arXiv*, 2017. 2, 8

[3] Hassan Abu Alhaija, Siva Karthik Mustikovela, Lars Mescheder, Andreas Geiger, and Carsten Rother. Augmented reality meets computer vision: Efficient data generation for urban driving scenes. *IJCV*, 2018. 7

[4] Alexander Amini, Igor Gilitschenski, Jacob Phillips, Julia Moseyko, Rohan Banerjee, Sertac Karaman, and Daniela Rus. Learning Robust Control Policies for End-to-End Autonomous Driving From Data-Driven Simulation. *RA-L*, 2020. 2

[5] Anand Bhattad and David A Forsyth. Cut-and-paste neural rendering. *arXiv*, 2020. 1, 3

[6] Konstantinos Bousmalis, Alex Irpan, Paul Wohlhart, Yunfei Bai, Matthew Kelcey, Mrinal Kalakrishnan, Laura Downs, Julian Ibarz, Peter Pastor, Kurt Konolige, et al. Using simulation and domain adaptation to improve efficiency of deep robotic grasping. In *ICRA*, 2018. 2

[7] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. *arXiv*, 2019. 2

[8] Caroline Chan, Shiry Ginosar, Tinghui Zhou, and Alexei A Efros. Everybody dance now. In *CVPR*, 2019. 3

[9] Ming-Fang Chang, John W Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, and James Hays. Argoverse: 3d tracking and forecasting with rich maps. In *CVPR*, 2019. 2, 3, 6

[10] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv*, 2017. 8

[11] Qifeng Chen and Vladlen Koltun. Photographic Image Synthesis with Cascaded Refinement Networks. *arXiv*, 2017. 1, 2

[12] Wenzheng Chen, Jun Gao, Huan Ling, Edward Smith, Jaakko Lehtinen, Alec Jacobson, and Sanja Fidler. Learning to predict 3d objects with an interpolation-based differentiable renderer. In *NIPS*, 2019. 3, 4, 5

[13] Xu Chen, Jie Song, and Otmar Hilliges. Monocular Neural Image Based Rendering with Continuous View Control. *arXiv*, 2019. 3

[14] Yun Chen, Bin Yang, Ming Liang, and Raquel Urtasun. Learning joint 2d-3d representations for depth completion. In *ICCV*, 2019. 6

[15] Yung-Yu Chuang, Dan B Goldman, Brian Curless, David H. Salesin, and Richard Szeliski. Shadow matting and compositing. *ACM Trans. Graph.*, 2003. 5

[16] Wenyan Cong, Jianfu Zhang, Li Niu, Liu Liu, Zhixin Ling, Weiyuan Li, and Liqing Zhang. Dovenet: Deep image harmonization via domain verification. In *CVPR*, 2020. 1

[17] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. *GitHub repository*, 2016. 2

[18] Paul Debevec. Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, 1998. 5

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

[20] Debidatta Dwibedi, Ishan Misra, and Martial Hebert. Cut, paste and learn: Surprisingly easy synthesis for instance detection. In *ICCV*, 2017. 1, 8

[21] Sébastien Ehrhardt, Oliver Groth, Aron Monszpart, Martin Engelcke, Ingmar Posner, Niloy Mitra, and Andrea Vedaldi. Relate: Physically plausible multi-object scene synthesis using structured latent spaces. *NeurIPS*, 2020. 3

[22] Epic Games. Unreal engine. 1

[23] Jin Fang, Dingfu Zhou, Feilong Yan, Tongtong , Feihu Zhang, Yu Ma, Liang Wang, and Ruigang Yang. Augmented LiDAR Simulator for Autonomous Driving. *arXiv*, 2019. 2

[24] Oran Gafni, Lior Wolf, and Yaniv Taigman. Vid2game: Controllable characters extracted from real-world videos. *arXiv*, 2019. 3

[25] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *IJRR*, 2013. 2, 3

[26] J Gonzales, F Zhang, K Li, and F Borrelli. Autonomous drifting with onboard sensors. In *Advanced Vehicle Control: Proceedings of the 13th International Symposium on Advanced Vehicle Control (AVEC16)*, 2016. 5

[27] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep blending for free-viewpoint image-based rendering. *TOG*, 2018. 3

[28] 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. *arXiv*, 2017. 7

[29] Seunghoon Hong, Xinchen Yan, Thomas Huang, and Honglak Lee. Learning Hierarchical Semantic Image Manipulation through Structured Representations. *arXiv*, 2018. 2, 6, 7, 8

[30] Seunghoon Hong, Dingdong Yang, Jongwook Choi, and Honglak Lee. Inferring semantic layout for hierarchical text-to-image synthesis. In *CVPR*, 2018. 2

[31] Abdul-Wahab Sami Ibrahim and Sarab Mhamed Taher. Inserting virtual static object with geometry consistency into real video. In *Journal of Physics: Conference Series*, 2020. 3

[32] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017. 2

[33] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial Transformer Networks. *arXiv*, 2016. 5

[34] Justin Johnson, Alexandre Alahi, and Fei-Fei Li. Perceptual losses for real-time style transfer and super-resolution. *CoRR*, 2016. 6

[35] Justin Johnson, Agrim Gupta, and Li Fei-Fei. Image generation from scene graphs. In *CVPR*, 2018. 1, 2

[36] Angjoo Kanazawa, Shubham Tulsiani, Alexei A. Efros, and Jitendra Malik. Learning Category-Specific Mesh Reconstruction from Image Collections. *arXiv*, 2018. 3, 4

[37] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2019. 2

[38] Kevin Karsch, Varsha Hedau, David Forsyth, and Derek Hoiem. Rendering synthetic objects into legacy photographs. *TOG*, 2011. 1

[39] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3D Mesh Renderer. *arXiv*, 2017. 3

[40] Alexander Kirillov, Yuxin Wu, Kaiming He, and Ross Girshick. Pointrend: Image segmentation as rendering. In *CVPR*, 2020. 4, 6

[41] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. Ai2-thor: An interactive 3d environment for visual ai. *arXiv*, 2017. 2

[42] Abhijit Kundu, Yin Li, and James M. Rehg. 3d-rcnn: Instance-level 3d object reconstruction via render-and-compare. In *CVPR*, 2018. 4

[43] Jean-François Lalonde, Derek Hoiem, Alexei A Efros, Carsten Rother, John Winn, and Antonio Criminisi. Photo clip art. *TOG*, 2007. 1

[44] Donghoon Lee, Sifei Liu, Jinwei Gu, Ming-Yu Liu, Ming-Hsuan Yang, and Jan Kautz. Context-aware synthesis and placement of object instances. In *NIPS*, 2018. 2, 6

[45] D. Lee, T. Pfister, and M. Yang. Inserting videos into videos. In *CVPR*, 2019. 3

[46] Wei Li, Chengwei Pan, Rong Zhang, Jiaping Ren, Yuexin Ma, Jin Fang, Feilong Yan, Qichuan Geng, Xinyu Huang, Huajun Gong, Weiwei Xu, Guoping Wang, Dinesh Manocha, and Ruigang Yang. AADS: Augmented Autonomous Driving Simulation using Data-driven Algorithms. *Science Robotics*, 2019. 2

[47] Xueting Li, Sifei Liu, Kihwan Kim, Xiaolong Wang, Ming-Hsuan Yang, and Jan Kautz. Putting humans in a scene: Learning affordance in 3d indoor environments. In *CVPR*, 2019. 2

[48] Ming Liang, Bin Yang, Wenyuan Zeng, Yun Chen, Rui Hu, Sergio Casas, and Raquel Urtasun. Pnpnet: End-to-end perception and prediction with tracking in the loop. In *CVPR*, 2020. 4, 6

[49] Chen-Hsuan Lin, Ersin Yumer, Oliver Wang, Eli Shechtman, and Simon Lucey. ST-GAN: Spatial Transformer Generative Adversarial Networks for Image Compositing. *arXiv*, 2018. 2

[50] Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. Soft Rasterizer: A Differentiable Renderer for Image-based 3d Reasoning. *arXiv*, 2019. 3

[51] Arun Mallya, Ting-Chun Wang, Karan Sapra, and Ming-Yu Liu. World-consistent video-to-video synthesis. In *ECCV*, 2020. 3

[52] Sivabalan Manivasagam, Shenlong Wang, Kelvin Wong, Wenyuan Zeng, Mikita Sazanovich, Shuhan Tan, Bin Yang, Wei-Chiu Ma, and Raquel Urtasun. Lidarsim: Realistic lidar simulation by leveraging the real world. *CVPR*, 2020. 2

[53] Mark Martinez, Chawin Sitawarin, Kevin Finch, Lennart Meincke, Alex Yablonski, and Alain Kornhauser. Beyond grand theft auto v for training, testing and enhancing deep learning in self driving cars. *arXiv*, 2017. 1

[54] Sangwoo Mo, Minsu Cho, and Jinwoo Shin. Instagan: Instance-aware image-to-image translation. *arXiv*, 2018. 2

[55] Eunbyung Park, Jimei Yang, Ersin Yumer, Duygu Ceylan, and Alexander C. Berg. Transformation-Grounded Image Generation Network for Novel 3d View Synthesis. *arXiv*, 2017. 3

[56] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *CVPR*, 2019. 1, 2, 6, 8

[57] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *arXiv*, 2016. 4

[58] Xiaojuan Qi, Qifeng Chen, Jiaya Jia, and Vladlen Koltun. Semi-parametric Image Synthesis. *arXiv*, 2018. 2

[59] Stephan R. Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for Data: Ground Truth from Computer Games. *arXiv*, 2016. 2

[60] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *CVPR*, 2016. 2

[61] Manolis Savva, Angel X Chang, Alexey Dosovitskiy, Thomas Funkhouser, and Vladlen Koltun. Minos: Multimodal indoor simulator for navigation in complex environments. *arXiv*, 2017. 2

[62] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, et al. Habitat: A platform for embodied ai research. In *ICCV*, 2019. 2

[63] Jens Schulz, Constantin Hubmann, Julian Löchner, and Darius Burschka. Interaction-Aware Probabilistic Behavior Prediction in Urban Environments. *arXiv*, 2018. 3, 5

[64] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *CVPR*, 2020. 2, 3

[65] Shao-Hua Sun, Minyoung Huh, Yuan-Hong Liao, Ning Zhang, and Joseph J Lim. Multi-view to novel view: Synthesizing novel views with self-learned confidence. In *ECCV*, 2018. 3

[66] Simon Suo, Sebastian Regalado, Sergio Casas, and Raquel Urtasun. Trafficsim: Learning to simulate realistic multi-agent behaviors. *arXiv*, 2021. 3, 5

[67] Abhijeet Tallavajhula, Cetin Mericli, and Alonzo Kelly. Off-road lidar simulation using data driven terrain primitives. In *ICRA*, 2018. 2

[68] Ayush Tewari, Ohad Fried, Justus Thies, Vincent Sitzmann, Stephen Lombardi, Kalyan Sunkavalli, Ricardo Martin-Brualla, Tomas Simon, Jason Saragih, Matthias Nießner, Rohit Pandey, Sean Fanello, Gordon Wetzstein, Jun-Yan Zhu, Christian Theobalt, Maneesh Agrawala, Eli Shechtman, Dan B. Goldman, and Michael Zollhöfer. State of the Art on Neural Rendering. *arXiv*, 2020. 1, 3

[69] Shubham Tulsiani, Richard Tucker, and Noah Snavely. Layer-structured 3d scene inference via view synthesis. In *ECCV*, 2018. 5

[70] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Guilin Liu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. Video-to-video synthesis. In *NeurIPS*, 2018. 3

[71] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs. *arXiv*, 2017. 1, 2, 6

[72] Xiaolong Wang and Abhinav Gupta. Generative image modeling using style and structure adversarial networks. In *ECCV*, 2016. 1, 2

[73] Yifan Wang, Brian Curless, and Steve Seitz. People as Scene Probes. *arXiv*, 2020. 5

[74] Fei Xia, Amir R. Zamir, Zhiyang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson Env: Real-World Perception for Embodied Agents. In *CVPR*, 2018. 2

[75] Chao Yang, Xin Lu, Zhe Lin, Eli Shechtman, Oliver Wang, and Hao Li. High-Resolution Image Inpainting using Multi-Scale Neural Patch Synthesis. *arXiv*, 2016. 1

[76] Zhenpei Yang, Yuning Chai, Dragomir Anguelov, Yin Zhou, Pei Sun, Dumitru Erhan, Sean Rafferty, and Henrik Kretzschmar. SurfelGAN: Synthesizing Realistic Sensor Data for Autonomous Driving. *arXiv*, 2020. 2

[77] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas Huang. Free-Form Image Inpainting with Gated Convolution. *arXiv*, 2019. 6

[78] Shuyang Zhang, Runze Liang, and Miao Wang. Shadowgan: Shadow synthesis for virtual objects with conditional adversarial networks. *Computational Visual Media*, 2019. 5

[79] Yuxuan Zhang, Wenzheng Chen, Huan Ling, Jun Gao, Yinan Zhang, Antonio Torralba, and Sanja Fidler. Image gans meet differentiable rendering for inverse graphics and interpretable 3d neural rendering. *arXiv*, 2020. 3

[80] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *CVPR*, 2017. 8

[81] Tinghui Zhou, Shubham Tulsiani, Weilun Sun, Jitendra Malik, and Alexei A. Efros. View Synthesis by Appearance Flow. *arXiv*, 2016. 3

[82] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, 2017. 2

[83] Jun-Yan Zhu, Zhoutong Zhang, Chengkai Zhang, Jiajun Wu, Antonio Torralba, Joshua B. Tenenbaum, and William T. Freeman. Visual Object Networks: Image Generation with Disentangled 3D Representation. *arXiv*, 2018. 3