Neural Scene Graphs for Dynamic Scenes

Julian Ost[1] Fahim Mannan[1] Nils Thuerey[2] Julian Knodt[3] Felix Heide[1,3]
[1]Algolux [2]Technical University of Munich [3]Princeton University

http://light.princeton.edu/neural-scene-graphs

Abstract

Recent implicit neural rendering methods have demonstrated that it is possible to learn accurate view synthesis for complex scenes by predicting their volumetric density and color supervised solely by a set of RGB images. However, existing methods are restricted to learning efficient representations of static scenes that encode all scene objects into a single neural network, and they lack the ability to represent dynamic scenes and decompose scenes into individual objects. In this work, we present the first neural rendering method that represents multi-object dynamic scenes as scene graphs. We propose a learned scene graph representation, which encodes object transformations and radiance, allowing us to efficiently render novel arrangements and views of the scene. To this end, we learn implicitly encoded scenes, combined with a jointly learned latent representation to describe similar objects with a single implicit function. We assess the proposed method on synthetic and real automotive data, validating that our approach learns dynamic scenes – only by observing a video of this scene – and allows for rendering novel photo-realistic views of novel scene compositions with unseen sets of objects at unseen poses.

1. Introduction

View synthesis and scene reconstruction from a set of captured images are fundamental problems in computer graphics and computer vision. Classical methods rely on sequential reconstructions and rendering pipelines that first recover a compact scene representation, such as a point-cloud or textured meshes using multi-view stereo [1, 12, 28, 29], which is then used to render novel views using efficient direct or global illumination rendering methods. These sequential pipelines also can recover hierarchical scene representations [13], representing dynamic scenes [8, 25], and efficiently rendering novel views [4]. However, traditional pipelines struggle to capture highly view-dependent features at discontinuities, or illumination-dependent reflectance of scene objects.

Recently, these challenges have been tackled by neural rendering methods. The most successful methods [22, 16] depart from explicit scene representations such as meshes and estimated BRDF models, and instead learn fully implicit representations that embed three dimensional scenes in functions, supervised by a sparse set of images during the training. Specifically, implicit scene representation like Neural Radiance Fields (NeRF) by Mildenhall et al. [22] encode scene representations within the weights of a neural network that map 3D locations and viewing directions to a neural radiance field. The novel renderings from this representation improve on previous methods of discretized voxel grids [31].

Besides their advantages, recent learned methods encode the entire scene into a single, static network that does not allow for hierarchical representations or dynamic scenes that are supported by traditional pipelines. Thus, existing neural rendering approaches assume that the training images stem from an underlying scene that does not change between view samples. More recent approaches, such as NeRF-W [19], attempt to improve on this shortcoming by learning to ignore dynamic objects that cause occlusions in the static scene. However, this approach still relies on a underlying static scene to learn the representation.

In this work, we present the first method that is able to learn a representation for complex, dynamic, multi-object scenes. Our method decomposes a scene into its static and dynamic parts and learns their representations structured as a scene graph, with transformations defined by tracking information and supervised by the RGB frames of a video. The proposed approach allows us to synthesize novel views of a scene, or render views for completely unseen arrangements of dynamic objects. Furthermore, we show that the method can also be used for 3D object detection via inverse rendering.

Using automotive data sets, our experiments confirm that our method is capable of representing scenes with highly dynamic objects. We assess the method by generating unseen views of novel scene compositions with unseen sets of objects at unseen poses.

Specifically, we make the following contributions:

- We propose a novel neural rendering method that decomposes dynamic, multi-object scenes into a learned scene graph with decoupled object transformations and scene representations.
- We learn object representations for each scene graph node directly from a set of video frames and corresponding tracking data. We encode instances of a class of objects using a shared sparse volumetric representation.
- We validate the method on simulated and experimental data by rendering novel unseen views and unseen dynamic scene arrangements of the represented scene.
- We demonstrate that the proposed method facilitates 3D object detection via inverse rendering.

2. Related Work

Recently, the combination of deep learning approaches and traditional rendering methods from computer graphics have enabled researchers to synthesize photo-realistic views for static scenes, using RGB images and corresponding poses of sparse views as input. We review related work in the areas of neural rendering, implicit scene representations, scene graphs, and latent object descriptors.

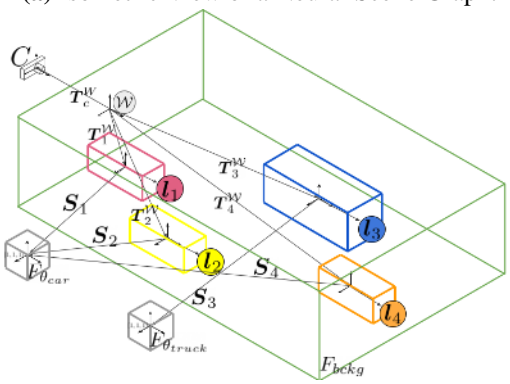
Implicit Scene Representations and Neural Rendering A rapidly growing body of work has explored neural rendering methods [38] for static scenes. Existing methods typically employ a learned scene representation and a differentiable renderer. Departing from traditional representations in computer graphics, which explicitly model every surface in a scene graph hierarchy, neural scene representations implicitly represent scene features as outputs of a neural network. Such scene representations can be learned by supervision using images, videos, point clouds, or a combination of those. Existing methods have proposed to learn features on discrete geometric primitives, such as points [2, 27], multi-planes [10, 17, 21, 35, 46], meshes [6, 39] higher-order primitives like voxel grids [16, 31, 44], or implicitly represent features [15, 20, 24, 26, 43, 22] using functions $F : \mathbb{R}^3 \rightarrow \mathbb{R}^n$ that map a point in the continuous 3D space to a n-dimensional feature space. While traditional and explicit scene representations are interpretable and allow for decomposition into individual objects, they do not scale with scene complexity. Implicit representations model scenes as functions, typically encoding the scene using a multilayer perceptron (MLP). This gives up interpretability, but allows for a continuous view interpolation [22]. The proposed method closes this gap by introducing a hierarchical scene-graph model with object-level implicit representations.

Differentiable rendering functions have made it possible to learn scene representations [16, 24, 22, 32] from a set of images. Given a camera's extrinsic and intrinsic parameters, existing methods rely on differentiable ray casting or ray marching through the scene volume to infer the features for sampled points along a ray. Varying the extrinsic parameters of the camera at test time makes novel view synthesis for a given static scene possible. Niemeyer et al. [24] used such an approach for learning an implicit representation for object shapes and textures. Departing from surface representations, Mescheder et al. [20] and Mildenhall et al. [22] output a color value conditioned on a ray's direction, allowing them to achieve high-quality, view-dependent effects. Our work builds on this approach to model dynamic scene objects, and we review NeRF's concept in the Supplemental Material. All methods described above rely on static scenes with consistent mappings from a 3D point to a 2D observation. The proposed method lifts this assumption and tackles dynamic scenes by introducing a learned scene graph representation.

Scene Graph Representations Scene graphs are traditional hierarchical representations of scenes. Introduced almost 30 years ago [42, 34, 33, 9], they model a scene as a directed graph which represents objects as leaf nodes. These leaf nodes are grouped hierarchically in the directed graph, with transformations, such as translation, rotation and scaling, applied to the sub-graph along each edge, with each transformation defined in the local frame of its parent node. The global transformation for each leaf node can be efficiently calculated by stacking transformations from the global scene frame to the leaf node, and allows for reuse of objects and groups that are common to multiple child nodes. Since the 1990s, scene graphs have been adopted in rendering and modeling pipelines, including Open Inventor [42], Java 3D [34], and VRML [3]. Different geometry leaf-node representations have been proposed in the past, including individual geometry primitives, lists or volumetric shapes [23]. Aside from the geometric description, a leaf node can also represent individual properties of their parent nodes such as materials and lighting. In this work, we revisit scene graphs as a powerful hierarchical representation which we combine with learned implicit leaf nodes.

Latent Class Encoding In order to better learn distributions of shapes, a number of existing works [20, 24, 26] propose to learn shape descriptors that generalize implicit scene representations across similar objects. By adding a latent vector $\boldsymbol{z}$ to the input 3D query point, similar objects can be modeled using the same network. Adding generalization across a class of objects can also be seen in other recent work that either use an explicit encoder network to predict a latent code from an image [20, 24], or use a latent code to predict network weights directly [32]. Instead of

(a) Isometric View of a Neural Scene Graph.



(b) Ego-vehicle View of a Neural Scene Graph.

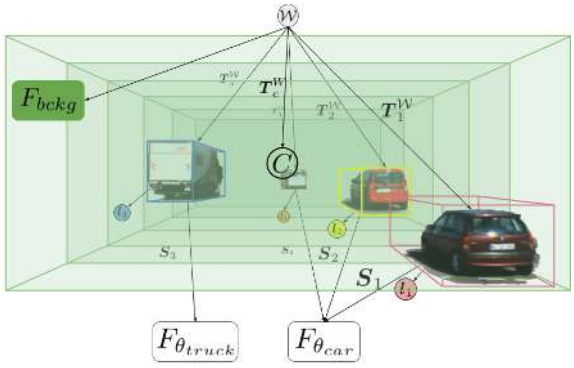


Figure 1: (a) Three dimensional and (b) projected view of the same scene graph $\mathcal{S}_i$. The nodes are visualized as boxes with their local cartesian coordinate axis. Edges with transformations and scaling between the parent and child coordinate frames are visualized by arrows with their transformation T_o^w and scaling matrix S_o respectively. The corresponding latent codes are denoted by $\boldsymbol{l}_o$ and the representation nodes for the unit-scaled bounding boxes by F_θ. No transformation is assigned to the background, which is already aligned with $\mathcal{W}$.

using an encoder network, Park et al. [26] propose to optimize latent descriptors for each object directly following Tan et al. [36]. From a probabilistic view, the resulting latent code for each object follows the posterior $p_\theta(\boldsymbol{z}_i|X_i)$, given the object shape X_i. In this work, we rely on this probabilistic formulation to reason over dynamic objects of the same class, allowing us to untangle global illumination effects conditioned on object position and type.

Before presenting the proposed method for dynamic scenes, we refer the reader to the Supplemental Material for a review of NeRF by Mildenhall et al. [22] as a successful implicit representation for static scenes.

3. Neural Scene Graphs

In this section we introduce the neural scene graph, which allows us to model scenes hierarchically. The scene graph $\mathcal{S}$, illustrated in Fig. 1, is composed of a camera, a static node and a set of dynamic nodes which represent the dynamic components of the scene, including the object appearance, shape, and class.

Graph Definition We define a scene uniquely using a directed acyclic graph, $\mathcal{S}$, given by

$$\mathcal{S} = \langle \mathcal{W}, C, F, L, E \rangle, \tag{1}$$

where C is a leaf node representing the camera and its intrinsics $K \in \mathbb{R}^{3\times3}$, $F = F_{\theta_{bckg}} \cup \{F_{\theta_c}\}_{c=1}^{N_{class}}$ are leaf nodes representing both static and dynamic representation models. The nodes $L = \{\boldsymbol{l}_o\}_{o=1}^{N_{obj}}$ are leaf nodes that assign latent object codes to each shape representation leaf node. The E are edges that either represent affine transformations from u to v or property assignments, that is

$$E = \{e_{u,v} \in \{[\,], \boldsymbol{M}_v^u\}\}, \tag{2}$$

$$\text{with } \boldsymbol{M}_v^u = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0} & 1 \end{bmatrix}, \boldsymbol{R} \in \mathbb{R}^{3\times3}, \boldsymbol{t} \in \mathbb{R}^3. \tag{3}$$

For a given scene graph, poses and locations for all objects can be extracted. For all edges originating from the root node $\mathcal{W}$, we assign transformations between the global world space and the local object or camera space with $\boldsymbol{T}_v^{\mathcal{W}}$. Representation models are shared and defined in a unit-scaled frame. To represent different scales of the same object type, we compute the non-uniform scaling $\boldsymbol{S}_o$, which is assigned at edge $e_{o,f}$ between an object node and its shared representation model F.

To retrieve an object's local transformation and position $\boldsymbol{p}_o = [x, z, y]_o$, we traverse from the root node $\mathcal{W}$, applying transformations until the desired object node F_{θ_o} is reached, eventually ending in the frame of reference, see Eq. 9.

Representation Models For the model representation nodes, F, that either model the background or dynamic objects, we follow Mildenhall et al. [22] and represent scene objects as augmented implicit neural radiance fields. In the following, we describe two augmented models for neural radiance fields which are illustrated in Fig. 2 representing scenes as shown in Fig. 1.

3.1. Background Node

Our scene graphs include a single background representation node, shown in the top of Fig. 2, that approximates all static scene parts. Departing from prior work, we represent static radiance on sparse planes instead of a volume. The static background node function $F_{\theta_{bckg}} : (\boldsymbol{x}, \boldsymbol{d}) \rightarrow (\boldsymbol{c}, \boldsymbol{\sigma})$ maps a point $\boldsymbol{x}$ to its density on sparse depth planes and, combined with a viewing direction, to an emitted color on the planes. Thus, the background representation is implicitly stored in the weights θ_{bckg}. We use a set of mapping functions, Fourier encodings [37], to aid learning high-frequency functions in the MLP model. We map the positional and directional inputs with $\gamma(\boldsymbol{x})$ and $\gamma(\boldsymbol{d})$ to higher-frequency feature vectors and pass those as inputs to the background model, resulting in the following two stages of the representation network:

$$[\sigma(\boldsymbol{x}), \boldsymbol{y}(\boldsymbol{x})] = F_{\theta_{bckg,1}}(\gamma_x(\boldsymbol{x})) \tag{4}$$

$$\boldsymbol{c}(\boldsymbol{x}) = F_{\theta_{bckg,2}}(\gamma_d(\boldsymbol{d}), \boldsymbol{y}(\boldsymbol{x})). \tag{5}$$

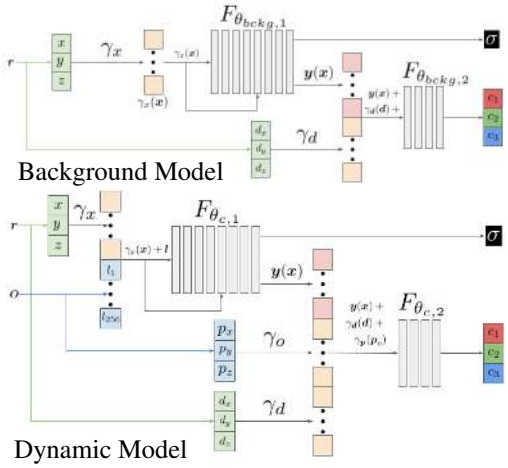


Figure 2: Architectures of representation networks for static and dynamic models. Inputs to the network are point $\boldsymbol{x}$ and ray direction $\boldsymbol{d}$, and for objects o we include a latent code $\boldsymbol{l}_o$ and the pose $\boldsymbol{p}_o$, outputting σ from the first stage, and $\boldsymbol{c}$ from the second stage.

3.2. Dynamic Nodes

For dynamic scene objects, we consider each individual rigid part of a scene that changes its pose through the duration of the capture. A single, connected object is denoted as a dynamic object o. Each object is represented by a neural radiance field in the local space of its node and position $\boldsymbol{p}_o$. Objects with a similar appearance are combined in a class c and share weights θ_c of the representation function F_{θ_c}. A learned, latent encoding vector $\boldsymbol{l}_o$ distinguishes the neural radiance fields of individual objects, representing a class of objects with

$$[\boldsymbol{c}(\boldsymbol{x}_o), \sigma(\boldsymbol{x}_o)] = F_{\theta_c}(\boldsymbol{l}_o, \boldsymbol{p}_o, \boldsymbol{x}_o, \boldsymbol{d}_o). \tag{6}$$

Latent Class Encoding Training an individual representation for each object in a dynamic scene can easily lead to a large number of models and training effort. Instead, we aim to minimize the number of models that represent all objects in a scene. We reason about shared object features and untangle global illumination effects from individual object radiance fields. Similar to Park et al. [26], we introduce a latent vector $\boldsymbol{l}$ encoding an object's representation. Adding $\boldsymbol{l}_o$ to the input of a volumetric scene function F_{θ_c} can be thought as a mapping from the representation function of class c to the radiance field of object o as in

$$F_{\theta_c}(\boldsymbol{l}_o, \boldsymbol{x}, \boldsymbol{d}) = F_{\theta_o}(\boldsymbol{x}, \boldsymbol{d}). \tag{7}$$

Conditioning on the latent code makes it possible to use shared weights θ_c between all objects of class c. Global illumination effects only visible for some objects during training are shared across all objects of the same class. We modify the input to the mapping function, conditioning the volume density on the global 3D location $\boldsymbol{x}$ of the sampled point and a latent vector $\boldsymbol{l}_o$, resulting in the following new first stage

$$[\boldsymbol{y}(\boldsymbol{x}), \sigma(\boldsymbol{x})] = F_{\theta_{c,1}}(\gamma_{\boldsymbol{x}}(\boldsymbol{x}), \boldsymbol{l}_o). \tag{8}$$

Object Coordinate Frame The global location $\boldsymbol{p}_o$ of a dynamic object changes between frames, thus its radiance field moves as well. We introduce local three-dimensional cartesian coordinate frames $\mathcal{F}_o$, fixed and aligned with an object's pose. The transformation of a point in the the global frame $\mathcal{F}_{\mathcal{W}}$ to $\mathcal{F}_o$ is given by

$$\boldsymbol{x}_o = \boldsymbol{S}_o \boldsymbol{T}_o^w \boldsymbol{x} \text{ with } \boldsymbol{x}_o \in [1, -1], \tag{9}$$

scaling with the inverse length of the bounding box size $\boldsymbol{s}_o = [L_o, H_o, W_o]$ with $\boldsymbol{S}_o = diag(1/\boldsymbol{s}_o)$. Scaling allows us to learn size-independent similarities in θ_c.

Object Representation The continuous volumetric scene functions F_{θ_c} from Eq. 7 are modeled with a MLP architecture presented in the bottom of Fig. 2. This model maps a latent vector $\boldsymbol{l}_o$, point $\boldsymbol{x} \in [-1, 1]$ and a viewing direction $\boldsymbol{d}$ in the local frame of o to its corresponding volumetric density $\boldsymbol{\sigma}$ and directional emitted color $\boldsymbol{c}$. The appearance of a dynamic object depends on its interaction with the scene and global illumination which is dependent on object location $\boldsymbol{p}_o$. To consider location dependent effects, we add $\boldsymbol{p}_o$ in the global frame as another input. We note that an object's volumetric density σ should not change based on on its pose in the scene, so to ensure volumetric consistency the pose is only considered for the emitted color and not the density. We add the pose to the inputs $\boldsymbol{y}(t)$ and $\boldsymbol{d}$ of the second stage:

$$\boldsymbol{c}(\boldsymbol{x}, \boldsymbol{l}_o, \boldsymbol{p}_o) = F_{\theta_{c,2}}(\gamma_d(\boldsymbol{d}), \boldsymbol{y}(\boldsymbol{x}, \boldsymbol{l}_o), \boldsymbol{p}_o). \tag{10}$$

Concatenating p_o with the viewing direction $\boldsymbol{d}$ preserves the view-consistent behavior from the original architecture and adds consistency across poses for the volumetric density. We again apply a Fourier feature mapping to all low dimensional inputs $\boldsymbol{x}_o$, $\boldsymbol{d}_o$, $\boldsymbol{p}_o$. This results in the following volumetric scene function for an object class c

$$F_{\theta_c} : (\boldsymbol{x}_o, \boldsymbol{d}_o, \boldsymbol{p}_o, \boldsymbol{l}_o) \rightarrow (\boldsymbol{c}, \boldsymbol{\sigma}); \forall \boldsymbol{x}_o \in [-1, 1]. \tag{11}$$

4. Neural Scene Graph Rendering

The proposed neural scene graph describes a dynamic scene and a camera using a hierarchical structure. In this section, we show how this scene description can be used to render images of the scene, illustrated in Fig. 3, and how the representation networks at the leaf nodes are learned given a training set of images.

4.1. Rendering Pipeline

Images of the learned scene are rendered using a ray casting approach. The rays are generated through the camera defined in scene S at node C, its intrinsics $\boldsymbol{K}$, and the camera transformation $\boldsymbol{T}_c^{\mathcal{W}}$. We model the camera C with a

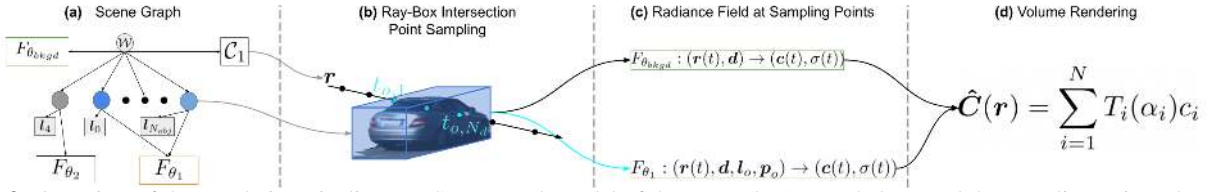


Figure 3: Overview of the Rendering Pipeline. (a) Scene graph model of the scene (b) A sampled ray and the sampling points along the ray inside the background (black) and an object (blue) node (c) Mapping of each point to a color and density value with the corresponding representation function F (d) Volume rendering integration over all sampling points along a ray from its origin to the far plane intersection.

pinhole camera model, tracing a set of rays $\boldsymbol{r} = \boldsymbol{o} + t\boldsymbol{d}$ through each pixel on the film of size $H \times W$. We sample points at all graph nodes a ray intersects with. At each point where a representation model is hit, a color and volumetric density are computed, and we calculate the pixel color by applying volumetric integration along the ray.

Multi-Plane Sampling To increase efficiency, we limit the sampling at the static node to multiple planes, resembling a 2.5 dimensional representation. We define N_s planes, parallel to the image plane of the initial camera pose $\boldsymbol{T}_{c,0}^{w}$ equispaced between the near clip d_n and far clip d_f distance. For a ray, $\boldsymbol{r}$, we calculate the intersections $\{t_i\}_{i=1}^{N_s}$ with each plane, instead of performing raymarching.

Ray-box Intersection For each ray, we predict color and density at each dynamic node the ray intersects with. We check each ray from the camera C for intersections with all dynamic nodes F_{θ_o}, by translating the ray to an object local frame and applying an efficient AABB-ray intersection test as proposed by Majercik et al. [18]. This computes all m ray-box entrance and exit points $(t_{o,1}, t_{o,N_d})$. For each pair of entrance and exit points, we sample N_d equidistant quadrature points

$$t_{o,n} = \frac{n-1}{N_d - 1}(t_{o,N_d} - t_{o,1}) + t_{o,1}, \tag{12}$$

and we sample from the model at $\boldsymbol{x}_o = \boldsymbol{r}(t_{o,n})$ and $\boldsymbol{d}_o$ with $n = [1, N_d]$. A small number of equidistant points $\boldsymbol{x}_o$ are enough to represent dynamic objects accurately while maintaining short rendering times.

Volumetric Rendering Each ray $\boldsymbol{r}_j$ traced through the scene is discretized at N_d sampling points at each of the m_j dynamic node intersections and at N_s planes, resulting in a set of quadrature points $\{\{t_i\}_{i=1}^{N_s+m_j N_d}\}_j$. The transmitted color $\boldsymbol{c}(\boldsymbol{r}(t_i))$ and volumetric density $\sigma(\boldsymbol{r}(t_i))$ at each intersection point is predicted from the respective radiance fields in the static background node $F_{\theta_{bckg}}$ or dynamic node F_{θ_c}. All sampling points and model outputs $(t, \sigma, \boldsymbol{c})$ are ordered along the ray $\boldsymbol{r}(t)$ as an ordered set

$$\{\{t_i, \sigma(\boldsymbol{x}_{j,i}), \boldsymbol{c}(\boldsymbol{x}_{j,i})\}_{i=1}^{N_s+m_j N_d} | t_{i-1} < t_i\}. \tag{13}$$

The pixel color is predicted using the rendering integral approximated with numerical quadrature as

$$\hat{\boldsymbol{C}}(\boldsymbol{r}) = \sum_{i=1}^{N_s+m_j N_d} T_i \alpha_i \boldsymbol{c}_i, \text{ where} \tag{14}$$

$$T_i = \exp(-\sum_{k=1}^{i-1} \sigma_k \delta_k) \text{ and } \alpha_i = 1 - \exp(-\sigma_i \delta_i), \tag{15}$$

where $\delta_i = t_{i+1} - t_i$ is the distance between adjacent points.

4.2. Joint Scene Graph Learning

For each dynamic scene, we optimize a set of representation networks at each node F. Our training set consists of N tuples $\{(\mathcal{I}_k, \mathcal{S}_k)\}_{k=1}^{N}$, the images $\mathcal{I}_k \in \mathbb{R}^{H \times W \times 3}$ of the scene. We sample rays for all camera nodes $\mathcal{C}_k$ for all frames k at each pixel j. From given 3D tracking data, we take the transformations $\boldsymbol{M}_v^u$ to form the reference scene graph edges. Pixel values $\hat{C}_{k,j}$ are predicted for all $H \times W \times N$ rays $\boldsymbol{r}_{k,j}$. We randomly sample a batch of rays $\mathcal{R}$ from all frames and associate each ray with the respective corresponding scene graphs $\mathcal{S}_k$ and ground truth pixel value $C_{k,j}$. We define the loss, in Eq. 16, as the total squared error between the predicted color $\hat{C}$ and the ground truth color C. As in DeepSDF [26], we assume a zero-mean multivariate Gaussian prior distribution over latent codes $p(\boldsymbol{z}_o)$ with a spherical covariance $\sigma^2 \boldsymbol{I}$. For these latent codes, we apply a regularization to all object descriptors with weight σ.

$$\mathcal{L} = \sum_{\boldsymbol{r} \in \mathcal{R}} \|\hat{C}(\boldsymbol{r}) - C(\boldsymbol{r})\|_2^2 + \frac{1}{\sigma^2}\|\boldsymbol{z}\|_2^2 \tag{16}$$

At each step, the gradient at each trainable node in L and F intersecting with the rays in $\boldsymbol{\mathcal{R}}$ is computed and back-propagated. The amount of nonzero gradients at a node in a step depends on the number of intersection points, leading to a varying amount of evaluation points across representation nodes. We balance this via ray sampling per batch.
We refer to the Supplemental Material for details on ray-plane and ray-box intersections and sampling.

5. Experiments

In this section, we validate the proposed neural scene graph method, by training neural scene graphs on existing automotive datasets. We then modify the learned graphs to synthesize unseen frames of novel object arrangements, temporal variations and renderings from novel views. We assess our approach with comparisons against state-of-the-art implicit neural rendering methods. The optimization for a scene takes about 36 hours or 400-500k iterations to converge on a single NVIDIA TITAN Xp GPU.

We choose to train our method on the KITTI dataset [11], and for experiments we use synthetic data from the Virtual

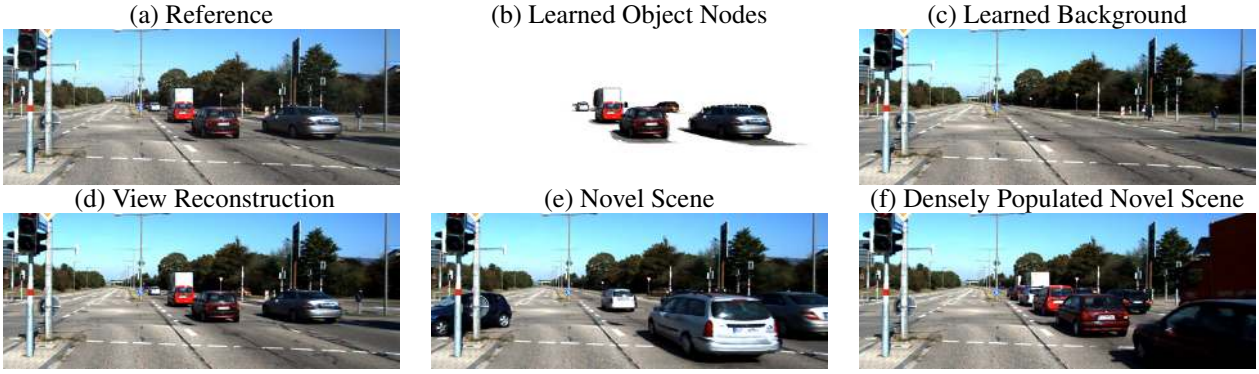


Figure 4: Renderings of a neural scene graph learned from a dynamic KITTI [11] scene. The representations of all objects and the background are trained on images from the scene in (a). The method naturally decomposes the representations into the background (c) and multiple object representations (b). Using learned scene graphs we can render reconstructions of the same scene as in (d), or unseen novel scene compositions with randomly sampled nodes and translations into the drivable space in (e) and (f).

KITTI 2 Dataset [5, 11], and for videos, we refer to our project page. Although the proposed method is not limited to automotive datasets, we use KITTI data as it has fueled research in object detection, tracking and scene understanding over the last decade. For each training scene, we optimize a neural scene graph with nodes for each tracked object and one for the static background. The transformation matrices $\boldsymbol{T}$ and $\boldsymbol{S}$ at the scene graph edges are computed from the tracking data. We use $N_s = 6$ planes and $N_d = 7$ object sampling points and a 256-dimensional latent vector l_o. The bounding box dimensions $\boldsymbol{s}_o$ are scaled to include the shadows of the object. We motivate the selected parameters and our design choices for $\boldsymbol{F}_{\theta_c}$ with an ablation study in the supplemental material. We train the scene graph using the Adam optimizer [14] with a linear learning rate decay.

5.1. Assessment

We present scene graph renderings for three scenes trained on dynamic tracking sequences from KITTI [11], captured using a stereo camera setup. Each sequence consists of up to 90 time steps and images of size 1242×375, each from two camera perspectives, and up to 12 unique, dynamic objects from different classes.

Foreground Background Decompositions Without additional supervision, the structure of the proposed scene graph naturally decomposes scenes into dynamic and static scene components. In Fig. 4, we present renderings of isolated nodes of a learned graph. Specifically, in Fig. 4 (c), we remove all dynamic nodes and render the image from a scene subgraph only consisting of the camera and background node. We observe that the rendered node only captures the static scene components. Similarly, we render dynamic parts in (b) from a subgraph excluding the static node. The shadow of each object is part of its dynamic representation and is visible in the rendering. Aside from decoupling the background and all dynamic parts, the method accurately reconstructs (d) the target image of the scene (a).

Figure 5: We synthesize multiple views of a learned scene graph leaf node by rotating yaw. The specular reflection on the trunk is maintained across rotations, demonstrating the ability of the method to learn view-dependent lighting.

Figure 6: A learned vehicle leaf node is translated by 2 meters perpendicular to the movement observed during training. The model changes the shadows and specular highlights on the vehicle consistent with learned global scene illumination.

Scene Graph Manipulations A learned neural scene graph can be manipulated at its edges and nodes. To demonstrate the flexibility of the proposed scene representation, we change the edge transformations of a learned node. Fig. 5 and 6 validate that these transformations preserve global light transport components such as reflections and shadows. The scene representation encodes global illumination cues implicitly through image color, a function of an object's location and viewing direction. Rotating a learned object node along its yaw axis in Fig. 5 validates that the specularity on the car trunk moves in accordance to fixed scene illumination, and retains a highlight with respect to the viewing direction. In Fig. 6, an object is translated away from its original position in the training set. In contrast to simply copying pixels, the model learns to correctly translate specular highlights after moving the vehicle.

Novel Scene Graph Compositions and View Synthesis In addition to pose manipulation and node removal from a learned scene graph, our method allows for constructing completely novel scene graphs, and novel view synthesis. Fig. 7 shows view synthesis results for novel scene graphs that were generated using randomly sampled objects and transformations. We constrain samples to the joint set of all road trajectories that were observed, which we define as the

Figure 7: Novel scene graph renderings. Rendered objects are learned from a subsequence of a scene from the KITTI data set [11]. The novel scene graphs are sampled from nodes and edges in the data set as well as new transformations sampled on the road lanes, not allowing for collisions between objects. Occlusion from the background and other objects can be observed in several frames.

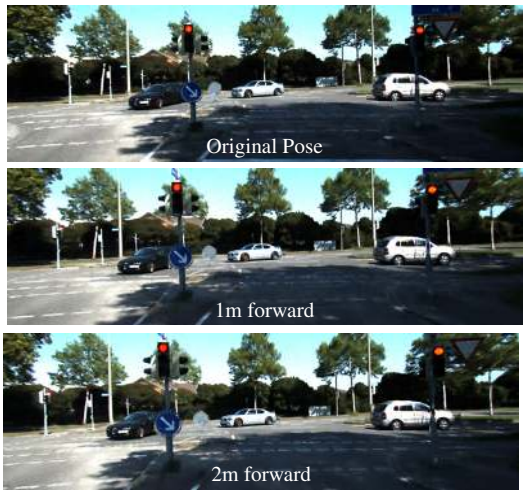


Figure 8: Unseen view synthesis results. Here we move the ego camera by approximately 2 m into the scene with all other nodes of the scene graph fixed. The method accurately handles occlusions from traffic lights and signs, highlighting the capabilities of the proposed scene graph for novel view synthesis.

driveable space. Note that these translations and arrangements have not been observed during training.

Similar to prior neural rendering methods, the proposed method allows for novel view synthesis after learning the scene representation. Fig. 8 shows novel views for ego-vehicle motion into the scene, where the ego-vehicle is driving $\approx$ 2 m forward. We refer to the Supplemental Material for additional view synthesis results.

Scene Graphs from Noisy Tracker Outputs Neural scene graphs can be learned from manual annotations or the outputs of tracking methods. While temporally filtered human labels exhibit low labeling noise [11], manual annotation can be prohibitively costly and time-consuming. We show that neural scene graphs learned from the output of existing tracking methods offer a representation quality comparable to the ones trained from manual annotations. Specifically, we compare renderings from scene graphs learned from KITTI [11] annotations and the following two tracking methods: we combine PointRCNN [30], a lidar point cloud object detection method, with the 3D tracker AB3DMOT [41], and we evaluate on labels from a camera-only tracker CenterTrack [47], i.e., resulting in a method solely relying on unannotated video frames. Fig. 9

Figure 9: Comparison of renderings learned from tracking labels and tracking data obtained from off-the-shelf trackers, see text.

shows qualitative results that indicate comparable neural scene graph quality when using different trackers. The monocular camera-only approach unsurprisingly degrades for objects at long distances, where the detection and tracking stack can no longer provide accurate depth information.

5.2. Quantitative Validation

We quantitatively validate our method with comparisons against Scene Representation Networks (SRNs) [32], NeRF [22], and a modified variant of NeRF on novel scene and reconstruction tasks. For comparison on the method complexity, we refer to the Supplemental Material.

Specifically, we learn neural scene graphs on video sequences of the KITTI data and assess the quality of reconstruction of seen frames using the learned graph, and novel scene compositions. SRNs and NeRF were designed for static scenes and are state-of-the-art approaches for implicit scene representations. To improve the capabilities of NeRF for dynamic scenes, we add a time parameter to the model inputs and denote this approach "NeRF + time". The time parameter is positionally encoded and concatenated with $\gamma_{\boldsymbol{x}}(\boldsymbol{x})$. We train both NeRF-based methods with the same configuration. Tab. 1 reports results which validate that complex dynamic scenes cannot be learned only by adding a time parameter to existing implicit methods.

Reconstruction Quality In Tab. 1, we evaluate all methods for scenes from KITTI and present quantitative results with the PSNR, SSIM[40], LPIPS [45] metrics. To also assess the consistency of the reconstruction for adjacent frames, we evaluate the renderings with two temporal metrics tOF and tLP [7]. The proposed method outperforms all baseline methods on all metrics. As shown in the top row of Fig. 10, vanilla SRN and NeRF implicit rendering fail to accurately reconstruct dynamic scenes. Optimizing SRNs leads to a single static representation for all frames. For NeRF, even small changes in the camera pose lead to a slightly different viewing direction for spatial points at all frames, and, as

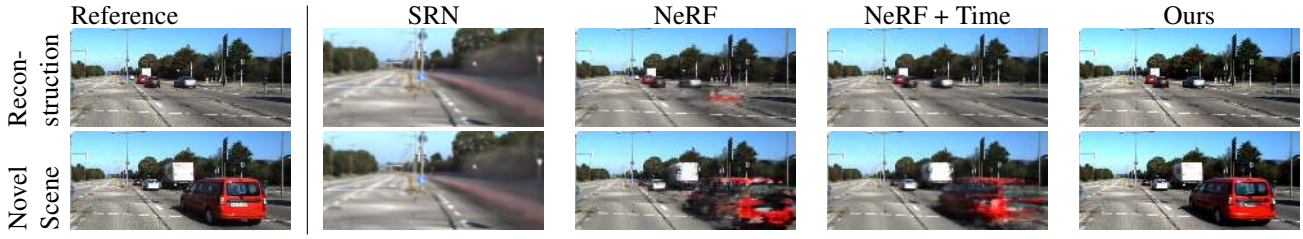


Figure 10: Qualitative results on reconstruction and novel scene arrangements of a scene from the KITTI dataset [11] for SRN [32], NeRF [22], a modified NeRF with a time parameter, and our neural scene graphs. Reconstruction here refers to the reproduction of a frame seen during training, and novel scene arrangements render a scene not seen in the training set. SRN learns to average all frames in the training set. NeRF and the modified variant struggle to learn dynamic parts of the scene adequately. Our neural scene graph method achieves high-quality view synthesis results regardless of these dynamics, allowing for high-quality dynamic scene reconstruction and novel scenes.

	SRN [32]	NeRF [22]	NeRF + time	Ours
Reconstruction				
PSNR ↑	18.83	23.34	24.18	**26.66**
SSIM ↑	0.590	0.662	0.677	**0.806**
LPIPS ↓	0.456	0.415	0.425	**0.186**
tOF $\times 10^6$ ↓	1.191	1.178	1.443	**0.765**
tLP $\times 100$ ↓	2.965	3.464	0.897	**0.246**
Novel Composition				
PSNR ↑	18.83	18.25	19.68	**25.11**
SSIM ↑	0.590	0.594	0.593	**0.789**
LPIPS ↓	0.456	0.442	0.473	**0.204**

Table 1: We report PSNR, SSIM, LPIPS, tOF and tLP results on scenes from KITTI [11] for SRN [32], NeRF [22], a modified NeRF for temporal scenes and our neural scene graph method. For PSNR and SSIM, higher is better; for LPIPS, tOF and tLP lower is better. Our method outperforms methods designed for static scenes for reconstructing dynamic scenes and for novel compositions in all image quality metrics.

such, the method suffers from severe ghosting. NeRF adjusted for temporal changes improves the quality, but still suffers from blurry, uncertain predictions. Significant improvement in the temporal metrics show that our method, which models each object individually, yields an improved temporal consistency for the reconstructions.

Novel Scene Compositions To compare the quality of novel scene compositions, we trained all methods on each image sequence leaving out a single frame from each. In Fig. 10 (bottom) we report results and Tab. 1 lists the corresponding evaluations. Even though NeRF and time-aware NeRF are able to recover changes in the scene when they occur with changing viewing direction, both methods are not able to reason about the scene dynamics. In contrast, the proposed method is able to adequately synthesize shadows and reflections without changing viewing direction.

6. 3D Object Detection as Inverse Rendering

The ability to decompose the scene into multiple objects in scene graphs also allows for improved scene understanding. In this section, we apply learned scene graphs to the problem of 3D object detection, using the proposed method as a forward model in a single shot learning approach.

We formulate 3D object detection as an image synthesis problem over the space of learned scene graphs that best reconstructs an observed image. Specifically, we sample anchor positions in a bird's-eye view plane and optimize over anchor box positions and latent object descriptors that minimize the ℓ_1 image loss between the rendering and the input. The resulting method is able to find the poses and dimensions of objects as shown in Fig. 11. This approach highlights the potential of neural rendering pipelines as a forward rendering model for computer vision tasks, a promising direction beyond view extrapolation and synthesis.

Figure 11: 3D Object detection with inverse neural scene graph rendering. The detected 3D object bounding boxes on the input images are the result of an optimization over the space of learned scene graphs. These outputs correspond to the scene graph that renders an image with a minimum distance to the observed image.

7. Conclusion

We present the first approach that tackles the challenge of representing dynamic, multi-object scenes implicitly using neural networks. Using video and tracking information, our method learns a graph-structured, spatial representation of multiple dynamic and static scene elements, automatically decomposing the scene into multiple independent representation nodes. Comprehensive experiments on simulated and real data achieve photo-realistic quality previously only attainable on static scenes. Building on these contributions, we also present the first method that tackles automotive object detection using a neural rendering approach.

Due to the nature of implicit methods, the learned representation quality of the proposed method is bounded by the variation and amount of training data. In the future, larger view extrapolations might be handled by scene priors learned from large-scale video datasets. We believe that the proposed approach opens up the field of neural rendering for dynamic scenes, and, encouraged by our detection results, may potentially serve as a path towards unsupervised training of computer vision models in the future.

Acknowledgements

Felix Heide was supported by an NSF CAREER Award (2047359) and a Sony Young Faculty Award.

References

[1] Sameer Agarwal, Yasutaka Furukawa, Noah Snavely, Ian Simon, Brian Curless, Steven M. Seitz, and Richard Szeliski. Building rome in a day. *Commun. ACM*, 54(10):105–112, Oct 2011. 1

[2] Kara-Ali Aliev, Artem Sevastopolsky, Maria Kolos, Dmitry Ulyanov, and Victor Lempitsky. Neural point-based graphics. 2020. 2

[3] Andrea L Ames, David R Nadeau, and John L Moreland. *The VRML 2.0 sourcebook*. John Wiley & Sons, Inc., 1997. 2

[4] Simone Bianco, Gianluigi Ciocca, and Davide Marelli. Evaluating the performance of structure from motion pipelines. *Journal of Imaging*, 4:98, 08 2018. 1

[5] Yohann Cabon, Naila Murray, and Martin Humenberger. Virtual kitti 2, 2020. 6

[6] Anpei Chen, Minye Wu, Yingliang Zhang, Nianyi Li, Jie Lu, Shenghua Gao, and Jingyi Yu. Deep surface light fields. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 1(1):1–17, Jul 2018. 2

[7] Mengyu Chu, You Xie, Laura Leal-Taixé, and Nils Thuerey. Temporally coherent gans for video super-resolution (tecogan). *arXiv preprint arXiv:1811.09393*, 1(2):3, 2018. 7

[8] J. Costeira and T. Kanade. A multi-body factorization method for motion analysis. In *Proceedings of IEEE International Conference on Computer Vision*, pages 1071–1076, 1995. 1

[9] S. Cunningham and M. Bailey. Lessons from scene graphs: using scene graphs to teach hierarchical modeling. *Comput. Graph.*, 25:703–711, 2001. 2

[10] John Flynn, Michael Broxton, Paul Debevec, Matthew DuVall, Graham Fyffe, Ryan Overbeck, Noah Snavely, and Richard Tucker. Deepview: View synthesis with learned gradient descent. *Proceedings IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2019. 2

[11] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 6, 7, 8

[12] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, USA, 2 edition, 2003. 1

[13] Heung-Yeung Shum, Qifa Ke, and Zhengyou Zhang. Efficient bundle adjustment with virtual key frames: a hierarchical approach to multi-frame structure from motion. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, volume 2, pages 538–543 Vol. 2, 1999. 1

[14] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015. 6

[15] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields, 2020. 2

[16] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *ACM Trans. Graph.*, 38(4):65:1–65:14, Jul 2019. 1, 2

[17] Erika Lu, Forrester Cole, Tali Dekel, Weidi Xie, Andrew Zisserman, David Salesin, William T. Freeman, and Michael Rubinstein. Layered neural rendering for retiming people in video, 2020. 2

[18] Alexander Majercik, Cyril Crassin, Peter Shirley, and Morgan McGuire. A ray-box intersection algorithm and efficient dynamic voxel rendering. *Journal of Computer Graphics Techniques (JCGT)*, 7(3):66–81, September 2018. 5

[19] Ricardo Martin-Brualla, Noha Radwan, Mehdi S. M. Sajjadi, Jonathan T. Barron, Alexey Dosovitskiy, and Daniel Duckworth. NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections. In *arXiv*, 2020. 1

[20] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2

[21] Ben Mildenhall, Pratul P. Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)*, 2019. 2

[22] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis, 2020. 1, 2, 3, 7, 8

[23] D. R. Nadeau. Volume scene graphs. In *2000 IEEE Symposium on Volume Visualization (VV 2000)*, pages 49–56, 2000. 2

[24] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[25] K. E. Ozden, K. Schindler, and L. Van Gool. Multibody structure-from-motion in practice. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(6):1134–1141, 2010. 1

[26] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2, 4, 5

[27] Francesco Pittaluga, Sanjeev J Koppal, Sing Bing Kang, and Sudipta N Sinha. Revealing scenes by inverting structure from motion reconstructions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 145–154, 2019. 2

[28] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-Motion Revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1

[29] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise View Selection for Unstructured Multi-View Stereo. In *Proceedings of the IEEE European Conf. on Computer Vision (ECCV)*, 2016. 1

[30] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point

cloud. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 7

[31] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhöfer. Deepvoxels: Learning persistent 3d feature embeddings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 2

[32] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In *Advances in Neural Information Processing Systems*, 2019. 2, 7, 8

[33] H. Sowizral. Scene graphs in the new millennium. *IEEE Computer Graphics and Applications*, 20(1):56–57, 2000. 2

[34] Henry A Sowizral, David R Nadeau, Michael J Bailey, and Michael F Deering. Introduction to programming with java 3d. *ACM SIGGRAPH 98 Course Notes*, 1998. 2

[35] P. P. Srinivasan, R. Tucker, J. T. Barron, R. Ramamoorthi, R. Ng, and N. Snavely. Pushing the boundaries of view extrapolation with multiplane images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 175–184, 2019. 2

[36] Shufeng Tan and Michael L. Mayrovouniotis. Reducing data dimensionality through optimizing neural network inputs. *AIChE Journal*, 41(6):1471–1480, 1995. 2

[37] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains, 2020. 3

[38] A. Tewari, O. Fried, J. Thies, V. Sitzmann, S. Lombardi, K. Sunkavalli, R. Martin-Brualla, T. Simon, J. Saragih, M. Nießner, R. Pandey, S. Fanello, G. Wetzstein, J.-Y. Zhu, C. Theobalt, M. Agrawala, E. Shechtman, D. B Goldman, and M. Zollhöfer. State of the Art on Neural Rendering. *Computer Graphics Forum (EG STAR 2020)*, 2020. 2

[39] Justus Thies, Michael Zollhöfer, and Matthias Nießner. Deferred neural rendering. *ACM Transactions on Graphics*, 38(4):1–12, Jul 2019. 2

[40] Z. Wang, Eero Simoncelli, and Alan Bovik. Multiscale structural similarity for image quality assessment. volume 2, pages 1398 – 1402 Vol.2, 12 2003. 7

[41] Xinshuo Weng, Jianren Wang, David Held, and Kris Kitani. AB3DMOT: A Baseline for 3D Multi-Object Tracking and New Evaluation Metrics. *ECCVW*, 2020. 7

[42] J. Wernecke. The inventor mentor - programming object-oriented 3d graphics with open inventor, release 2. 1993. 2

[43] Qiangeng Xu, Weiyue Wang, Duygu Ceylan, Radomir Mech, and Ulrich Neumann. Disn: Deep implicit surface network for high-quality single-view 3d reconstruction, 2019. 2

[44] Xinchen Yan, Jimei Yang, Ersin Yumer, Yijie Guo, and Honglak Lee. Perspective transformer nets: Learning single-view 3d object reconstruction without 3d supervision, 2016. 2

[45] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 7

[46] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. In *SIGGRAPH*, 2018. 2

[47] Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. Tracking objects as points. *Proceedings of the IEEE European Conf. on Computer Vision (ECCV)*, 2020. 7