SUDS: Scalable Urban Dynamic Scenes

Haithem Turki[1*] Jason Y. Zhang[1] Francesco Ferroni[2] Deva Ramanan[1]
[1]Carnegie Mellon University [2]Argo AI

Abstract

We extend neural radiance fields (NeRFs) to dynamic large-scale urban scenes. Prior work tends to reconstruct single video clips of short durations (up to 10 seconds). Two reasons are that such methods (a) tend to scale linearly with the number of moving objects and input videos because a separate model is built for each and (b) tend to require supervision via 3D bounding boxes and panoptic labels, obtained manually or via category-specific models. As a step towards truly open-world reconstructions of dynamic cities, we introduce two key innovations: (a) we factorize the scene into three separate hash table data structures to efficiently encode static, dynamic, and far-field radiance fields, and (b) we make use of unlabeled target signals consisting of RGB images, sparse LiDAR, off-the-shelf self-supervised 2D descriptors, and most importantly, 2D optical flow. Operationalizing such inputs via photometric, geometric, and feature-metric reconstruction losses enables SUDS to decompose dynamic scenes into the static background, individual objects, and their motions. When combined with our multi-branch table representation, such reconstructions can be scaled to tens of thousands of objects across 1.2 million frames from 1700 videos spanning geospatial footprints of hundreds of kilometers, (to our knowledge) the largest dynamic NeRF built to date. We present qualitative initial results on a variety of tasks enabled by our representations, including novel-view synthesis of dynamic urban scenes, unsupervised 3D instance segmentation, and unsupervised 3D cuboid detection. To compare to prior work, we also evaluate on KITTI and Virtual KITTI 2, surpassing state-of-the-art methods that rely on ground truth 3D bounding box annotations while being 10x quicker to train.

1. Introduction

Scalable geometric reconstructions of cities have transformed our daily lives, with tools such as Google Maps and Streetview [6] becoming fundamental to how we navigate and interact with our environments. A watershed moment

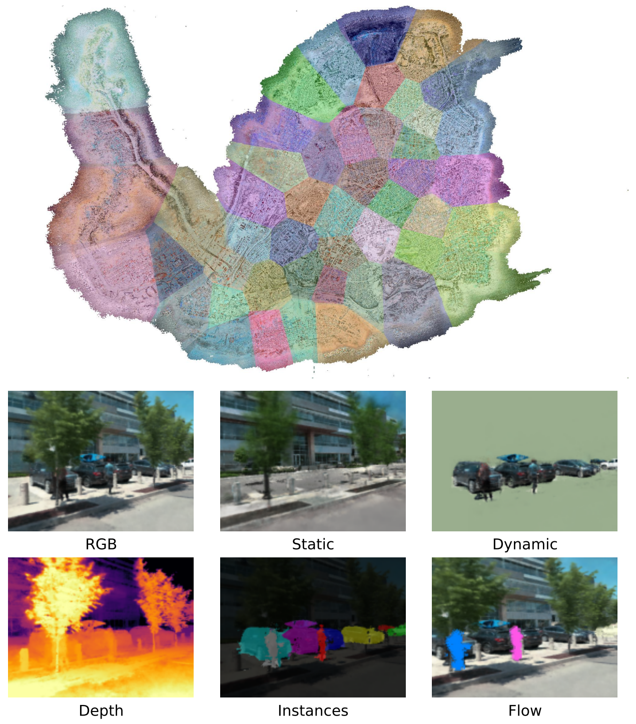


Figure 1. **SUDS.** We scale neural reconstructions to city scale by dividing the area into multiple cells and training hash table representations for each. We show our full city-scale reconstruction **above** and the derived representations **below.** Unlike prior methods, our approach handles dynamism across multiple videos, disentangling dynamic objects from static background and modeling shadow effects. We use unlabeled inputs to learn scene flow and semantic predictions, enabling category- and object-level scene manipulation.

in the development of such technology was the ability to scale structure-from-motion (SfM) algorithms to city-scale footprints [4]. Since then, the advent of Neural Radiance Fields (NeRFs) [33] has transformed this domain by allowing for photorealistic interaction with a reconstructed scene via view synthesis.

Recent works have attempted to scale such representations to neighborhood-scale reconstructions for virtual drive-throughs [47] and photorealistic fly-throughs [52]. However, these maps remain static and frozen in time. This makes capturing bustling human environments—

*Work done as an intern at Argo AI.

complete with moving vehicles, pedestrians, and objects—impossible, limiting the usefulness of the representation.

Challenges. One possible solution is a dynamic NeRF that conditions on time or warps a canonical space with a time-dependent deformation [38]. However, reconstructing dynamic scenes is notoriously challenging because the problem is inherently under-constrained, particularly when input data is constrained to limited viewpoints, as is typical from egocentric video capture [20]. One attractive solution is to scale up reconstructions to *many* videos, perhaps collected at different days (e.g., by an autonomous vehicle fleet). However, this creates additional challenges in jointly modeling fixed geometry that holds for all time (such as buildings), geometry that is locally static but *transient* across the videos (such as a parked car), and geometry that is truly dynamic (such as a moving person).

SUDS. In this paper, we propose SUDS: Scalable Urban Dynamic Scenes, a 4D representation that targets both *scale* and *dynamism*. Our key insight is twofold; (1) SUDS makes use of a rich suite of informative but freely available input signals, such as LiDAR depth measurements and optical flow. Other dynamic scene representations [27, 37] require supervised inputs such as panoptic segmentation labels or bounding boxes, which are difficult to acquire with high accuracy for our in-the-wild captures. (2) SUDS decomposes the world into 3 components: a *static* branch that models stationary topography that is consistent across videos, a *dynamic* branch that handles both transient (e.g., parked cars) and truly dynamic objects (e.g., pedestrians), and an *environment map* that handles far-field objects and sky. We model each branch using a multi-resolution hash table with scene partitioning, allowing SUDS to scale to an entire city spanning over 100 km^2.

Contributions. We make the following contributions: (1) to our knowledge, we build the first large-scale dynamic NeRF, (2) we introduce a scalable three-branch hash table representation for 4D reconstruction, (3) we present state-of-the-art reconstruction on 3 different datasets. Finally, (4) we showcase a variety of downstream tasks enabled by our representation, including free-viewpoint synthesis, 3D scene flow estimation, and even unsupervised instance segmentation and 3D cuboid detection.

2. Related Work

The original Neural Radiance Fields (NeRF) paper [33] inspired a wide body of follow-up work based on the original approach. Below, we describe a non-exhaustive list of such approaches along axes relevant to our work.

Scale. The original NeRF operated with bounded scenes. NeRF++ [63] and mip-NeRF 360 [7] use non-linear scene parameterization to model unbounded scenes. However, scaling up the size of the scene with a fixed size MLP leads to blurry details and training instability while the cost of naively increasing the size of the MLP quickly becomes intractable. BungeeNeRF [58] introduced a coarse-to-fine approach that progressively adds more capacity to the network representation. Block-NeRF [47] and Mega-NeRF [52] partition the scene spatially and train separate NeRFs for each partition. To model appearance variation, they incorporate per-image embeddings like NeRF-W [31]. Our approach similarly partitions the scene into sub-NeRFs, making use of depth to improve partition efficiency and scaling over an area 200x larger than Block-NeRF's Alamo Square Dataset. Both of these methods work only on static scenes.

Dynamics. Neural 3D Video Synthesis [28] and Space-time Neural Irradiance Fields [57] add time as an input to handle dynamic scenes. Similar to our work, NSFF [29], NeRFlow [15], and DyNeRF [19] incorporate 2D optical flow input and warping-based regularization losses to enforce plausible transitions between observed frames. Multiple methods [38–40, 50] instead disentangle scenes into a canonical template and per-frame deformation field. BANMo [60] further incorporates deformable shape models and canonical embeddings to train articulated 3D models from multiple videos. These methods focus on single-object scenes, and all but [28] and [60] use single video sequences.

While many of the previous works use segmentation data to factorize dynamic from static objects, D^{2}NeRF [56] does this automatically through regularization and explicitly handling shadows. Neural Groundplans [44] uses synthetic data to do this decomposition from a single image. We borrow some of these ideas and scale beyond synthetic and indoor scenes.

Object-centric approaches. Several approaches [24,36, 37, 59, 61, 62] represent scenes as the composition of per-object NeRF models and a background model. NSG [37] is most similar to us as it also targets automotive data but cannot handle ego-motion as our approach can. None of these methods target multi-video representations and are fundamentally constrained by the memory required to represent each object, with NSG needing over 1TB of memory to represent a 30 second video in our experience.

Semantics. Follow-up works have explored additional semantic outputs in addition to predicting color. Semantic-NeRF [65] adds an extra head to NeRF that predicts extra semantic category logits for any 3D position. Panoptic-NeRF [16] and Panoptic Neural Fields [27] extend this to produce panoptic segmentations and the latter uses a similar bounding-box based object and background decomposition as NSG. NeSF [53] generalizes the notion of a semantic field to unobserved scenes. As these methods are highly reliant on accurate annotations which are difficult to reliably obtain in the wild at our scale, we instead use a similar approach to recent works [26,51] that distill the outputs of 2D self-supervised feature descriptors into 3D radiance fields to enable semantic understanding without the use of human

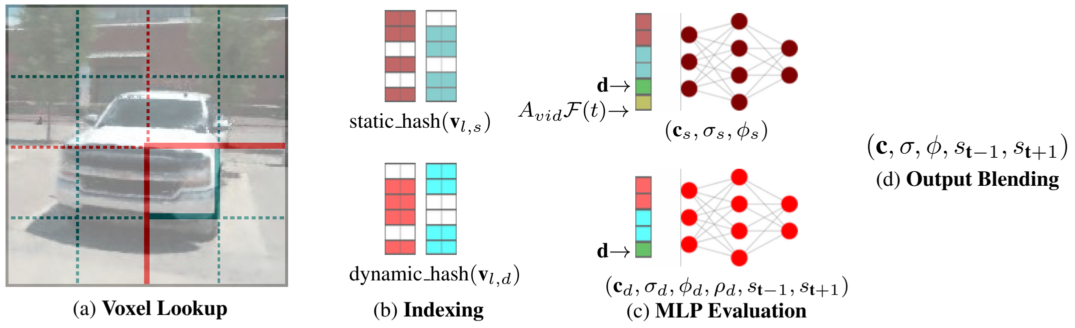

static_hash($\mathbf{v}_{l,s}$)
dynamic_hash($\mathbf{v}_{l,d}$)
$\mathbf{d}\rightarrow$
$A_{vid}\mathcal{F}(t)\rightarrow$
$(\mathbf{c}_s, \sigma_s, \phi_s)$
$\mathbf{d}\rightarrow$
$(\mathbf{c}_d, \sigma_d, \phi_d, \rho_d, s_{\mathbf{t-1}}, s_{\mathbf{t+1}})$
$(\mathbf{c}, \sigma, \phi, s_{\mathbf{t-1}}, s_{\mathbf{t+1}})$
(a) **Voxel Lookup**
(b) **Indexing**
(c) **MLP Evaluation**
(d) **Output Blending**

Figure 2. **Model Architecture.** (a) For a given input coordinate, we find the surrounding voxels at L resolution levels for both the static and dynamic branches (far-field branch omitted for clarity). (b) We assign indices to their corners by hashing based on position in the static branch and position, time, and video id in the dynamic branch. We look up the feature vectors corresponding to the corners and interpolate according to the relative position of the input coordinate within the voxel. (c) We concatenate the result of each level, along with auxiliary inputs such as viewing direction, and pass the resulting vector into an MLP to obtain per-branch color, density, and feature logits along with scene flow and the shadow ratio. (d) We blend scolor, opacity, and feature logits as the weighted sum of the branches.

labels and extend them to larger dynamic settings.

Fast training. The original NeRF took 1-2 days to train. Plenoxels [43] and DVGO [46] directly optimize a voxel representation instead of an MLP to train in minutes or even seconds. TensoRF [11] stores its representation as the outer product of low-rank tensors, reducing memory usage. Instant-NGP [35] takes this further by encoding features in a multi-resolution hash table, allowing training and rendering to happen in real-time. We use these tables as the base block of our three-branch representation and use our own hashing method to support dynamics across multiple videos.

Depth. Depth provides a valuable supervisory signal for learning high-quality geometry. DS-NeRF [14] and Dense Depth Priors [42] incorporate noisy point clouds obtained by structure from motion (SfM) in the loss function during optimization. Urban Radiance Fields [41] supervises with collected LiDAR data. We also use LiDAR but demonstrate results on dynamic environments.

3. Approach

3.1. Inputs

Our goal is to learn a global representation that facilitates free-viewpoint rendering, semantic decomposition, and 3D scene flow at arbitrary poses and time steps. Our method takes as input ordered RGB images from N videos (taken at different days with diverse weather and lighting conditions) and their associated camera poses. Crucially, we make use of additional data as "free" sources of supervision given contemporary sensor rigs and feature descriptors. Specifically, we use (1) aligned sparse LiDAR depth measurements, (2) 2D self-supervised pixel (DINO [10]) descriptors to enable semantic manipulation, and (3) 2D optical flow predictions to model scene dynamics. All model inputs are generated without any human labeling or intervention.

3.2. Representation

Preliminaries. We build upon NeRF [33], which represents a scene within a continuous volumetric radiance field that captures both geometry and view-dependent appearance. It encodes the scene within the weights of a multilayer perceptron (MLP). At render time, NeRF projects a camera ray $\mathbf{r}$ for each image pixel and samples along the ray, querying the MLP at sample position $\mathbf{x}_i$ and ray viewing direction $\mathbf{d}$ to obtain opacity and color values σ_i and $\mathbf{c}_i$. It then composites a color prediction $\hat{C}(\mathbf{r})$ for the ray using numerical quadrature $\sum_{i=0}^{N-1} T_i(1-\exp(-\sigma_i\delta_i))\,\mathbf{c}_i$, where $T_i = \exp(-\sum_{j=0}^{i-1}\sigma_j\delta_j)$ and δ_i is the distance between samples. The training process optimizes the model by sampling batches R of image pixels and minimizing the loss function $\sum_{\mathbf{r}\in\mathcal{R}} \left\|C(\mathbf{r}) - \hat{C}(\mathbf{r})\right\|^2$. NeRF samples rays through a two-stage hierarchical sampling process and uses frequency encoding to capture high-frequency details. We refer the reader to [33] for more details.

Scene composition. To model large-scale dynamic environments, SUDS factorizes the scene into three branches: (a) a static branch containing non-moving topography consistent across videos, (b) a dynamic branch to disentangle video-specific objects [19,29,56], moving or otherwise, and (c) a far-field environment map to represent far-away objects and the sky, which we found important to separately model in large-scale urban scenes [41,52,63].

However, conventional NeRF training with MLPs is computationally prohibitive at our target scales. Inspired by Instant-NGP [35], we implement each branch using multiresolution hash tables of F-dimensional feature vectors followed by a small MLP, along with our own hash functions to index across videos.

Hash tables (Fig. 2). For a given input coordinate $(\mathbf{x}, \mathbf{d}, \mathbf{t}, \mathbf{vid})$ denoting the position $\mathbf{x} \in \mathbb{R}^3$, viewing direction $\mathbf{d} \in \mathbb{R}^3$, frame index $F \in \{1, ..., T\}$, and video id

$\mathbf{vid} \in \{1, ..., N\}$, we find the surrounding voxels in each table at $l \in L$ resolution levels, doubling the resolution between levels, which we denote as $\mathbf{v}_{l,s}$, $\mathbf{v}_{l,d}$, $\mathbf{v}_{l,e}$ for the static, dynamic, and far-field. The static branch makes use of 3D spatial voxels $\mathbf{v}_{l,s}$, while the dynamic branch makes use of 4D spacetime voxels $\mathbf{v}_{l,d}$. Finally, the far-field branch makes use of 3D voxels $\mathbf{v}_{l,e}$ (implemented via normalized 3D direction vectors) that index an environment map. Similar to Instant-NGP [35], rather than storing features at voxel corners, we compute hash indices $\mathbf{i}_{l,s}$ (or $\mathbf{i}_{l,d}$ or $\mathbf{i}_{l,e}$) for each corner with the following hash functions:

$$\mathbf{i}_{l,s} = \text{static\_hash}(space(\mathbf{v}_{l,s})) \tag{1}$$
$$\mathbf{i}_{l,d} = \text{dynamic\_hash}(space(\mathbf{v}_{l,d}), time(\mathbf{v}_{l,d}), \mathbf{vid}) \tag{2}$$
$$\mathbf{i}_{l,e} = \text{env\_hash}(dir(\mathbf{v}_{l,e}), \mathbf{vid}) \tag{3}$$

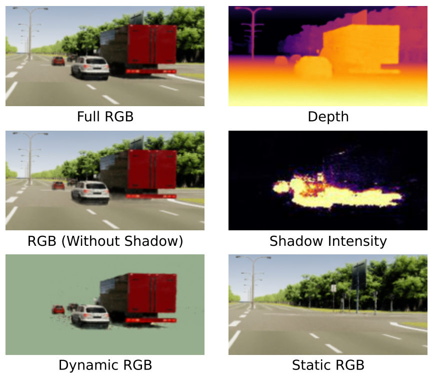
We linearly interpolate features up to the nearest voxel vertices (but now relying on *quad*linear interpolation for the dynamic 4D branch) and rely on gradient averaging to handle hash collisions. Finally, to model the fact that different videos likely contain distinct moving objects and illumination conditions, we add **vid** as an auxiliary input to the hash, but do *not* use it for interpolation (since averaging across distinct movers is unnatural). From this perspective, we leverage hashing to effectively index separate interpolating functions for each video, *without* a linear growth in memory with the number of videos. We concatenate the result of each level into a feature vector $f \in \mathbb{R}^{LF}$, along with auxiliary inputs such as viewing direction, and pass the resulting vector into an MLP to obtain per-branch outputs.

Static branch. We generate RGB images by combining the outputs of our three branches. The static branch maps the feature vector obtained from the hash table into a view-dependent color $\mathbf{c}_s$ and a view-independent density σ_s. To model lighting variations which could be dramatic across videos but smooth *within* a video, we condition on a latent embedding computed as a product of a video-specific matrix A_{vid} and a fourier-encoded time index $\mathcal{F}(t)$ (as in [60]):

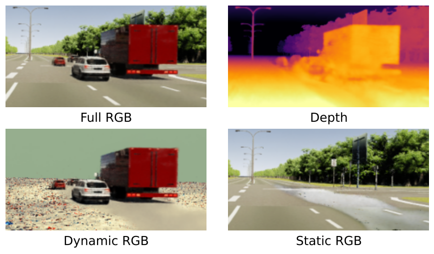
$$\sigma_s(\mathbf{x}) \in \mathbb{R} \tag{4}$$
$$\mathbf{c}_s(\mathbf{x}, \mathbf{d}, A_{vid}\mathcal{F}(t)) \in \mathbb{R}^3. \tag{5}$$

Dynamic branch. While the static branch assumes the density σ_s is static, the dynamic branch allows both the density σ_d and color $\mathbf{c}_d$ to depend on time (and video). We therefore omit the latent code when computing the dynamic radiance. Because we find shadows to play a crucial role in the appearance of urban scenes (Fig. 3), we explicitly model a *shadow field* of scalar values $\rho_d \in [0, 1]$, used to scale down the static color $\mathbf{c}_s$ (as done in [56]):

$$\sigma_d(\mathbf{x}, \mathbf{t}, \mathbf{vid}) \in \mathbb{R} \tag{6}$$
$$\rho_d(\mathbf{x}, \mathbf{t}, \mathbf{vid}) \in [0, 1] \tag{7}$$
$$\mathbf{c}_d(\mathbf{x}, \mathbf{t}, \mathbf{vid}, \mathbf{d}) \in \mathbb{R}^3 \tag{8}$$



(a) Shadow Field



(b) No Shadow Field

Figure 3. **Shadows.** We learn an explicit shadow field (**a**) as a pointwise reduction on static color, enabling better depth reconstruction and static/dynamic factorization than without (**b**).

Far-field branch. Because the sky requires reasoning about far-field radiance and because it can change dramatically across videos, we model far-field radiance with an environment map $\mathbf{c}_e(\mathbf{d}, \mathbf{vid}) \in \mathbb{R}^3$ that depends on viewing direction $\mathbf{d}$ [22, 41] and a video id $\mathbf{vid}$.

Rendering. We derive a single density and radiance value for any position by computing the weighted sum of the static and dynamic components, combined with the pointwise shadow reduction:

$$\sigma(\mathbf{x}, \mathbf{t}, \mathbf{vid}) = \sigma_s(\mathbf{x}) + \sigma_d(\mathbf{x}, \mathbf{t}, \mathbf{vid}) \tag{9}$$
$$\begin{aligned}\mathbf{c}(\mathbf{x}, \mathbf{t}, \mathbf{vid}, \mathbf{d}) = {} & \frac{\sigma_s}{\sigma}(1 - \rho_d)\mathbf{c}_s(\mathbf{x}, \mathbf{d}, A_{vid}\mathcal{F}(t)) \\ & + \frac{\sigma_d}{\sigma}\mathbf{c}_d(\mathbf{x}, \mathbf{t}, \mathbf{vid}, \mathbf{d})\end{aligned} \tag{10}$$

We then calculate the color $\hat{C}$ for a camera ray $\mathbf{r}$ with direction $\mathbf{d}$ at a given frame $\mathbf{t}$ and video $\mathbf{vid}$ by accumulating the transmittance along sampled points $\mathbf{r}(t)$ along the ray, forcing the ray to intersect the far-field environment map if it does not hit geometry within the foreground:

$$\begin{aligned}\hat{C}(\mathbf{r}, \mathbf{t}, \mathbf{vid}) = {} & \int_0^{+\infty} T(t)\sigma(\mathbf{r}(t), \mathbf{t}, \mathbf{vid})\mathbf{c}(\mathbf{r}(t), \mathbf{t}, \mathbf{vid}, \mathbf{d})dt \\ & + T(+\infty)\mathbf{c}_e(\mathbf{d}, \mathbf{vid}),\end{aligned} \tag{11}$$

$$\text{where } T(t) = \exp\left(-\int_0^t \sigma(\mathbf{r}(s), \mathbf{t}, \mathbf{vid})ds\right). \tag{12}$$

Feature distillation. We build semantic awareness into SUDS to enable the open-world tasks described in Sec. 4.2. Similar to recent work [26, 51], we distill the outputs of a self-supervised 2D feature extractor, namely DINO [10], as a teacher model into our network. For a feature extractor that transforms an image into a dense $\mathbb{R}^{H\times W\times C}$ feature grid, we add a C-dimensional output head to each of our branches:

$$\Phi_s(\mathbf{x}) \in \mathbb{R}^C \tag{13}$$
$$\Phi_d(\mathbf{x}, \mathbf{t}, \mathbf{vid}) \in \mathbb{R}^C \tag{14}$$
$$\Phi_e(\mathbf{d}, \mathbf{vid}) \in \mathbb{R}^C, \tag{15}$$

which are combined into a single value Φ at any 3D location and rendered into $\hat{F}(\mathbf{r})$ per camera ray, following the equations for color (10, 11).

Scene flow. We train our model to predict 3D scene flow and model scene dynamics. Inspired by previous work [15, 19, 29], we augment our dynamic branch to predict forward and backward 3D scene flow vectors $s_{t'\in[-1,1]}(\mathbf{x}, \mathbf{t}, \mathbf{vid}) \in \mathbb{R}^3$. We make use of these vectors to enforce consistency between observed time steps through multiple loss terms (Sec. 3.3), which we find crucial to generating plausible renderings at novel time steps (Table 4).

Spatial partitioning. We scale our representation to arbitrarily large environments by decomposing the scene into individually trained models [47, 52], each with its own static, dynamic, and far-field branch. Intuitively, the reconstruction for neighborhood X can be done largely independantly of the reconstruction in neighborhood Y, provided one can assign the relevant input data to each reconstruction. To do so, we follow the approach of Mega-NeRF [52] and split the scene into K spatial cells with centroids $k \in \mathbb{R}^3$. Crucially, we generate separate training datasets for each spatial cell by making use of *visibility* reasoning [17]. Mega-NeRF includes only those datapoints whose associated camera rays intersect the spatial cell. However, this may still include datapoints that are not visible due to an intervening occluder (e.g., a particular camera in neighborhood X can be *pointed* at neighborhood Y, but may not see anything there due to occluding buildings). To remedy this, we make use of depth measurements to prune irrelevant pixel rays that do not terminate within the spatial cell of interest (making use of nearest-neighbor interpolation to impute depth for pixels without a LiDAR depth measurement). This further reduces the size of each trainset by 2x relative to Mega-NeRF. Finally, given such separate reconstructions, one can still produce a globally consistent rendering by querying the appropriate spatial cell when sampling points along new-view rays (as in [52]).

3.3. Optimization

We jointly optimize all three of our model branches along with the per-video weight matrices A_{vid} by sampling random batches of rays across our N input videos and minimizing the following loss:

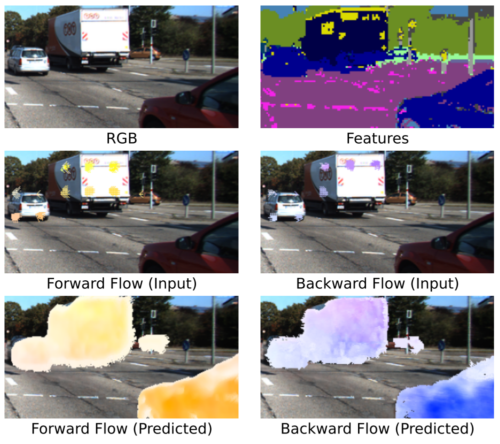


Figure 4. **Scene Flow.** We minimize the photometric and feature-metric loss of warped renderings relative to ground truth inputs **(top)**. We use 2D optical flow from off-the-shelf estimators or sparse correspondences computed directly from 2D DINO features [5] **(middle)** to supervise our flow predictions **(bottom)**.

$$\mathcal{L} = \underbrace{\left(\mathcal{L}_c + \lambda_f\mathcal{L}_f + \lambda_d\mathcal{L}_d + \lambda_o\mathcal{L}_o\right)}_{\text{reconstruction losses}} + \underbrace{\left(\mathcal{L}_c^w + \lambda_f\mathcal{L}_f^w\right)}_{\text{warping losses}}$$
$$\lambda_{flo}\underbrace{\left(\mathcal{L}_{cyc} + \mathcal{L}_{sm} + \mathcal{L}_{slo}\right)}_{\text{flow losses}} + \underbrace{\left(\lambda_e\mathcal{L}_e + \lambda_d\mathcal{L}_d\right)}_{\text{static-dynamic factorization}} + \lambda_\rho\mathcal{L}_\rho. \tag{16}$$

Reconstruction losses. We minimize the L2 photometric loss $\mathcal{L}_c(\mathbf{r}) = \left\|C(\mathbf{r}) - \hat{C}(\mathbf{r})\right\|^2$ as in the original NeRF equation [33]. We similarly minimize the L1 difference $\mathcal{L}_f(\mathbf{r}) = \left\|F(\mathbf{r}) - \hat{F}(\mathbf{r})\right\|_1$ between the feature outputs of the teacher model and that of our network.

To make use of our depth measurements, we project the LiDAR sweeps onto the camera plane and compare the expected depth $\hat{D}(r)$ with the measurement $D(\mathbf{r})$ [14, 41]:

$$\mathcal{L}_d(\mathbf{r}) = \left\|D(\mathbf{r}) - \hat{D}(\mathbf{r})\right\|^2 \tag{17}$$
$$\text{where } \hat{D}(\mathbf{r}) = \int_0^{+\infty} T(s)\sigma(\mathbf{r}(s))ds \tag{18}$$

Flow. We supervise our 3D scene flow predictions based on 2D optical flow (Sec. 4.1). We generate a 2D displacement vector for each camera ray by first predicting its position in 3D space as the weighted sum of the scene flow neighbors along the ray:

$$\hat{X}_{t'}(\mathbf{r}) = \int_0^{+\infty} T(t)\sigma(r(t))(r(t) + s_{t'}(\mathbf{r}(t)))dt \tag{19}$$

which we then "render" into 2D using the camera matrix of the neighboring frame index. We minimize its distance from the observed optical flow via $\mathcal{L}_o(\mathbf{r}) =$

$\sum_{t' \in [-1,1]} \left\| X(\mathbf{o}) - \hat{X}_{t'}(\mathbf{r}) \right\|_1$. We anneal λ_o over time as these estimates are noisy.

3D warping. The above loss ensures that rendered 3D flow will be consistent with the observed 2D flow. We also found it useful to enforce 3D color (and feature) constancy; i.e., colors remain constant even when moving. To do so, we use the predicted forward and backward 3D flow $s_{\mathbf{t}+1}$ and $s_{\mathbf{t}-1}$ to *advect* each sample along the ray into the next/previous frame:

$$\sigma_{t'}^{w}(\mathbf{x} + s_{t'}, \mathbf{t} + t', \mathbf{vid}) \in \mathbb{R} \tag{20}$$

$$\mathbf{c}_{t'}^{w}(\mathbf{x} + s_{t'}, \mathbf{t} + t', \mathbf{vid}, \mathbf{d}) \in \mathbb{R}^3 \tag{21}$$

$$\Phi_{t'}^{w}(\mathbf{x} + s_{t'}, \mathbf{t} + t', \mathbf{vid}) \in \mathbb{R}^C \tag{22}$$

The warped radiance $\mathbf{c}^w$ and density σ^w are rendered into warped color $\hat{C}^w(\mathbf{r})$ and feature $\hat{F}^w(\mathbf{r})$ (10, 11). We add a loss to ensure that the warped color (and feature) match the ground-truth input for the current frame, similar to [19, 29]. As in NSFF [29], we found it important to downweight this loss in ambiguous regions that may contain occlusions. However, instead of learning explicit occlusion weights, we take inspiration from Kwea's method [1] and use the difference between the dynamic geometry and the warped dynamic geometry to downweight the loss:

$$w_{t'}(\mathbf{x}, \mathbf{t}, \mathbf{vid}) = \left| \frac{\sigma_d}{\sigma} - \frac{\sigma_{t'}^{w}}{\sigma} \right| \tag{23}$$

$$\hat{W}_{t'}(\mathbf{r}) = \int_0^{+\infty} T(t)\sigma(r(t))w_{t'}(r(t))dt \tag{24}$$

resulting in the following warping loss terms:

$$\mathcal{L}_c^w(\mathbf{r}) = \sum_{t' \in [-1,1]} (1 - W_{t'})(\mathbf{r})) \left\| C(\mathbf{r}) - \hat{C}_{t'}^{w}(\mathbf{r}) \right\|^2 \tag{25}$$

$$\mathcal{L}_f^w(\mathbf{r}) = \sum_{t' \in [-1,1]} (1 - W_{t'})(\mathbf{r}) \left\| F(\mathbf{r}) - \hat{F}_{t'}^{w}(\mathbf{r}) \right\|_1 \tag{26}$$

Flow regularization. As in prior work [19, 29] we use a 3D scene flow cycle term to encourage consistency between forward and backward scene flow predictions, downweighing the loss in areas ambiguous due to occlusions:

$$\mathcal{L}_{cyc}(\mathbf{r}) = \sum_{t' \in [-1,1]} \sum_{\mathbf{x}} w_{t'}(\mathbf{x}, \mathbf{t}) \left\| s_{t'}(\mathbf{x}, \mathbf{t}) + s_{\mathbf{t}}(\mathbf{x} + s_{t'}, \mathbf{t} - t') \right\|_1, \tag{27}$$

with **vid** omitted for brevity. We also encourage spatial and temporal smoothness through $\mathcal{L}_{sm}(\mathbf{r})$ as described in Sec. C. We finally regularize the magnitude of predicted scene flow vectors to encourage the scene to be static through $\mathcal{L}_{slo}(\mathbf{r}) = \sum_{t' \in [\mathbf{t}-1, \mathbf{t}+1]} \sum_{\mathbf{x}} \left\| s_{t'}(\mathbf{x}, \mathbf{t}) \right\|_1$.

Static-dynamic factorization. As physically plausible solutions should have any point in space occupied by *either* a static or dynamic object, we encourage the spatial ratio of static vs dynamic density to either be 0 or 1 through a skewed binary entropy loss that favors static explanations of the scene [56]:

$$\mathcal{L}_e(\mathbf{r}) = \int_0^{+\infty} H\left(\frac{\sigma_d(\mathbf{r}(t))}{\sigma_s(\mathbf{r}(t)) + \sigma_d(\mathbf{r}(t))}^{k} \right) dt \tag{28}$$

$$\text{where } H(x) = -(x \cdot log(x) + (1-x) \cdot log(1-x)),$$

and with k set to 1.75, and further penalize the maximum dynamic ratio $\mathcal{L}_d(\mathbf{r}) = \max(\frac{\sigma_d(\mathbf{r}(t))}{\sigma_s + \sigma_d})$ along each ray.

Shadow loss. We penalize the squared magnitude of the shadow ratio $\mathcal{L}_\rho(\mathbf{r}) = \int_0^{+\infty} \rho_d(\mathbf{r}(t))^2 \, dt$ along each ray to prevent it from over-explaining dark regions [56].

4. Experiments

We demonstrate SUDS's city-scale reconstruction capabilities by presenting quantitative results against baseline methods (Table 1). We also show initial qualitative results for a variety of downstream tasks (Sec. 4.2). Even though we focus on reconstructing dynamic scenes at city scale, to faciliate comparisons with prior work, we also show results on small-scale but highly-benchmarked datasets such as KITTI and Virtual KITTI 2 (Sec. 4.3). We evaluate the various components of our method in Sec. 4.4.

4.1. Experimental Setup

2D feature extraction. We use Amir et al's feature extractor implementation [5] based on the dino_vits8 model. We downsample our images to fit into GPU memory and then upsample with nearest neighbor interpolation. We L2-normalize the features at the 11th layer of the model and reduce the dimensionality to 64 through incremental PCA [3].

Flow supervision. We explored using an estimator trained on synthetic data [49] in addition to directly computing 2D correspondences from DINO itself [5]. Although the correspondences are sparse (less than 5% of pixels) and expensive to compute, we found its estimates more robust and use it for our experiments unless otherwise stated.

Training. We train SUDS for 250,000 iterations with 4098 rays per batch and use a proposal sampling strategy similar to Mip-NeRF 360 [7] (Sec. B). We use Adam [25] with a learning rate of 5×10^{-3} decaying to 5×10^{-4}.

Metrics. We report quantitative results based on PSNR, SSIM [54], and the AlexNet implementation of LPIPS [64].

4.2. City-Scale Reconstruction

City-1M dataset. We evaluate SUDS's large-scale reconstruction abilities on our collection of 1.28 million images across 1700 videos gathered across a 105 km^2 urban area using a vehicle-mounted platform with seven ring cameras and two LiDAR sensors. Due to the scale, we supervise optical flow with an off-the-shelf estimator trained on synthetic data [49] instead of DINO for efficiency.

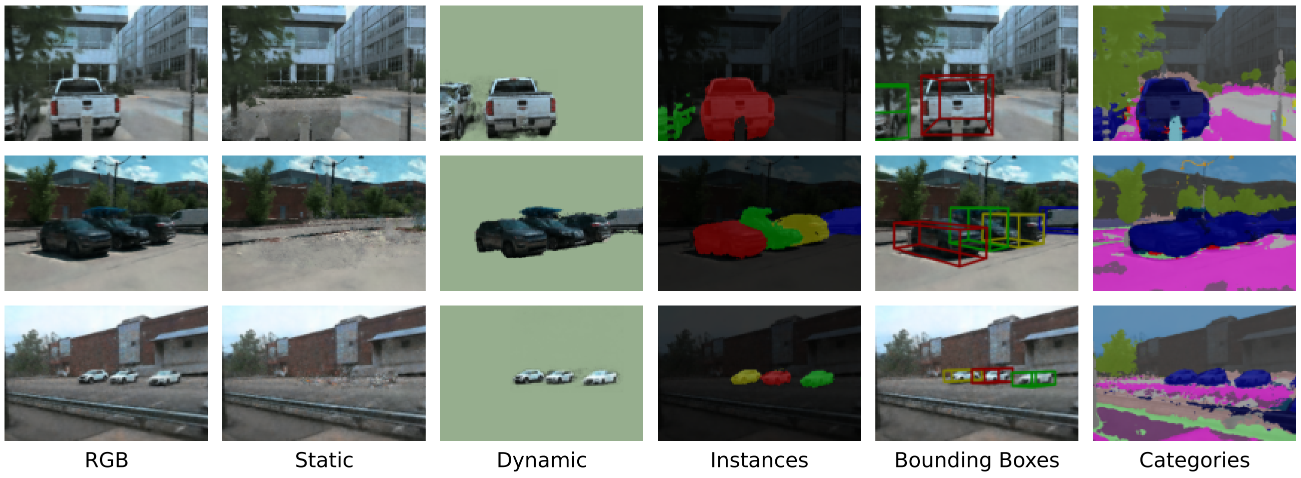


Figure 5. **City-1M.** We demonstrate SUDS's capabilities on multiple downstream tasks, including instance segmentation and 3D bounding box estimation without any labeled data (by just making use of geometric clustering). In the last column, we show category-level semantic classification by matching 3D (DINO) descriptors to a held-out video annotated with semantic labels. Please see text for more details.

	Mega-NeRF [52]	Mega-NeRF-T	Mega-NeRF-A	SUDS
PSNR ↑	16.42	16.46	16.70	**21.67**
SSIM ↑	0.493	0.493	0.493	**0.562**
LPIPS ↓	0.879	0.877	0.850	**0.554**

Table 1. **City-scale view synthesis on City-1M.** SUDS outperforms all baselines by a wide margin.

Baselines. We compare SUDS to the official Mega-NeRF [52] implementation alongside two variants: Mega-NeRF-T which directly adds time as an input parameter to compute density and radiance, and Mega-NeRF-A which instead uses the latent embedding $A_{vid}\mathcal{F}(t)$ used by SUDS.

Results. We train both SUDS and the baselines using 48 cells and summarize our results in Table 1. SUDS outperforms all Mega-NeRF variants by a large margin. We provide qualitative results on view synthesis, static/dynamic factorization, unsupervised 3D instance segmentation and unsupervised 3D cuboid detection in Fig. 5. We present additional qualititive tracking results in Fig. 7.

Instance segmentation. We derive the instance count as in prior work [44] by sampling dynamic density values σ_d, projecting those above a given threshold onto a discretized ground plane before applying connected component labeling. We apply k-means to obtain 3D centroids and volume render instance predictions as for semantic segmentation.

3D cuboid detection. After computing point-wise instance assignments in 3D, we derive oriented bounding boxes based on the PCA of the convex hull of points belonging to each instance [2].

Semantic segmentation. Note the above tasks of instance segmentation and 3D cuboid detection do not require any additional labels as they make use of geometric clustering. We now show that the representation learned by SUDS can also enable downstream semantic tasks, by making use of a small number of 2D segmentation labels provided on a held-out video sequence. We compute the average 2D DINO descriptor for each semantic class from the held out frames and derive 3D semantic labels for all reconstructions by matching each 3D descriptor to the closest class centroid. This allows to produce 3D semantic label fields that can then be rendered in 2D as shown in Fig. 5.

4.3. KITTI Benchmarks

Baselines. We compare SUDS to SRN [45], the original NeRF implementation [33], a variant of NeRF taking time as an additional input, NSG [37], and PNF [27]. Both NSG and PNF are trained and evaluated using ground truth object bounding box and category-level annotations.

Image reconstruction. We compare SUDS's reconstruction capabilities using the same KITTI [21] subsequences and experimental setup as prior work [27, 37]. We present results in Table 3. As PNF's implementation is not publicly available, we rely on their reported numbers. SUDS surpasses the state-of-the-art in PSNR and SSIM.

Novel view synthesis. We demonstrate SUDS's capabilities to generate plausible renderings at time steps unseen during training. As NSG does not handle scenes with ego-motion, we use subsequences of KITTI and Virtual KITTI 2 [18] with little camera movement. We evaluate the methods using different train/test splits, holding out every 4th time step, every other time step, and finally training with only one in every four time steps. We summarize our findings in Table 2 along with qualitative results in Fig. 6. SUDS achieves the best results across all splits and metrics. Both NeRF variants fail to properly represent the scene, especially in dynamic areas. Although we provide NSG with the ground truth object poses at render time, it fails to learn a clean decomposition between objects and the background,

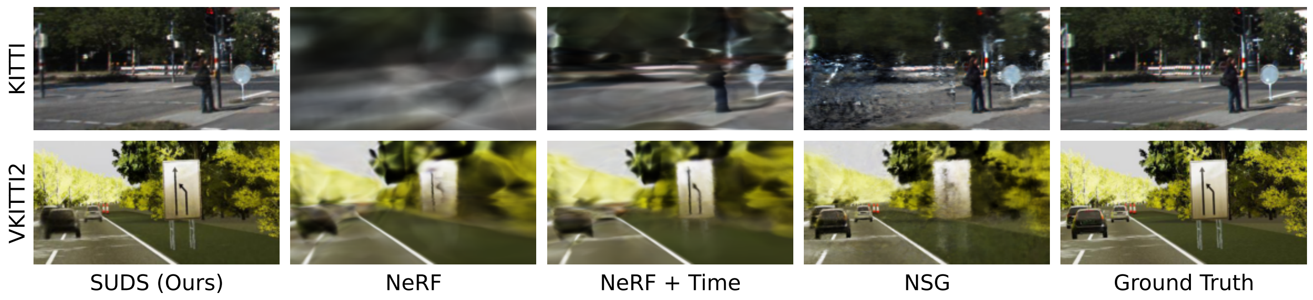


Figure 6. **KITTI and VKITTI2 view synthesis**. Prior work fails to represent the scene and NSG [37] renders ghosting artifacts near areas of movement. Our method forecasts plausible trajectories and generates higher-quality renderings.

	KITTI - 75%			KITTI - 50%			KITTI - 25%		
	↑PSNR	↑SSIM	↓LPIPS	↑PSNR	↑SSIM	↓LPIPS	↑PSNR	↑SSIM	↓LPIPS
NeRF [33]	18.56	0.557	0.554	19.12	0.587	0.497	18.61	0.570	0.510
NeRF + Time	21.01	0.612	0.492	21.34	0.635	0.448	19.55	0.586	0.505
NSG [37]	21.53	0.673	0.254	21.26	0.659	0.266	20.00	0.632	0.281
SUDS	**22.77**	**0.797**	**0.171**	**23.12**	**0.821**	**0.135**	**20.76**	**0.747**	**0.198**
	VKITTI2 - 75%			VKITTI2 - 50%			VKITTI2 - 25%		
	↑PSNR	↑SSIM	↓LPIPS	↑PSNR	↑SSIM	↓LPIPS	↑PSNR	↑SSIM	↓LPIPS
NeRF [33]	18.67	0.548	0.634	18.58	0.544	0.635	18.17	0.537	0.644
NeRF + Time	19.03	0.574	0.587	18.90	0.565	0.610	18.04	0.545	0.626
NSG [37]	23.41	0.689	0.317	23.23	0.679	0.325	21.29	0.666	0.317
SUDS	**23.87**	**0.846**	**0.150**	**23.78**	**0.851**	**0.142**	**22.18**	**0.829**	**0.160**

Table 2. **Novel View Synthesis.** As the fraction of training views decreases, accuracy drops for all methods. However, SUDS consistently outperforms prior work, presumably due to more accurate representations learned by our diverse input signals (such as depth and flow).

	SRN [45]	NeRF [33]	NeRF + Time	NSG [37]	PNF [27]	Ours
PSNR ↑	18.83	23.34	24.18	26.66	27.48	**28.31**
SSIM ↑	0.590	0.662	0.677	0.806	0.870	**0.876**

Table 3. **KITTI image reconstruction.** We outperform past work on image reconstruction accuracy, following their experimental protocol and self-reported accuracies [27, 37].

	↑PSNR	↑SSIM	↓LPIPS
w/o Depth loss	22.74	0.715	0.292
w/o Optical flow loss	22.18	0.708	0.302
w/o Warping loss	17.53	0.622	0.478
w/o Appearance embedding	22.54	0.704	0.296
w/o Occlusion weights	22.56	0.711	0.297
w/o Separate branches	19.73	0.570	0.475
Full Method	**22.95**	**0.718**	**0.289**

Table 4. **Diagnostics.** Flow-based warping is the single-most important input, while depth is the least crucial input.

especially as the number of training view decreases, and generates ghosting artifacts near areas of movement.

4.4. Diagnostics

We ablate the importance of major SUDS components by removing their respective loss terms along with occlusion weights, the latent embedding $A_{vid}\mathcal{F}(t)$ used to compute static color $\mathbf{c}_s$, and separate model branches (Sec. D). We run all approaches for 125,000 iterations across our datasets and summarize the results in Table 4. Although all components help performance, flow-based warping is by far the single most important input. Interestingly, depth is the least crucial input, suggesting that SUDS can generalize to settings where depth measurements are not available.

5. Conclusion

We present a modular approach towards building dynamic neural representations at previously unexplored scale. Our multi-branch hash table structure enables us to disentangle and efficiently encode static geometry and transient objects across thousands of videos. SUDS makes use of unlabeled inputs to learn semantic awareness and scene flow, allowing it to perform several downstream tasks while surpassing state-of-the-art methods that rely on human labeling. Although we present a first attempt at building city-scale dynamic environments, many open challenges remain ahead of building truly photorealistic representations.

Acknowledgments

This research was supported by the CMU Argo AI Center for Autonomous Vehicle Research.

References

[1] Kwea123's nsff implementation. https://github.com/kwea123/nsff_pl. Accessed: 2022-10-29. 6

[2] Open3d oriented bounding box implementation. http://www.open3d.org/docs/latest/python_api/open3d.geometry.OrientedBoundingBox.html#open3d.geometry.OrientedBoundingBox.create_from_axis_aligned_bounding_box. Accessed: 2022-11-06. 7

[3] Scikit incremental pca. https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.IncrementalPCA.html. Accessed: 2022-10-29. 6

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

[5] Shir Amir, Yossi Gandelsman, Shai Bagon, and Tali Dekel. Deep vit features as dense visual descriptors. *arXiv preprint arXiv:2112.05814*, 2021. 5, 6, 13

[6] Dragomir Anguelov, Carole Dulong, Daniel Filip, Christian Frueh, Stéphane Lafon, Richard Lyon, Abhijit Ogale, Luc Vincent, and Josh Weaver. Google street view: Capturing the world at street level. *Computer*, 43(6):32–38, 2010. 1

[7] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *CVPR*, 2022. 2, 6, 12

[8] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000. 13

[9] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013. 13

[10] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. 2021. 3, 5

[11] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *ECCV*, 2022. 3

[12] Shin-Fang Chng, Sameera Ramasinghe, Jamie Sherrah, and Simon Lucey. Gaussian activated neural radiance fields for high fidelity reconstruction and pose estimation. In *ECCV*, page 264–280, Berlin, Heidelberg, 2022. Springer-Verlag. 13

[13] Tali Dekel, Shaul Oron, Michael Rubinstein, Shai Avidan, and William T. Freeman. Best-buddies similarity for robust template matching. In *CVPR*, pages 2021–2029, 2015. 12

[14] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised NeRF: Fewer views and faster training for free. In *CVPR*, June 2022. 3, 5

[15] Yilun Du, Yinan Zhang, Hong-Xing Yu, Joshua B. Tenenbaum, and Jiajun Wu. Neural radiance flow for 4d view synthesis and video processing. In *ICCV*, 2021. 2, 5

[16] Xiao Fu, Shangzhan Zhang, Tianrun Chen, Yichong Lu, Lanyun Zhu, Xiaowei Zhou, Andreas Geiger, and Yiyi Liao. Panoptic nerf: 3d-to-2d label transfer for panoptic urban scene segmentation. In *International Conference on 3D Vision (3DV)*, 2022. 2

[17] Thomas A Funkhouser, Carlo H Sequin, and Seth J Teller. Management of large amounts of data in interactive building walkthroughs. In *Proceedings of the 1992 symposium on Interactive 3D graphics*, pages 11–20, 1992. 5

[18] Adrien Gaidon, Qiao Wang, Yohann Cabon, and Eleonora Vig. Virtual worlds as proxy for multi-object tracking analysis. In *CVPR*, pages 4340–4349, 2016. 7, 13

[19] Chen Gao, Ayush Saraf, Johannes Kopf, and Jia-Bin Huang. Dynamic view synthesis from dynamic monocular video. In *ICCV*, 2021. 2, 3, 5, 6

[20] Hang Gao, Ruilong Li, Shubham Tulsiani, Bryan Russell, and Angjoo Kanazawa. Monocular dynamic view synthesis: A reality check. In *NeurIPS*, 2022. 2

[21] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012. 7, 13

[22] Zekun Hao, Arun Mallya, Serge Belongie, and Ming-Yu Liu. GANcraft: Unsupervised 3D Neural Rendering of Minecraft Worlds. In *ICCV*, 2021. 4

[23] Yoonwoo Jeong, Seokjun Ahn, Christopher Choy, Anima Anandkumar, Minsu Cho, and Jaesik Park. Self-calibrating neural radiance fields. In *ICCV*, pages 5846–5854, October 2021. 13

[24] Zhang Jiakai, Liu Xinhang, Ye Xinyi, Zhao Fuqiang, Zhang Yanshun, Wu Minye, Zhang Yingliang, Xu Lan, and Yu Jingyi. Editable free-viewpoint video using a layered neural representation. In *ACM SIGGRAPH*, 2021. 2

[25] DP Kingma, J Ba, Y Bengio, and Y LeCun. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations*, 2015. 6

[26] Sosuke Kobayashi, Eiichi Matsumoto, and Vincent Sitzmann. Decomposing nerf for editing via feature field distillation. In *Advances in Neural Information Processing Systems*, volume 35, 2022. 2, 5

[27] Abhijit Kundu, Kyle Genova, Xiaoqi Yin, Alireza Fathi, Caroline Pantofaru, Leonidas Guibas, Andrea Tagliasacchi, Frank Dellaert, and Thomas Funkhouser. Panoptic Neural Fields: A Semantic Object-Aware Neural Scene Representation. In *CVPR*, 2022. 2, 7, 8

[28] Tianye Li, Mira Slavcheva, Michael Zollhoefer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, and Zhaoyang Lv. Neural 3d video synthesis. In *CVPR*, 2022. 2

[29] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *CVPR*, 2021. 2, 3, 5, 6, 12

[30] Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. Barf: Bundle-adjusting neural radiance fields. In *IEEE International Conference on Computer Vision (ICCV)*, 2021. 13

[31] 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 *CVPR*, 2021. 2

[32] Quan Meng, Anpei Chen, Haimin Luo, Minye Wu, Hao Su, Lan Xu, Xuming He, and Jingyi Yu. GNeRF: GAN-based Neural Radiance Field without Posed Camera. In *ICCV*, 2021. 13

[33] 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. In *ECCV*, 2020. 1, 2, 3, 5, 7, 8

[34] Thomas Müller. tiny-cuda-nn, 4 2021. 13

[35] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, July 2022. 3, 4

[36] Michael Niemeyer and Andreas Geiger. Giraffe: Representing scenes as compositional generative neural feature fields. In *CVPR*, 2021. 2

[37] Julian Ost, Fahim Mannan, Nils Thuerey, Julian Knodt, and Felix Heide. Neural scene graphs for dynamic scenes. In *CVPR*, pages 2856–2865, June 2021. 2, 7, 8

[38] Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Steven M. Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. In *ICCV*, 2021. 2

[39] Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Ricardo Martin-Brualla, and Steven M. Seitz. Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields. *ACM Trans. Graph.*, 40(6), dec 2021. 2

[40] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-NeRF: Neural Radiance Fields for Dynamic Scenes. In *CVPR*, 2020. 2

[41] Konstantinos Rematas, Andrew Liu, Pratul P. Srinivasan, Jonathan T. Barron, Andrea Tagliasacchi, Tom Funkhouser, and Vittorio Ferrari. Urban radiance fields. *CVPR*, 2022. 3, 4, 5

[42] Barbara Roessle, Jonathan T. Barron, Ben Mildenhall, Pratul P. Srinivasan, and Matthias Nießner. Dense depth priors for neural radiance fields from sparse input views. In *CVPR*, June 2022. 3

[43] Sara Fridovich-Keil and Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *CVPR*, 2022. 3

[44] Prafull Sharma, Ayush Tewari, Yilun Du, Sergey Zakharov, Rares Andrei Ambrus, Adrien Gaidon, William T. Freeman, Fredo Durand, Joshua B. Tenenbaum, and Vincent Sitzmann. Neural groundplans: Persistent neural scene representations from a single image, 2023. 2, 7

[45] 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. 7, 8

[46] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *CVPR*, 2022. 3

[47] Matthew Tancik, Vincent Casser, Xinchen Yan, Sabeek Pradhan, Ben Mildenhall, Pratul P. Srinivasan, Jonathan T. Barron, and Henrik Kretzschmar. Block-nerf: Scalable large scene neural view synthesis. In *CVPR*, pages 8248–8258, June 2022. 1, 2, 5

[48] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Justin Kerr, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, David McAllister, and Angjoo Kanazawa. Nerfstudio: A modular framework for neural radiance field development. *arXiv preprint arXiv:2302.04264*, 2023. 13

[49] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow (extended abstract). In *IJCAI*, 2021. 6

[50] Edgar Tretschk, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Christoph Lassner, and Christian Theobalt. Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video. In *ICCV*. IEEE, 2021. 2

[51] Vadim Tschernezki, Iro Laina, Diane Larlus, and Andrea Vedaldi. Neural Feature Fusion Fields: 3D distillation of self-supervised 2D image representation. In *Proceedings of the International Conference on 3D Vision (3DV)*, 2022. 2, 5

[52] Haithem Turki, Deva Ramanan, and Mahadev Satyanarayanan. Mega-nerf: Scalable construction of large-scale nerfs for virtual fly-throughs. In *CVPR*, pages 12922–12931, June 2022. 1, 2, 3, 5, 7

[53] Suhani Vora*, Noha Radwan*, Klaus Greff, Henning Meyer, Kyle Genova, Mehdi S. M. Sajjadi, Etienne Pot, Andrea Tagliasacchi, and Daniel Duckworth. Nesf: Neural semantic fields for generalizable semantic segmentation of 3d scenes. *Transactions on Machine Learning Research*, 2022. https://openreview.net/forum?id=ggPhsYCsm9. 2

[54] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004. 6

[55] Zirui Wang, Shangzhe Wu, Weidi Xie, Min Chen, and Victor Adrian Prisacariu. NeRF−−: Neural radiance fields without known camera parameters. *arXiv preprint arXiv:2102.07064*, 2021. 13

[56] Tianhao Wu, Fangcheng Zhong, Andrea Tagliasacchi, Forrester Cole, and Cengiz Oztireli. D^{2}nerf: Self-supervised decoupling of dynamic and static objects from a monocular video. In *Advances in Neural Information Processing Systems*, 2022. 2, 3, 4, 6

[57] Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. Space-time neural irradiance fields for free-viewpoint video. In *CVPR*, pages 9421–9431, 2021. 2

[58] Yuanbo Xiangli, Linning Xu, Xingang Pan, Nanxuan Zhao, Anyi Rao, Christian Theobalt, Bo Dai, and Dahua Lin. Bungeenerf: Progressive neural radiance field for extreme multi-scale scene rendering. In *ECCV*, 2022. 2

[59] Bangbang Yang, Yinda Zhang, Yinghao Xu, Yijin Li, Han Zhou, Hujun Bao, Guofeng Zhang, and Zhaopeng Cui. Learning object-compositional neural radiance field for editable scene rendering. In *ICCV*, October 2021. 2

[60] Gengshan Yang, Minh Vo, Neverova Natalia, Deva Ramanan, Vedaldi Andrea, and Joo Hanbyul. Banmo: Building animatable 3d neural models from many casual videos. In *CVPR*, 2022. 2, 4

[61] Hong-Xing Yu, Leonidas J. Guibas, and Jiajun Wu. Unsupervised discovery of object radiance fields. In *International Conference on Learning Representations*, 2022. 2

[62] Wentao Yuan, Zhaoyang Lv, Tanner Schmidt, and Steven Lovegrove. Star: Self-supervised tracking and reconstruction of rigid objects in motion with neural rendering. In *CVPR*, pages 13144–13152, 2021. 2

[63] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv:2010.07492*, 2020. 2, 3

[64] 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. 6

[65] Shuaifeng Zhi, Tristan Laidlow, Stefan Leutenegger, and Andrew Davison. In-place scene labelling and understanding with implicit scene representation. In *ICCV*, 2021. 2

Supplemental Materials

A. Tracking

We can compute mask and keypoint-level correspondences across frames after detecting instances (Sec. 4.2) by using Best-Buddies similarity [13] on features Φ within or between instances. As a 3D representation, SUDS can track correspondences through 2D occluders. We show an example in Fig. 7.

B. Proposal Sampling

We use a proposal sampling strategy similar to Mip-NeRF 360 [7] that first queries a lightweight occupancy proposal network at uniform intervals along each camera ray and then picks additional samples based on the initial samples. We model our proposal network with separate hash table-backed static and dynamic branches as in Sec. 3.2. We train each branch of the proposal network with histogram loss [7] using the weights of the respective branch of our main model and regularize the resulting sample distances and weights using distortion loss [7]. We find that proposal sampling gives a 2-4x speedup.

C. Smoothness Priors

We use the same spatial and temporal smoothness priors as NSFF [29] to regularize our scene flow. We specifically denote:

$$\mathcal{L}_{sm}(\mathbf{r}) = \sum_{\mathbf{x}} \sum_{t' \in [-1,1]} e^{-2\left\|\mathbf{x}-\mathbf{x}'\right\|_2} \left\| s_{t'}(\mathbf{x}, \mathbf{t}) - s_{t'}(\mathbf{x}', \mathbf{t}) \right\|_1 + \sum_{\mathbf{x}} \left\| s_{\mathbf{t}-1}(\mathbf{x}, \mathbf{t}) + s_{\mathbf{t}+1}(\mathbf{x}, \mathbf{t}) \right\|_1, \tag{29}$$

where $\mathbf{x}$ and $\mathbf{x}'$ indicate neighboring points along the camera ray $\mathbf{r}$.

D. Ablation Details

w/o Depth loss. We remove depth from the reconstruction loss term:

$$\mathcal{L}_{rec} = \mathcal{L}_c + \lambda_f \mathcal{L}_f + \lambda_o \mathcal{L}_o \tag{30}$$

w/o Optical flow loss. We remove optical flow from the reconstruction loss term:

$$\mathcal{L}_{rec} = \mathcal{L}_c + \lambda_f \mathcal{L}_f + \lambda_d \mathcal{L}_d \tag{31}$$

w/o Warping loss. We remove all warping and flow-related loss terms:

$$\mathcal{L} = \underbrace{\left(\mathcal{L}_c + \lambda_f \mathcal{L}_f + \lambda_d \mathcal{L}_d\right)}_{\text{reconstruction losses}} + \underbrace{\left(\lambda_e \mathcal{L}_e + \lambda_d \mathcal{L}_d\right)}_{\text{static-dynamic factorization}} + \lambda_\rho \mathcal{L}_\rho. \tag{32}$$

w/o Appearance embedding. We compute static color without the latent embedding vector $A_{vid}\mathcal{F}(t)$:

$$\mathbf{c}_s(\mathbf{x}, \mathbf{d}) \in \mathbb{R}^3 \tag{33}$$

w/o Occlusion weights. We do not use occlusion weights (24) to downweight the warping loss terms (25, 26):

$$\mathcal{L}^w_c(\mathbf{r}) = \sum_{t' \in [-1,1]} \left\| C(\mathbf{r}) - \hat{C}^w_{t'}(\mathbf{r}) \right\|^2 \tag{34}$$

$$\mathcal{L}^w_f(\mathbf{r}) = \sum_{t' \in [-1,1]} \left\| F(\mathbf{r}) - \hat{F}^w_{t'}(\mathbf{r}) \right\|_1 \tag{35}$$

w/o Separate branches. We generate all model outputs using a single time-dependent branch:

$$\sigma(\mathbf{x}, \mathbf{t}, \mathbf{vid}) \in \mathbb{R} \tag{36}$$

$$\mathbf{c}(\mathbf{x}, \mathbf{t}, \mathbf{vid}, \mathbf{d}) \in \mathbb{R}^3 \tag{37}$$

$$\Phi(\mathbf{x}, \mathbf{t}, \mathbf{vid}) \in \mathbb{R}^C \tag{38}$$

$$s_{t' \in [-1,1]}(\mathbf{x}, \mathbf{t}, \mathbf{vid}) \in \mathbb{R}^3 \tag{39}$$

We accordingly remove factorization-related loss terms:

$$\mathcal{L} = \underbrace{\left(\mathcal{L}_c + \lambda_f \mathcal{L}_f + \lambda_d \mathcal{L}_d + \lambda_o \mathcal{L}_o\right)}_{\text{reconstruction losses}} + \underbrace{\left(\mathcal{L}^w_c + \lambda_f \mathcal{L}^w_f\right)}_{\text{warping losses}} \underbrace{\lambda_{flo}\left(\mathcal{L}_{cyc} + \mathcal{L}_{sm} + \mathcal{L}_{slo}\right)}_{\text{flow losses}} \tag{40}$$

E. Additional Training Details

We divide City-1M into 48 cells using camera-based k-means clustering. Each cell covers 2.9 km^2 and 32k frames across 98 videos on average. We evaluate the effect of geographic coverage and number of frames/videos on cell quality in Table 5. We train with 1 A100 (40 GB) GPU per cell for 2 days (same for each KITTI scene). We can fit all cells on a single A100 at inference time.

F. Assets

City-1M. Our dataset is constructed from street-level videos collected across a vehicle fleet with seven ring cameras that collect 2048x1550 resolution images at 20 Hz with a combined 360° field of view. Both VLP-32C LiDAR sensors are synchronized with the cameras and produce point clouds with 100,000 points at 10 Hz on average. We localize camera poses using a combination of GPS-based and sensor-based methods.

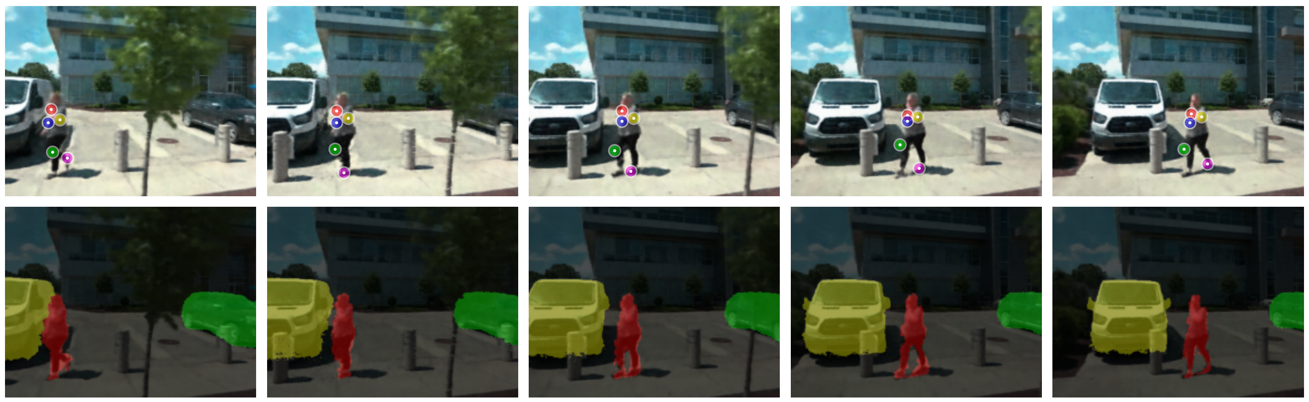


Figure 7. **Tracking.** We track keypoints (**above**) and instance masks (**below**) across several frames. As a 3D representation, SUDS can track correspondences through 2D occluders.

	$\leq$ 15k	15-30k	30-45k	$\geq$ 45k
↑PSNR	22.86	21.99	21.35	20.75
↑SSIM	0.583	0.569	0.557	0.538
↓LPIPS	0.516	0.545	0.564	0.578

Images

	$\leq$ 60	60-90	90-120	$\geq$ 120
↑PSNR	22.47	21.72	21.68	21.11
↑SSIM	0.587	0.556	0.559	0.555
↓LPIPS	0.526	0.557	0.557	0.565

Videos

	$\leq$ 2 km^2	2-3 km^2	3-4 km^2	$\geq$ 4 km^2
↑PSNR	22.73	21.47	21.53	22.18
↑SSIM	0.609	0.556	0.561	0.557
↓LPIPS	0.512	0.564	0.555	0.536

Area

Table 5. **City-1M scaling.** We evaluate the effect of geographic coverage and the number of images and videos on cell quality. Although performance degrades sublinearly across all metrics, image and video counts have the largest impact.

Third-party assets. We primarily base the SUDS implementation on Nerfstudio [48] and tiny-cuda-nn [34] along with various utilities from OpenCV [8], Scikit [9], and Amir et al's feature extractor implementation [5], all of which are freely available for noncommercial use. KITTI [21] is similarly available under an Apache license, whereas VKITTI2 [18] uses the noncommercial CC BY-NC-SA 3.0 license.

G. Limitations

Video boundaries. Although our global representation of static geometry is consistent across all videos used for reconstruction, all dynamic objects are video-specific. Put otherwise, our method does not allow us to extrapolate the movement of objects outside of the boundaries of videos from which they were captured, nor does it provide a straightforward way of rendering dynamic visuals at boundaries where camera rays intersect regions with training data originating from disjoint video sequences.

Camera accuracy. Accurate camera extrinsics and intrinsics are arguably the largest contributors to high NeRF rendering quality. Although multiple efforts [12, 23, 30, 32, 55] attempt to jointly optimize camera parameters during NeRF optimization, we found the results lacking relative to using offline structure-from-motion based approaches as a preprocessing step.

Flow quality. Although our method tolerates some degree of noisiness in the supervisory optical flow input, high-quality flow still has a measurable impact on model performance (and completely incorrect supervision degrades quality). We also assume that flow is linear between observed timestamps to simplify our scene flow representation.

Resources. Modeling city scale requires a large amount of dataset preprocessing, including, but not limited to: extracting DINO features, computing optical flow, deriving normalized coordinate bounds, and storing randomized batches of training data to disk. Collectively, our intermediate representation required more than 20TB of storage even after compression.

Shadows. SUDS attempts to disentangle shadows underneath transient objects. However, if a shadow is present in all observations for a given location (eg: a parking spot that is always occupied, even by different cars), SUDS may attribute the darkness to the static topology, as evidenced in several of our videos, even if the origin of the shadow is correctly assigned to the dynamic branch.

Instance-level tasks. Although we provide initial qualitative results on instance-level tasks as a first step towards true 3D segmentation backed by neural radiance field, SUDSis not competitive with conventional approaches.

H. Societal Impact

As SUDS attempts to model dynamic urban scenes with pedestrians and vehicles, our approach carries surveillance and privacy concerns related to the intentional or inadvertent capture or privacy-sensitive information such as human faces and vehicle license plate numbers. As we distill semantic knowledge into SUDS, we are able to (imperfectly) filter out either entire categories (people) or components (faces) at render time. However this information would still reside in the model itself. This could in turn be mitigated by preprocessing the input data used to train the model.