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):

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

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

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

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

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

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

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

We accordingly remove factorization-related loss terms:

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

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.

	≤ 15k	15-30k	30-45k	≥ 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

	≤ 60	60-90	90-120	≥ 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

	≤ 2 km^2	2-3 km^2	3-4 km^2	≥ 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.