Neural Scene Flow Fields for Space-Time View Synthesis of Dynamic Scenes Supplementary Material

Zhengqi Li[1] Simon Nikalus[2] Noah Snavely[1] Oliver Wang[2]
[1] Cornell Tech [2] Adobe Research

1. Scene Flow Regularization Details

Recall that $\mathcal{L}_{\text{reg}}$ is used as a regularization loss for the predicted scene flow fields, consisting of three terms with equal weights: $\mathcal{L}_{\text{reg}} = \mathcal{L}_{\text{sp}} + \mathcal{L}_{\text{temp}} + \mathcal{L}_{\text{min}}$, corresponding to spatial smoothness, temporal smoothness, and minimal scene flow.

Scene flow spatial smoothness [5] minimizes the weighted ℓ_1 difference between scenes flows sampled at neighboring 3D position along each ray $\mathbf{r}_i$. In particular, the spatial smoothness term is written as:

$$\mathcal{L}_{\text{sp}} = \sum_{\mathbf{x}_i} \sum_{\mathbf{y}_i \in \mathcal{N}(\mathbf{x}_i)} \sum_{j \in \{i \pm 1\}} w^{\text{dist}}(\mathbf{x}_i, \mathbf{y}_i) ||\mathbf{f}_{i \to j}(\mathbf{x}_i) - \mathbf{f}_{i \to j}(\mathbf{y}_i)||_1, \tag{1}$$

where $\mathcal{N}(\mathbf{x}_i)$ is the neighboring points of $\mathbf{x}_i$ sampled along the ray $\mathbf{r}_i$, and weights are computed by the Euclidean distance between the two points: $w^{\text{dist}}(\mathbf{x}, \mathbf{y}) = \exp(-2||\mathbf{x} - \mathbf{y}||_2)$.

Scene flow temporal smoothness, inspired by Vo *et al.* [9], encourages 3D point trajectories to be piece-wise linear with least kinetic energy prior. This is equivalent to minimizing sum of forward scene flow and backward scene flow from each sampled 3D point along the ray:

$$\mathcal{L}_{\text{temp}} = \frac{1}{2} \sum_{\mathbf{x}_i} ||\mathbf{f}_{i \to i+1}(\mathbf{x}_i) + \mathbf{f}_{i \to i-1}(\mathbf{x}_i)||_2^2 \tag{2}$$

Lastly, we encourage scene flow to be minimal in most of 3D space [8] by applying a l_1 regularization term to each predicted scene flow:

$$\mathcal{L}_{\text{min}} = \sum_{\mathbf{x}_i} \sum_{j \in \{i \pm 1\}} ||\mathbf{f}_{i \to j}(\mathbf{x}_i)||_1 \tag{3}$$

2. Data Driven Prior Details

Geometric consistency prior. Recall the geometric consistency prior minimizes the reprojection error of scene flow displaced 3D points w.r.t. the derived 2D optical flow. Suppose $\mathbf{p}_i$ is a 2D pixel position at time i. The corresponding 2D pixel location in the neighboring frame at time j displaced through 2D optical flow $\mathbf{u}_{i \to j}$ can be computed as $\mathbf{p}_{i \to j} = \mathbf{p}_i + \mathbf{u}_{i \to j}$.

To estimate the expected 2D point location $\hat{\mathbf{p}}_{i \to j}$ at time j displaced by predicted scene flow fields, we first compute the expected scene flow $\hat{\mathbf{F}}_{i \to j}(\mathbf{r}_i)$ and the expected 3D point location $\hat{\mathbf{X}}_i(\mathbf{r}_i)$ of the ray $\mathbf{r}_i$ through volume rendering:

$$\hat{\mathbf{F}}_{i \to j}(\mathbf{r}_i) = \int_{t_n}^{t_f} T_i(t)\, \sigma_i(\mathbf{r}_i(t))\, \mathbf{f}_{i \to j}(\mathbf{r}_i(t)) dt, \tag{4}$$

$$\hat{\mathbf{X}}_i(\mathbf{r}_i) = \int_{t_n}^{t_f} T_i(t)\, \sigma_i(\mathbf{r}_i(t))\, \mathbf{x}_i(\mathbf{r}_i(t)) dt. \tag{5}$$

$\hat{\mathbf{p}}_{i \to j}$ is then computed by performing perspective projection of the expected 3D point location displaced by the scene flow (i.e. $\hat{\mathbf{X}}_i(\mathbf{r}_i) + \hat{\mathbf{F}}_{i \to j}(\mathbf{r}_i)$) into the viewpoint corresponding to the frame at time j:

$$\hat{\mathbf{p}}_{i \to j}(\mathbf{r}_i) = \pi(K(\mathbf{R}^j(\hat{\mathbf{X}}_i(\mathbf{r}_i) + \hat{\mathbf{F}}_{i \to j}(\mathbf{r}_i)) + \mathbf{t}^j)), \tag{6}$$

where $(\mathbf{R}^j, \mathbf{t}^j) \in SE(3)$ are rigid body transformations that transform 3D points from the world coordinate system to the coordinate system of frame at time j. K is a camera intrinsic matrix shared among all the frames, and π is perspective division operation. The geometric consistency can be applied by comparing the l_1 difference between $\hat{\mathbf{p}}_{i\to j}$ and $\mathbf{p}_{i\to j}$:

$$\mathcal{L}_{\text{geo}} = \sum_{\mathbf{r}_i} \sum_{j\in\mathcal{N}(i)} ||\hat{\mathbf{p}}_{i\to j}(\mathbf{r}_i) - \mathbf{p}_{i\to j}(\mathbf{r}_i))||_1. \tag{7}$$

Single-view depth prior. The single view depth prior encourages the expected termination depth $\hat{Z}_i$ computed along each ray to be close to the depth Z_i predicted from a pre-trained single-view depth network [7]. As single-view depth predictions are defined up to an unknown scale and shift, we utilize a robust scale-shift invariant loss [7]:

$$\mathcal{L}_z = \sum_{\mathbf{r}_i} ||\hat{Z}_i^*(\mathbf{r}_i) - Z_i^*(\mathbf{r}_i)||_1 \tag{8}$$

We normalize the depths to have zero translation and unit scale using robust estimator:

$$\begin{gathered} Z^*(\mathbf{r}_i) = \frac{Z(\mathbf{r}_i) - \text{shift}(Z)}{\text{scale}(Z)}, \\ \text{where shift}(Z) = \text{median}(Z),\ \text{scale}(Z) = \text{mean}(|Z - \text{shift}(Z)|). \end{gathered} \tag{9}$$

Due to computational limits, we are not able normalize the entire depth image during training, so we normalize the depth value using the shift and scale estimate from current sampled points in each training iteration. Furthermore, since we reconstruct the entire scene in normalized device coordinate (NDC) space, and the MiDAS model [7] predicts disparity in Euclidean space with an unknown scale and shift, we can use the NDC ray space derivation from NeRF [4] to derive that the depth in NDC space is equal to negative disparity in Euclidean space up to scale and shift, so our single-view term is implemented as:

$$\mathcal{L}_z = \sum_{\mathbf{r}_i} ||\hat{Z}_i^*(\mathbf{r}_i) + \frac{1}{Z_i}^*(\mathbf{r}_i)||_1 \tag{10}$$

3. Space-Time Interpolation Visualization

In Sec 3.4 of our main paper, we propose a splatting-based plane-sweep volume tracing approach to perform space-time interpolation to synthesize novel views in at novel view points, and in between input time indices. We show a visual illustration of this in Fig. 1. In practice, we use the CUDA implementation of average splatting from Niklaus *et al*. [6] to efficiently perform forward splatting of the 3D points through scene flow fields.

4. Volume Rendering Equation Approximation

Recall in Sec 3.3, the combined rendering equation is written as:

$$\hat{\mathbf{C}}_i^{\text{cb}}(\mathbf{r}_i) = \int_{t_n}^{t_f} T_i^{\text{cb}}(t)\, \sigma_i^{\text{cb}}(t)\, \mathbf{c}_i^{\text{cb}}(t) dt,, \tag{11}$$

where $\sigma_i^{\text{cb}}(t)\, \mathbf{c}_i^{\text{cb}}(t)$ is a linear combination of static scene components $\mathbf{c}(\mathbf{r}_i(t), \mathbf{d}_i)\, \sigma(\mathbf{r}_i(t))$ and dynamic scene components $\mathbf{c}_i(\mathbf{r}_i(t), \mathbf{d}_i)\, \sigma_i(\mathbf{r}_i(t))$, weighted by $v(\mathbf{r}_i(t))$:

$$\sigma_i^{\text{cb}}(t)\, \mathbf{c}_i^{\text{cb}}(t) = v(t)\, \mathbf{c}(t)\, \sigma(t) + (1\text{-}v(t))\, \mathbf{c}_i(t)\, \sigma_i(t). \tag{12}$$

We approximate this combined rendering equation using the same quadrature approximations technique described in prior work [3, 4]. Suppose $\{t^l\}_{l=1}^L$ are the points sampled within the near and far bounds and we denote the distance between every sampled points $\delta^l = t^{l+1} - t^l$, the discrete approximation of Eq. 11 is then written as:

$$\begin{aligned} \hat{\mathbf{C}}_i^{\text{cb}}(\mathbf{r}_i) &= \sum_{l=1}^{L} T_i^{\text{cb}}(t^l) \Big(v(t^l)\, \alpha(\sigma(t^l)\delta^l)\, \mathbf{c}(t^l) + (1 - v(t^l))\, \alpha(\sigma_i(t^l)\delta^l)\, \mathbf{c}_i(t^l) \Big), \\ \text{where } T_i^{\text{cb}}(t^l) &= \exp\Big(- \sum_{l'=1}^{l-1} \big(v(t^{l'})\, \sigma(t^{l'}) + (1 - v(t^{l'}))\, \sigma_i(t^{l'}) \big) \delta_{l'} \Big), \\ \text{and } \alpha(x) &= 1 - \exp(-x) \end{aligned} \tag{13}$$

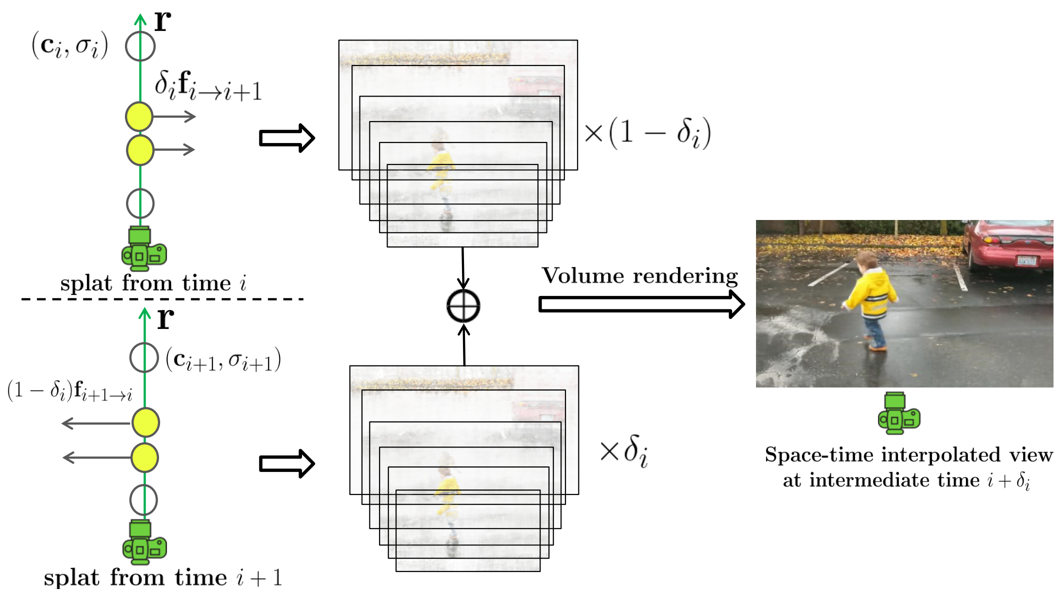
$\mathbf{r}$
$(\mathbf{c}_i, \sigma_i)$
$\delta_i \mathbf{f}_{i \to i+1}$
$\times(1-\delta_i)$
splat from time i
Volume rendering
$\mathbf{r}$
$(\mathbf{c}_{i+1}, \sigma_{i+1})$
$(1-\delta_i)\mathbf{f}_{i+1 \to i}$
$\times \delta_i$
Space-time interpolated view at intermediate time $i+\delta_i$
splat from time $i+1$

Figure 1: **Space-time view synthesis.** We propose a 3D splatting-based approach to perform space-time interpolation at specified novel viewpoint (shown as green camera) at an intermediate time $i + \delta_i$. Specifically, we sweep over every ray $\mathbf{r}$ emitted from the specified novel viewpoint from front to back. At each sampled step t along the ray, we query the color and density information $(\mathbf{c}, \alpha)$, as well as the scene flows at both times i and $i + 1$. We then displace the 3D points by the scaled scene flow $\delta_i \mathbf{f}_{i \to i+1}$, $(1 - \delta_i)\mathbf{f}_{i \to i-1}$ respectively (left). The 3D displaced points and their associated attributes are then splatted from time i and $i + 1$ onto a $(\mathbf{c}, \alpha)$ accumulation buffer at the specified novel viewpoint. The splats the accumulation buffer are blended with linear weights $1 - \delta_i, \delta_i$ (middle) followed by standard volume rendering to obtain final rendered image (right).

5. Network Architecture

Our network architecture is a variant of the original NeRF, which adopts MLPs as a backbone. Our full model consists of two separate MLPs, corresponding to a static (time-independent) scene representation (Fig. 2) and a dynamic (time-dependent) scene representation (Fig. 3).

6. Implementation Details

Initialization. We denote the initialization stage as the first $1000N$ iterations during training, where N is the number training views. To warm up the optimization, during the initialization stage, we only compute the temporal losses only in temporal window of size 3, i.e. $j \in \{i, i \pm 1\}$, and switch to a temporal window of size 5, i.e. $j \in \mathcal{N}(i) = \{i, i \pm 1, i \pm 2\}$ after the initialization stage.

Additionally, as both of the data-driven priors are noisy (in that they rely on inaccurate or incorrect predictions), we use these for *initialization only*, and linearly decay the weight of $\mathcal{L}_{\text{data}}$ to zero during training for a fixed number of iterations. In particular, we linearly decrease the weight by factor of 10 every $1000N$ iterations.

Hard mining sampling. Optionally, to sufficiently initialize the depth and scenes flows of small, fast moving objects such as the limbs of a person, we precompute a coarse binary motion segmentation mask from each frame, and sample an additional 512 points from the motion mask regions during the initialization stage. These additionally sampled points are added to the loss used in our dynamic (time-variant) scene representation.

Similar to prior work [10], we compute the above coarse binary motion segmentation using a combination of physical and semantic estimates of rigidity. In particular, the physical mask is wherever the distance between the optical flow [2] at each pixel and its corresponding epipolar line from the neighboring frame at time $j \in \mathcal{N}(i)$ is greater than 1 pixel, and the semantic mask is computed using an off-the-shelf instance segmentation network [1] to label all pixels corresponding to possible moving objects such as people and animals. Finally, We union the two masks followed by morphological dilation to obtain the final binary mask. Note this coarse motion segmentation is mainly used to slightly increase the number of samples for the data-driven priors during initialization and does not need to be accurate.

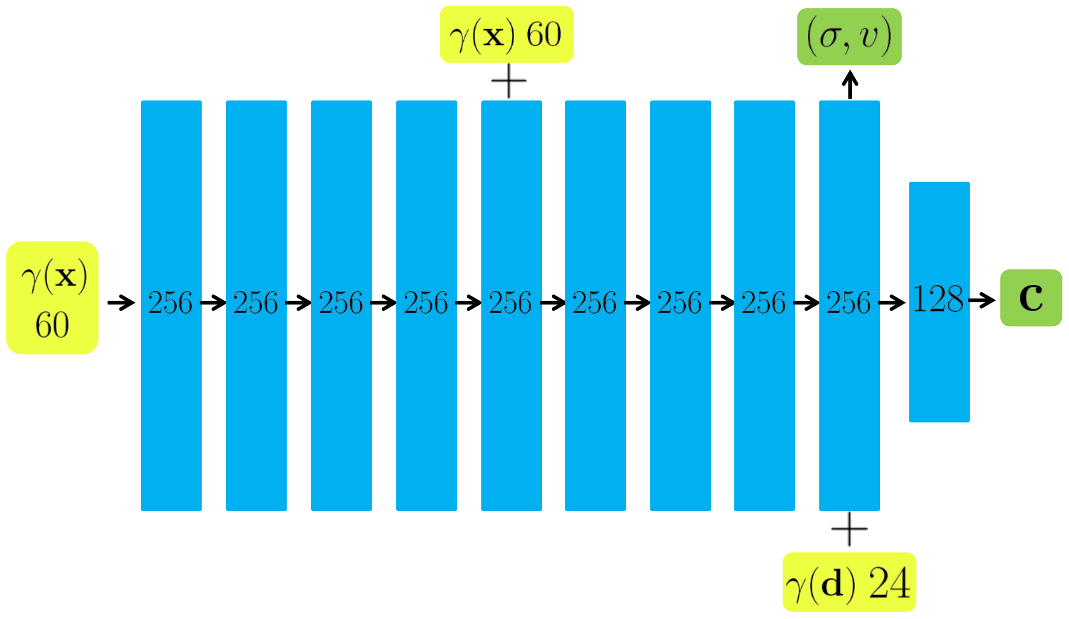

Figure 2: **Network architecture of static (time-invariant) scene representation.** Modified from the original NeRF architecture diagram. We predict an extra blending weight field v from intermediate features along with opacity σ.

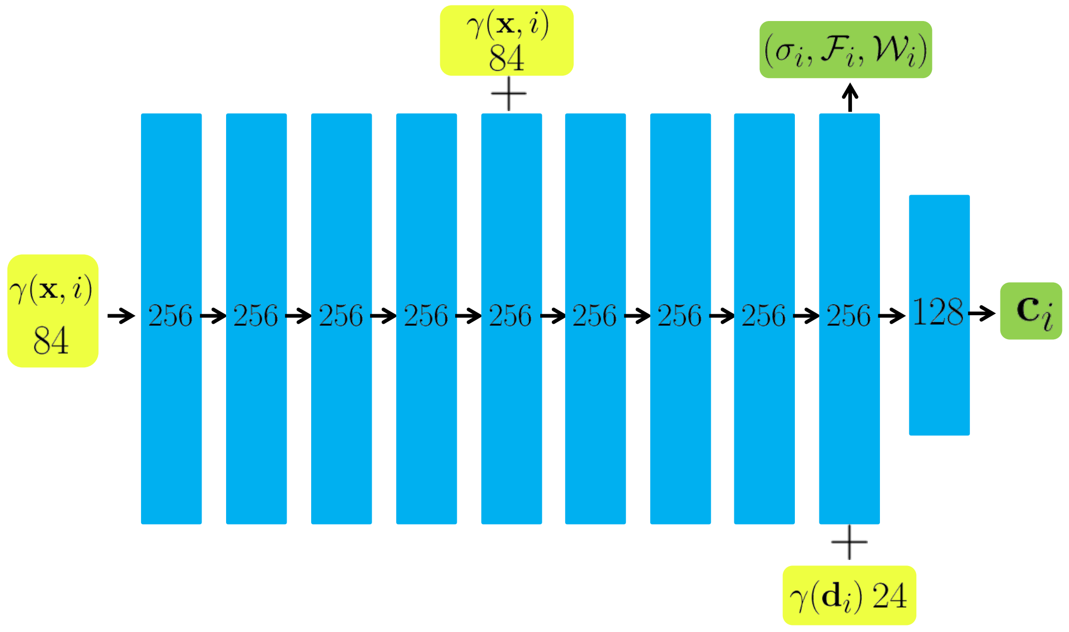
$\gamma(\mathbf{x}, i)$ 84
$(\sigma_i, \mathcal{F}_i, \mathcal{W}_i)$
$\gamma(\mathbf{x}, i)$ 84
256
128
$\mathbf{c}_i$
$\gamma(\mathbf{d}_i)$ 24

Figure 3: **Network Architecture of dynamic (time-variant) scene representation.** Modified from the original NeRF architecture diagram. We encode and input time indices i into the MLP and predict time-dependent scene flow fields $\mathcal{F}_i$ and disocculusion weight fields $\mathcal{W}_i$ from the intermediate features along with opacity σ_i.

Hyperparameters and evaluation details. We implement our framework using PyTorch. We empirically set $\beta_{\text{cyc}} = 1, \beta_{\text{reg}} = 0.1$, $\beta_{\text{data}} \in \{0.2, 0.4\}$ in our experiments. We also perform forward-backward checks after we use pretrained network to estimate optical flow between adjacent frames. When we evaluate all the baselines, we resize their rendered images to be same as our rendered images before performing evaluation.

In order to accurately determine which region is moving, we compute ground truth dynamic masks for numerical evaluation

(Dynamic Only described in the main manuscript) from the multi-view videos. In particular, we compute optical flow between reference time instance and its neighboring time instances at the same viewpoint, and segment out the dynamic region where the flow magnitude is larger than one pixel.

References

[1] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proc. Int. Conf. on Computer Vision (ICCV)*, pages 2961–2969, 2017. 3

[2] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pages 2462–2470, 2017. 3

[3] Ricardo Martin-Brualla, N. Radwan, Mehdi S. M. Sajjadi, J. Barron, A. Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. *ArXiv*, abs/2008.02268, 2020. 2

[4] 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. *Proc. European Conf. on Computer Vision (ECCV)*, 2020. 2

[5] Richard A Newcombe, Dieter Fox, and Steven M Seitz. DynamicFusion: Reconstruction and tracking of non-rigid scenes in real-time. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, 2015. 1

[6] Simon Niklaus and Feng Liu. Softmax splatting for video frame interpolation. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pages 5436–5445, 2020. 2

[7] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. 2020. 2

[8] Jack Valmadre and S. Lucey. General trajectory prior for non-rigid reconstruction. pages 1394–1401, 2012. 1

[9] Minh Vo, S. Narasimhan, and Yaser Sheikh. Spatiotemporal bundle adjustment for dynamic 3d reconstruction. *Proc. Computer Vision and Pattern Recognition (CVPR)*, pages 1710–1718, 2016. 1

[10] Jonas Wulff, Laura Sevilla-Lara, and Michael J Black. Optical flow in mostly rigid scenes. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pages 4671–4680, 2017. 3