Robust Dynamic Radiance Fields

Yu-Lun Liu[2*] Chen Gao[1] Andreas Meuleman[3*] Hung-Yu Tseng[1] Ayush Saraf[1]
Changil Kim[1] Yung-Yu Chuang[2] Johannes Kopf[1] Jia-Bin Huang[1,4]
[1]Meta [2]National Taiwan University [3]KAIST [4]University of Maryland, College Park

CVPR 2023

Presenter: Hao Wang

Advisor: Prof. Chia-Wen Lin

Outline

- Introduction
- Related Work
- Framework
- Method
- Experiment
- Conclusion

Introduction

- Present a space-time synthesis algorithm from a dynamic monocular video that does not require known camera poses and camera intrinsics as input.

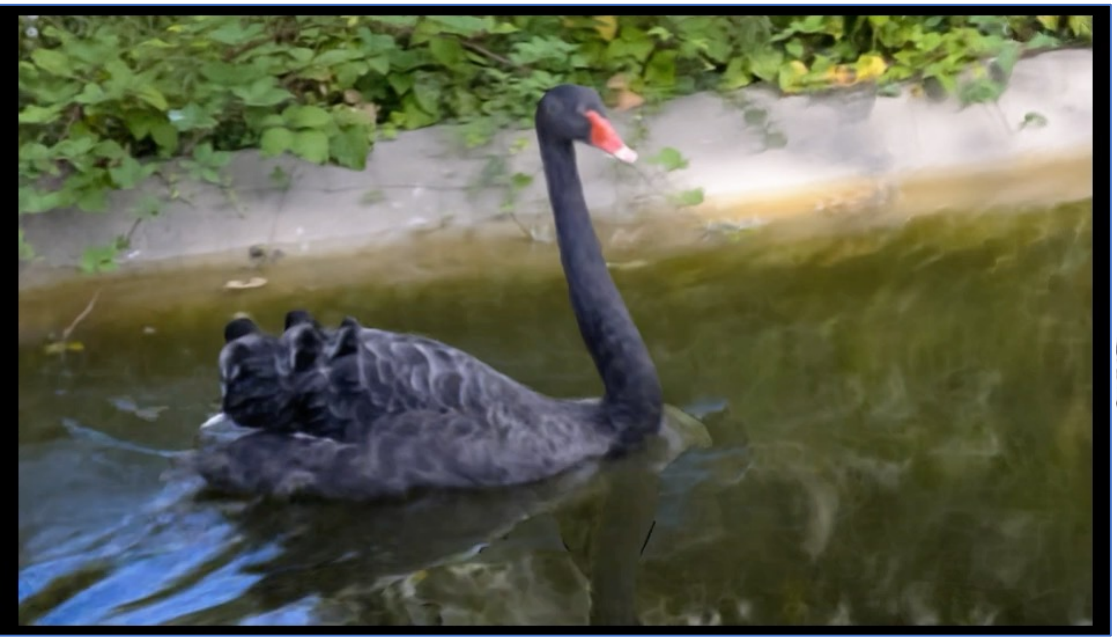
- Our proposed careful architecture designs and auxiliary losses improve the robustness of camera pose estimation and dynamic radiance field reconstruction.

- Quantitative and qualitative evaluations demonstrate the robustness of our method over other state-of-the-art methods on several challenging datasets that typical SfM systems fail to estimate camera poses.

Table 1. **Categorization of view synthesis methods.**

	Known camera poses	*Unknown* camera poses
Static scene	NeRF [44], SVS [59], NeRF++ [82], Mip-NeRF [4], Mip-NeRF 360 [5], DirectVoxGO [68], Plenoxels [23], Instant-ngp [45], TensoRF [12]	NeRF - - [73], BARF [40], SC-NeRF [31], NeRF-SLAM [60]
Dynamic scene	NV [43], D-NeRF [56], NR-NeRF [71], NSFF [39], DynamicNeRF [24], Nerfies [52], HyperNeRF [53], TiNeuVox [20], T-NeRF [25]	**Ours**

Introduction



Outline

- Introduction
- Related Work
- Framework
- Method
- Experiment
- Conclusion

Related Work

- D-NeRF: Neural Radiance Fields for Dynamic Scenes
  - CVPR 2021
- Dynamic View Synthesis from Dynamic Monocular Video
  - ICCV 2021

Related Work – D-NeRF

D-NeRF: Neural Radiance Fields for Dynamic Scenes

Albert Pumarola[1] Enric Corona[1] Gerard Pons-Moll[2,3] Francesc Moreno-Noguer[1]
[1]Institut de Robòtica i Informàtica Industrial, CSIC-UPC
[2]University of Tübingen
[3]Max Planck Institute for Informatics

CVPR 2021

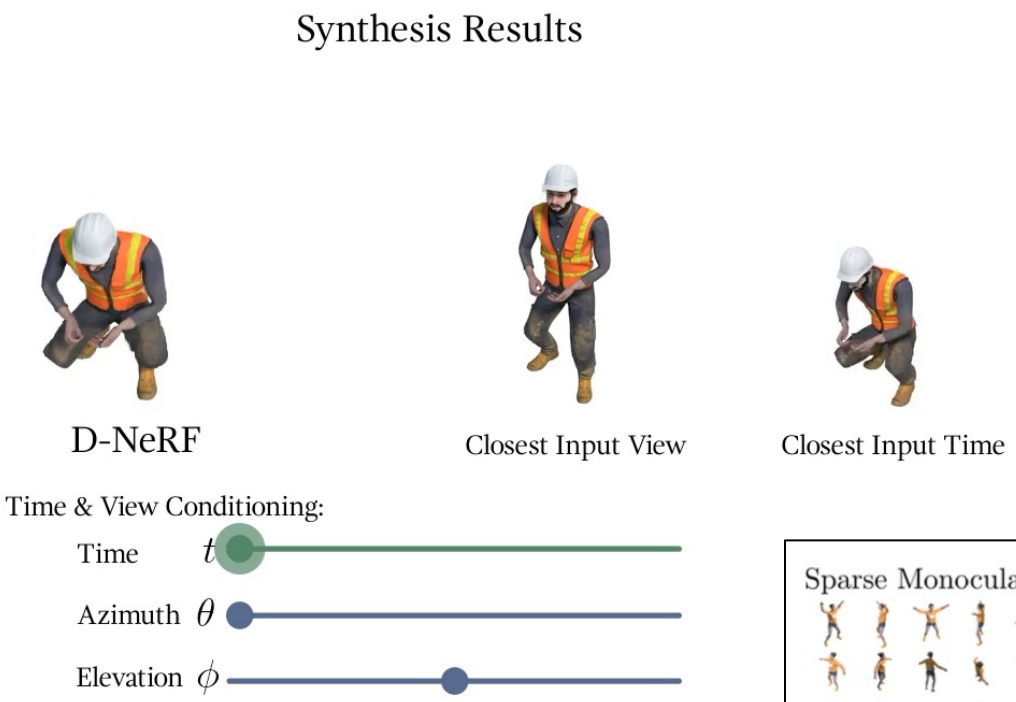
Introduction

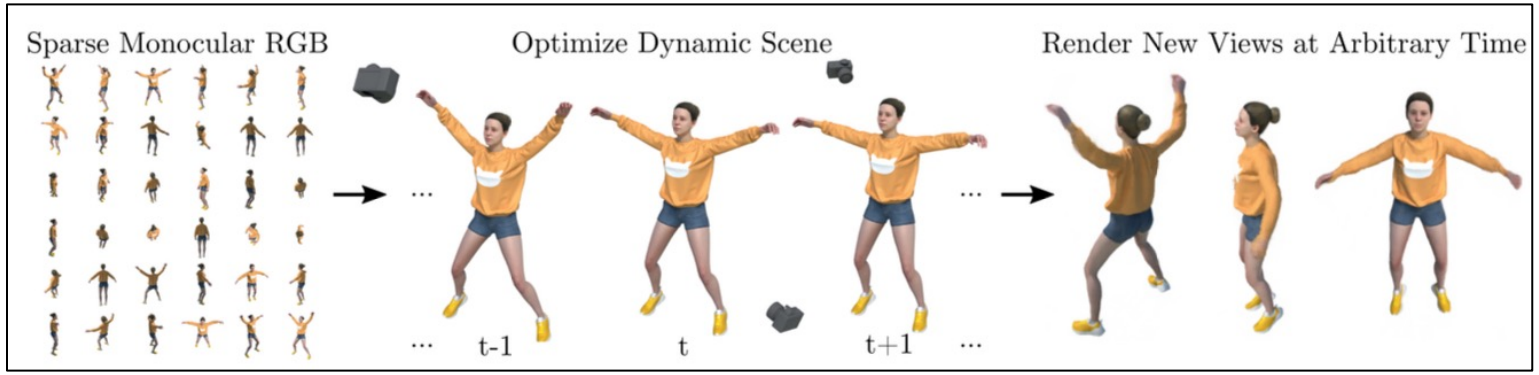
- The first end-to-end neural rendering system that is applicable to dynamic scenes.

- Core idea to build our method is to decompose learning in two modules. Both mappings are learned with deep fully connected networks without convolutional layers.

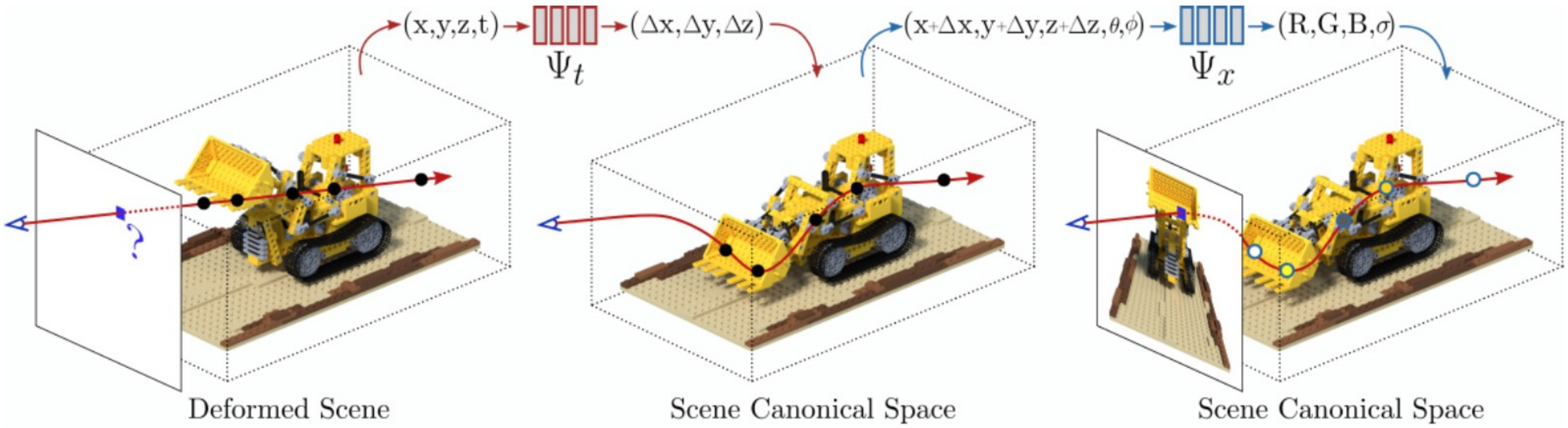
- Allows to synthesize novel images, providing control in the continuum (θ, φ, t) of the camera views and time component.

Introduction



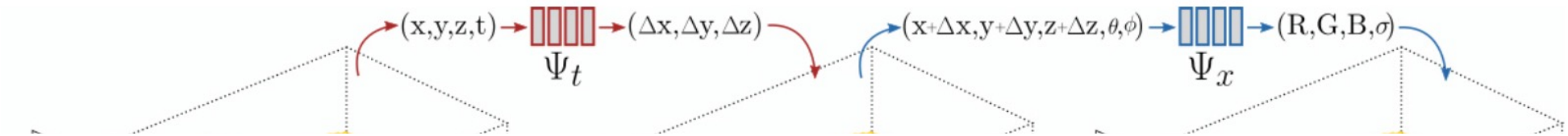


Framework



- The proposed architecture consists of two main blocks
  - **Deformation network Ψt** predicts a deformation field defining the transformation between the scene at time t and the scene in its canonical configuration.
  - **Canonical network Ψx** regressing volume density and view-dependent RGB color from every camera ray

Method



- Deformation Network

$$\Psi_t(\mathbf{x}, t) = \begin{cases} \Delta\mathbf{x}, & \text{if } t \neq 0 \\ 0, & \text{if } t = 0 \end{cases} \qquad (1)$$

  - Optimized to estimate the deformation field between the scene at a specific time and the scene in canonical space

- Canonical Network

  - The canonical network Ψx is trained so as to encode volumetric density and color of the scene in canonical configuration.
  - First, encode x into a 256-dimensional feature vector. This feature vector is then concatenated with the camera viewing direction d

Volume Rendering

$$C(p,t) = \int_{h_n}^{h_f} \mathcal{T}(h,t)\sigma(\mathbf{p}(h,t))\mathbf{c}(\mathbf{p}(h,t),\mathbf{d})dh, \quad (2)$$

$$\text{where} \quad \mathbf{p}(h,t) = \mathbf{x}(h) + \Psi_t(\mathbf{x}(h),t), \quad (3)$$

$$[\mathbf{c}(\mathbf{p}(h,t),\mathbf{d}), \sigma(\mathbf{p}(h,t))] = \Psi_x(\mathbf{p}(h,t),\mathbf{d}), \quad (4)$$

$$\text{and} \quad \mathcal{T}(h,t) = \exp\left(-\int_{h_n}^{h} \sigma(\mathbf{p}(s,t))ds\right). \quad (5)$$

- Approximated via numerical quadrature
  - To select a random set of quadrature points $\{h_n\}_{n=1}^{N} \in [h_n, h_f]$ a stratified sampling strategy

$$C'(p,t) = \sum_{n=1}^{N} \mathcal{T}'(h_n,t)\alpha(h_n,t,\delta_n)\mathbf{c}(\mathbf{p}(h_n,t),\mathbf{d}), \quad (6)$$

$$\text{where} \quad \alpha(h,t,\delta) = 1 - \exp(-\sigma(\mathbf{p}(h,t))\delta), \quad (7)$$

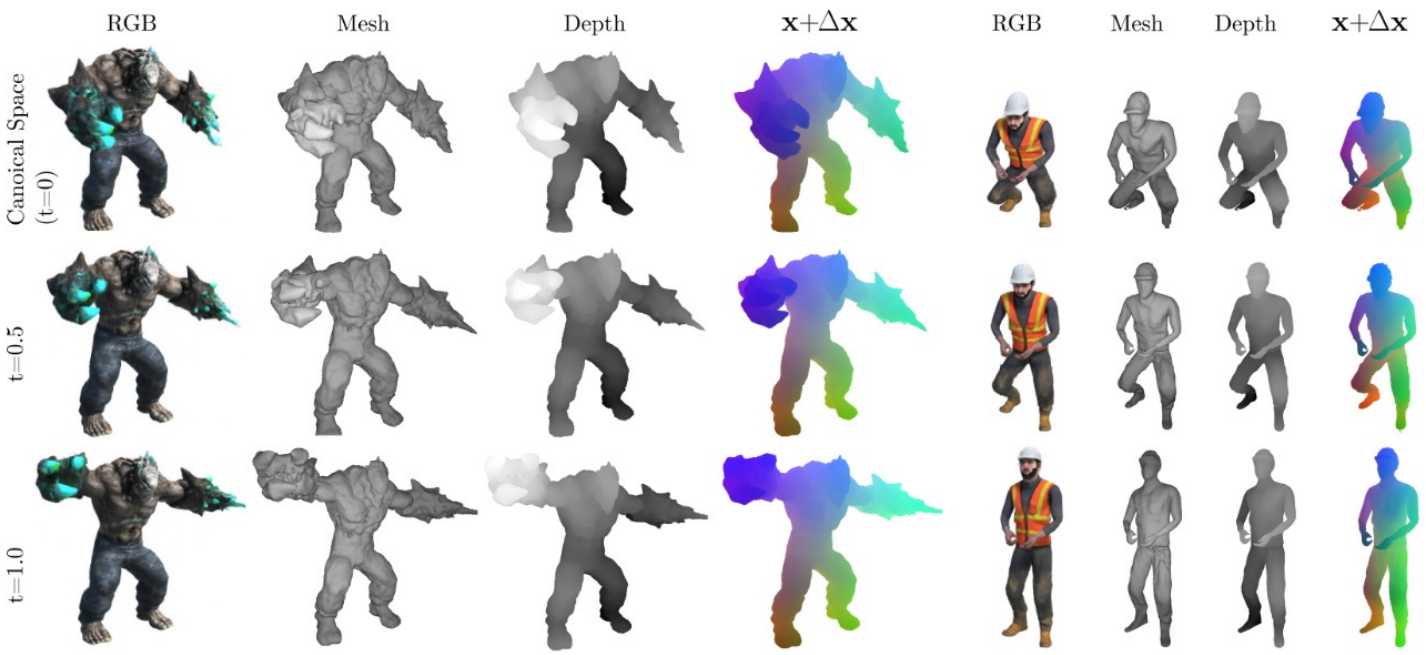
$$\text{and} \quad \mathcal{T}'(h_n,t) = \exp\left(-\sum_{m=1}^{n-1} \sigma(\mathbf{p}(h_m,t))\delta_m\right), \quad (8)$$

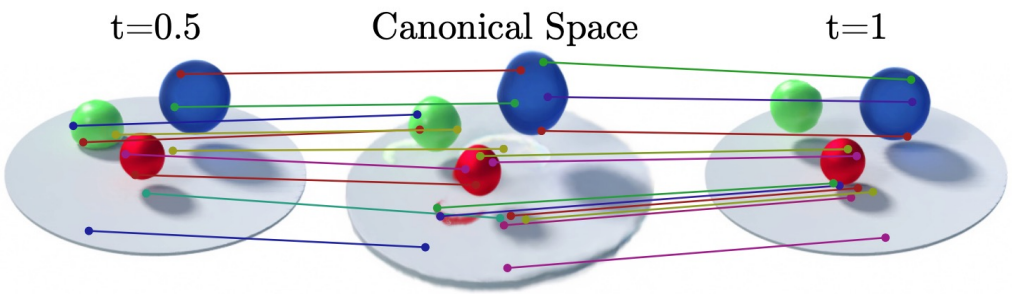
Learning Loss

$$\mathcal{L} = \frac{1}{N_s} \sum_{i=1}^{N_s} \left\| \hat{C}(p,t) - C'(p,t) \right\|_2^2 \tag{9}$$

- Trained with 400×400 images during 800k iterations
- Batch size of Ns = 4096 rays, each sampled 64 times along the ray
- Both network consists on simple 8-layers MLPs with ReLU activations

Experiment

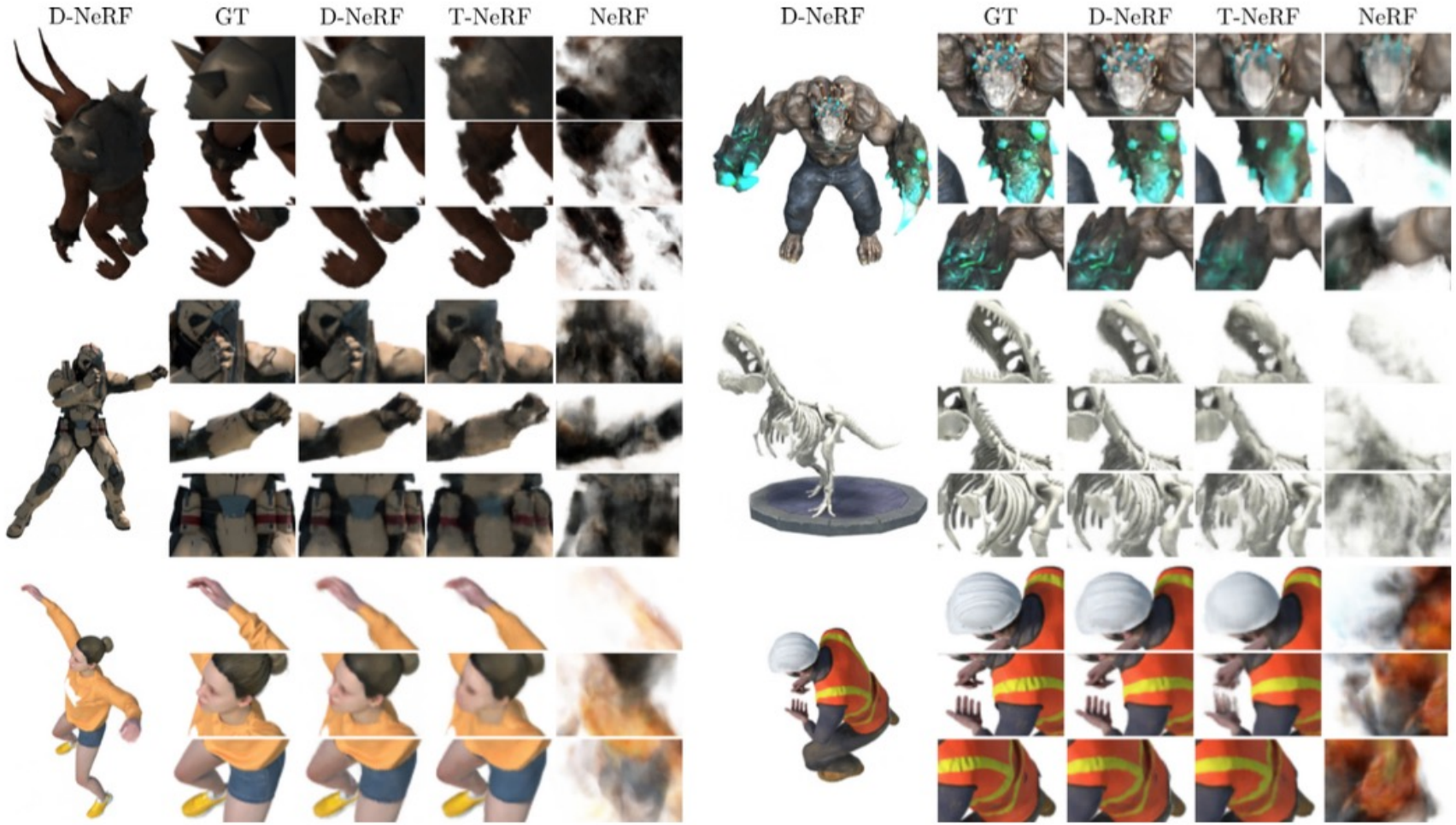




- The same colors on corresponding points indicate the correctness of such mapping

- Different materials (plastic –green–, translucent glass –blue– and metal –red–)
- Able to synthesize the shading effects

Experiment



- T-NeRF scene is represented by a 6D input (x, y, z, θ, φ, t)
- D-NeRF, retains high details of the original image in the novel views

	Hell Warrior				Mutant				Hook				Bouncing Balls			
Method	MSE↓	PSNR↑	SSIM↑	LPIPS↓	MSE↓	PSNR↑	SSIM↑	LPIPS↓	MSE↓	PSNR↑	SSIM↑	LPIPS↓	MSE↓	PSNR↑	SSIM↑	LPIPS↓
NeRF	44e-3	13.52	0.81	0.25	9e-4	20.31	0.91	0.09	21e-3	16.65	0.84	0.19	94e-4	20.26	0.91	0.2
T-NeRF	47e-4	23.19	0.93	0.08	8e-4	30.56	0.96	0.04	18e-4	27.21	0.94	**0.06**	16e-5	37.81	0.98	0.12
D-NeRF	**31e-4**	**25.02**	**0.95**	**0.06**	**7e-4**	**31.29**	**0.97**	**0.02**	**11e-4**	**29.25**	**0.96**	0.11	**12e-5**	**38.93**	**0.98**	**0.1**
	Lego				T-Rex				Stand Up				Jumping Jacks			
Method	MSE↓	PSNR↑	SSIM↑	LPIPS↓	MSE↓	PSNR↑	SSIM↑	LPIPS↓	MSE↓	PSNR↑	SSIM↑	LPIPS↓	MSE↓	PSNR↑	SSIM↑	LPIPS↓
NeRF	9e-3	20.30	0.79	0.23	3e-3	24.49	0.93	0.13	1e-2	18.19	0.89	0.14	1e-2	18.28	0.88	0.23
T-NeRF	**3e-4**	**23.82**	**0.90**	**0.15**	9e-3	30.19	0.96	0.13	7e-4	31.24	0.97	0.02	6e-4	32.01	0.97	0.03
D-NeRF	6e-4	21.64	0.83	0.16	**6e-3**	**31.75**	**0.97**	**0.03**	**5e-4**	**32.79**	**0.98**	**0.02**	**5e-4**	**32.80**	**0.98**	**0.03**

Table 1: **Quantitative Comparison.** We report MSE/LPIPS (lower is better) and PSNR/SSIM (higher is better).

Related Work – Dynamic-NeRF

Dynamic View Synthesis from Dynamic Monocular Video

Chen Gao
Virginia Tech

Ayush Saraf
Facebook

Johannes Kopf
Facebook

Jia-Bin Huang
Virginia Tech

ICCV 2021

Introduction

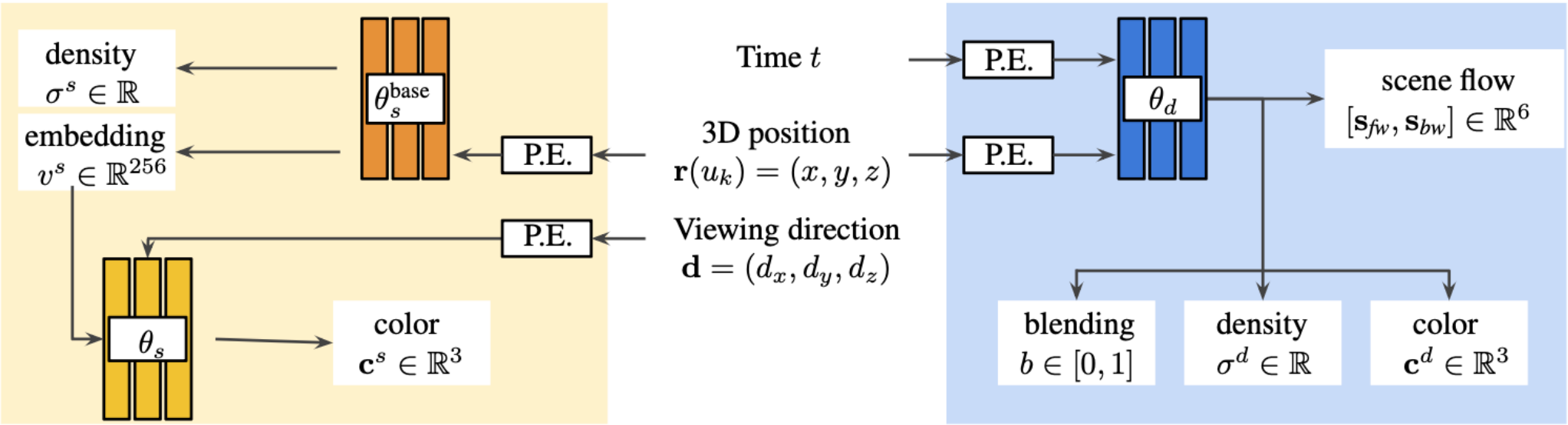
- Present an algorithm for generating novel views at arbitrary viewpoints and any input time step given a monocular video of a dynamic scene.

- Jointly train a time-invariant static NeRF and a time-varying dynamic NeRF, and learn how to blend the results in an unsupervised manner.

- To resolve the ambiguity, we introduce multi-view constraints and regularization losses to encourage a more physically plausible solution.

Introduction

Framework



(a) Static NeRF (Section 3.2) (b) Dynamic NeRF (Section 3.3)

- propose to use two different models to scene components
  - **(a) Static NeRF:** reconstruct the background's structure and appearance without moving objects
  - **(b) Dynamic NeRF:** model a dynamic scene from a single video, leverage the multi-view constraints

Static NeRF

$$\mathbf{r}(u_k) = \mathbf{o} + u_k \mathbf{d}$$

$$(\sigma^s, \mathbf{c}^s) = \mathrm{MLP}_\theta\left(\mathbf{r}(u_k)\right), \tag{1}$$

- using numerical quadrature for approximating the volume rendering interval

$$\mathbf{C}^s(\mathbf{r}) = \sum_{k=1}^{K} T^s(u_k)\, \alpha^s(\sigma^s(u_k)\, \delta_k)\, \mathbf{c}^s(u_k), \tag{2}$$

$$T^s(u_k) = \exp\left(-\sum_{k'=1}^{k-1} \sigma^s(u_k)\, \delta_k\right), \tag{3}$$

$$\mathcal{L}_{static} = \sum_{ij} \|(\mathbf{C}^s(\mathbf{r}_{ij}) - \mathbf{C}^{gt}(\mathbf{r}_{ij})) \cdot (1 - \mathbf{M}(\mathbf{r}_{ij}))\|_2^2 \tag{4}$$

Dynamic NeRF

- Train an MLP that takes a 3D position and time (x, y, z, t) as input to model the volume density and color of the dynamic objects at each time instance
- Lacks multi-view constraints. We predict the forward and backward scene flow and use them to create a **warped radiance field**.

$$\left(\mathbf{s}_{fw}, \mathbf{s}_{bw}, \sigma_t^d, \mathbf{c}_t^d, b\right) = \mathrm{MLP}_{\theta_d}\left(\mathbf{r}(u_k), t\right) \tag{5}$$

$$\left(\sigma_{t+1}^d,\ \mathbf{c}_{t+1}^d\right) = \mathrm{MLP}_{\theta_d}(\mathbf{r}(u_k) + \mathbf{s}_{fw},\ t+1) \tag{6}$$

$$\left(\sigma_{t-1}^d,\ \mathbf{c}_{t-1}^d\right) = \mathrm{MLP}_{\theta_d}(\mathbf{r}(u_k) + \mathbf{s}_{bw},\ t-1) \tag{7}$$

Dynamic NeRF

- **Dynamic rendering photometric loss**
  - **warped radiance field** by resampling the radiance fields implicitly modeled at time $t+1$ and $t-1$

$$\left(\sigma_{t+1}^{d},\, \mathbf{c}_{t+1}^{d}\right) = \mathrm{MLP}_{\theta_d}\left(\mathbf{r}(u_k) + \mathbf{s}_{fw},\, t+1\right) \tag{6}$$

$$\left(\sigma_{t-1}^{d},\, \mathbf{c}_{t-1}^{d}\right) = \mathrm{MLP}_{\theta_d}\left(\mathbf{r}(u_k) + \mathbf{s}_{bw},\, t-1\right) \tag{7}$$

$$\mathbf{C}_{t'}^{d}(\mathbf{r}) = \sum_{k=1}^{K} T_{t'}^{d}(u_k)\, \alpha^{d}\!\left(\sigma_{t'}^{d}(u_k)\, \delta_k\right) \mathbf{c}_{t'}^{d}(u_k) \tag{8}$$

$$\mathcal{L}_{dyn} = \sum_{t' \in \{t,\, t-1,\, t+1\}} \sum_{ij} \left\| \left(\mathbf{C}_{t'}^{d}(\mathbf{r}_{ij}) - \mathbf{C}^{gt}(\mathbf{r}_{ij})\right) \right\|_2^2 \tag{9}$$

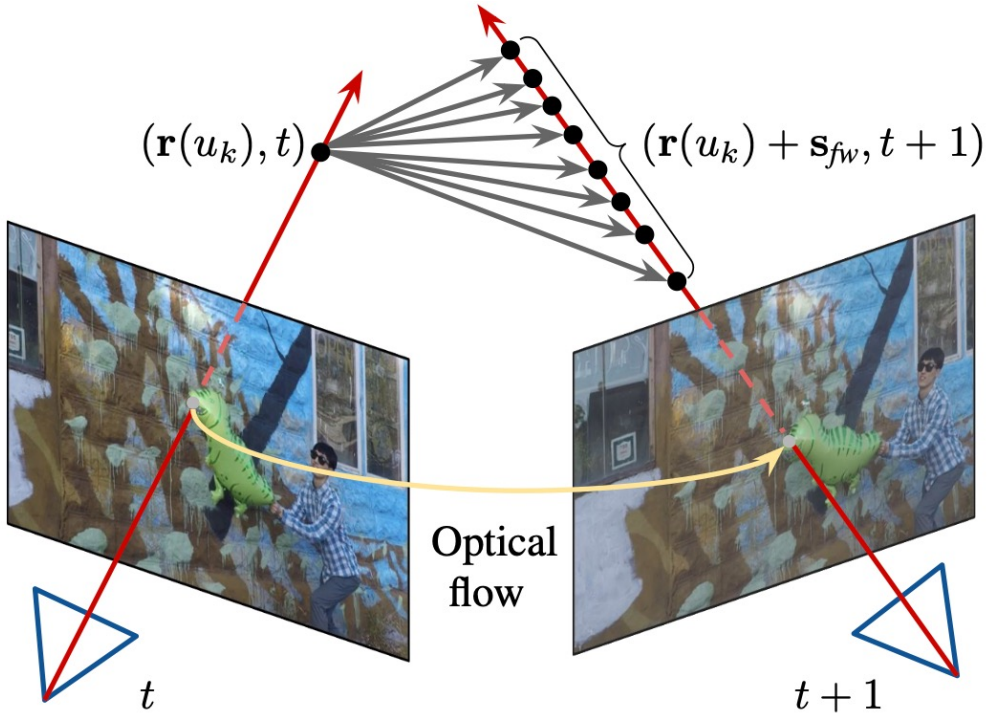
Regularization Losses for Dynamic NeRF

- **Motion matching loss**
  - Minimize the endpoint error between the estimated optical flow and our scene flow induced optical flow
  - Since we jointly train our model with both photometric loss and motion matching loss, finally, our learned volume density helps render a more accurate flow than the estimated.

(a) Input (b) Induced flow (c) Estimated flow

Regularization Losses for Dynamic NeRF

- **Motion regularization**
  - 2D optical flow does not fully resolve all ambiguity, since 1D family vectors produces the same 2D optical flow
  - Regularize the scene flow to be slow and temporally smooth
  - Cycle consistency regularization improve the consistency of the scene flow



$$\mathcal{L}_{slow} = \sum_{ij} \|\mathbf{s}_{fw}(\mathbf{r}_{ij})\|_1 + \|\mathbf{s}_{bw}(\mathbf{r}_{ij})\|_1 \tag{10}$$

$$\mathcal{L}_{smooth} = \sum_{ij} \|\mathbf{s}_{fw}(\mathbf{r}_{ij}) + \mathbf{s}_{bw}(\mathbf{r}_{ij})\|_2^2 \tag{11}$$

$$\mathcal{L}_{cyc} = \sum \|\mathbf{s}_{fw}(\mathbf{r}, t) + \mathbf{s}_{bw}(\mathbf{r} + \mathbf{s}_{fw}(\mathbf{r}, t),\ t+1)\|_2^2 \tag{12}$$

$$+ \|\mathbf{s}_{bw}(\mathbf{r}, t) + \mathbf{s}_{fw}(\mathbf{r} + \mathbf{s}_{bw}(\mathbf{r}, t),\ t-1)\|_2^2 \tag{13}$$

Regularization Losses for Dynamic NeRF

- **Sparsity regularization**
  - Minimize the entropy of the rendering weights $T^d \alpha^d$ along each ray so that few samples dominate the rendering

- **Depth order loss**
  - For a moving object, we can either interpret it as
    - close and slowly
    - far away and fast
  - Leverage the MiDaS depth estimation to estimate the input depth.
  - With static NeRF estimates accurate depth, we constrain our dynamic NeRF with it

$$\mathcal{L}_{depth} = \sum_{ij} \left\| \overline{\mathbf{D}^d}(\mathbf{r}_{ij}) - \overline{\mathbf{D}^{gt}}(\mathbf{r}_{ij}) \right\|_2^2 + \left\| (\mathbf{D}^d(\mathbf{r}_{ij}) - \mathbf{D}^s(\mathbf{r}_{ij})) \cdot (1 - \mathbf{M}(\mathbf{r}_{ij})) \right\|_2^2,$$

Regularization Losses for Dynamic NeRF

- **3D temporal consistency loss**
  - If an object remains unmoved for a while, the network can not learn the correct volume density and color of the occluded background, the model may generate holes.
  - Enforce the volume density and color of each 3D position to match its scene flow neighbors'

- **Rigidity regularization of the scene flow**
  - If 3D position has no motion, model prefers to explain by the static NeRF, blending weight b to be closed to 1 and the scene flow is forced to be zero.
  - For a non-rigid position, the blending weight b should be 0.

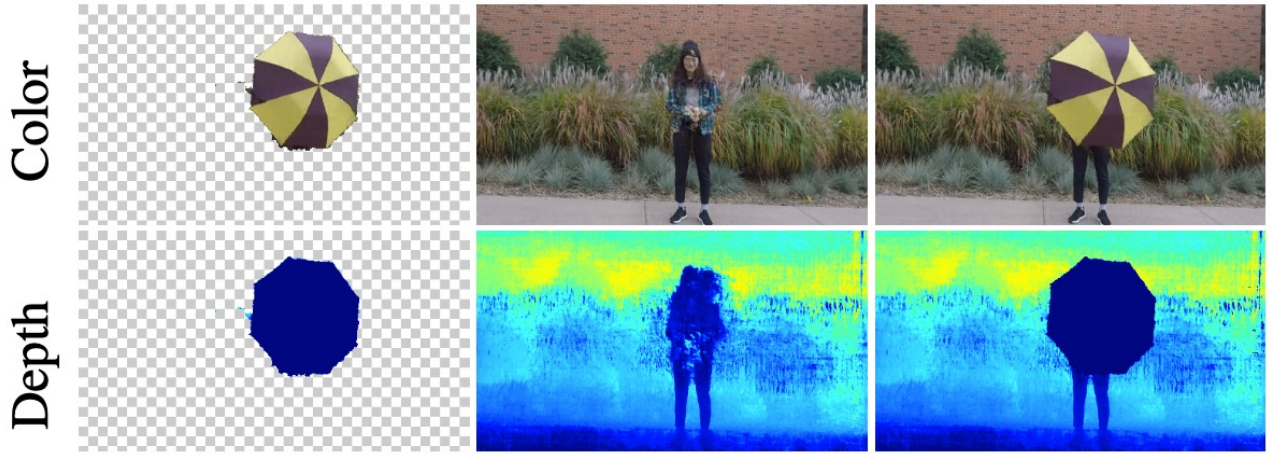
Final Loss

- **Combined model**

$$\mathbf{C}^{full}(\mathbf{r}) = \sum_{k=1}^{K} T^{full}\Big(\alpha^d(\sigma^d\delta_k)(1-b)\mathbf{c}^d + \alpha^s(\sigma^s\delta_k)b\mathbf{c}^s\Big) \tag{14}$$

- **Full rendering photometric loss**

$$\mathcal{L}_{full} = \sum_{ij} \left\| \mathbf{C}^{full}(\mathbf{r}_{ij}) - \mathbf{C}^{gt}(\mathbf{r}_{ij}) \right\|_2^2 \tag{15}$$



(a) Dynamic NeRF (b) Static NeRF (c) Full model

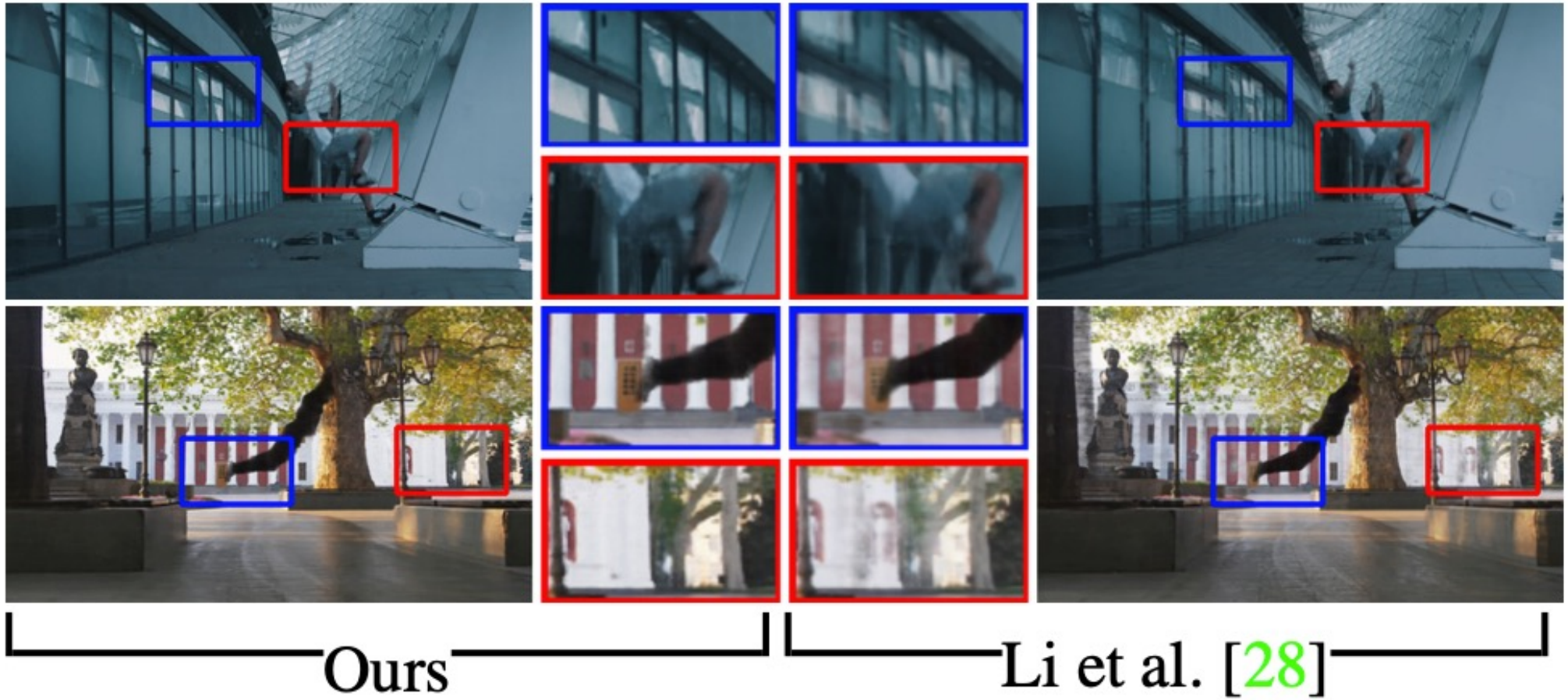
Experiment

PSNR ↑ / LPIPS ↓	Jumping	Skating	Truck	Umbrella	Balloon1	Balloon2	Playground	Average
NeRF	20.58 / 0.305	23.05 / 0.316	22.61 / 0.225	21.08 / 0.441	19.07 / 0.214	24.08 / 0.098	20.86 / 0.164	21.62 / 0.252
NeRF + time	16.72 / 0.489	19.23 / 0.542	17.17 / 0.403	17.17 / 0.752	17.33 / 0.304	19.67 / 0.236	13.80 / 0.444	17.30 / 0.453
Yoon et al. [62]	20.16 / 0.148	21.75 / 0.135	23.93 / **0.109**	20.35 / 0.179	18.76 / 0.178	19.89 / 0.138	15.09 / 0.183	19.99 / 0.153
Tretschk et al. [55]	19.38 / 0.295	23.29 / 0.234	19.02 / 0.453	19.26 / 0.427	16.98 / 0.353	22.23 / 0.212	14.24 / 0.336	19.20 / 0.330
Li et al. [28]	24.12 / 0.156	**28.91** / 0.135	**25.94** / 0.171	22.58 / 0.302	21.40 / 0.225	24.09 / 0.228	20.91 / 0.220	23.99 / 0.205
Ours	**24.23** / **0.144**	28.90 / **0.124**	25.78 / 0.134	**23.15** / **0.146**	**21.47** / **0.125**	**25.97** / **0.059**	**23.65** / **0.093**	**24.74** / **0.118**

NeRF + time | Yoon et al. [62] | Tretschk et al. [55] | Li et al. [28] | Ours | Ground truth

Experiment

- **rigidity regularization** are the keys to better visual results
  - We learn a time-varying blending weight.
  - Without this regularization, the background becomes time-variant and leads to floating artifacts



Ablation study

- **depth order loss**
  - Training with depth order loss ensures the correct relative depth of the dynamic object.
- **Motion regularize loss**
  - Regularizing our scene flow prediction in dynamic NeRF can help handle videos with large object motion.

Without depth order loss | *With* depth order loss | *Without* motion regularization | *With* motion regularization

	PSNR ↑	SSIM ↑	LPIPS ↓
Ours w/o $\mathcal{L}_{depth}$	22.99	0.8170	0.117
Ours w/o $\mathcal{L}_{motion}$	22.61	0.8027	0.137
Ours w/o rigidity	22.73	0.8142	0.118
Ours	**23.65**	**0.8452**	**0.093**

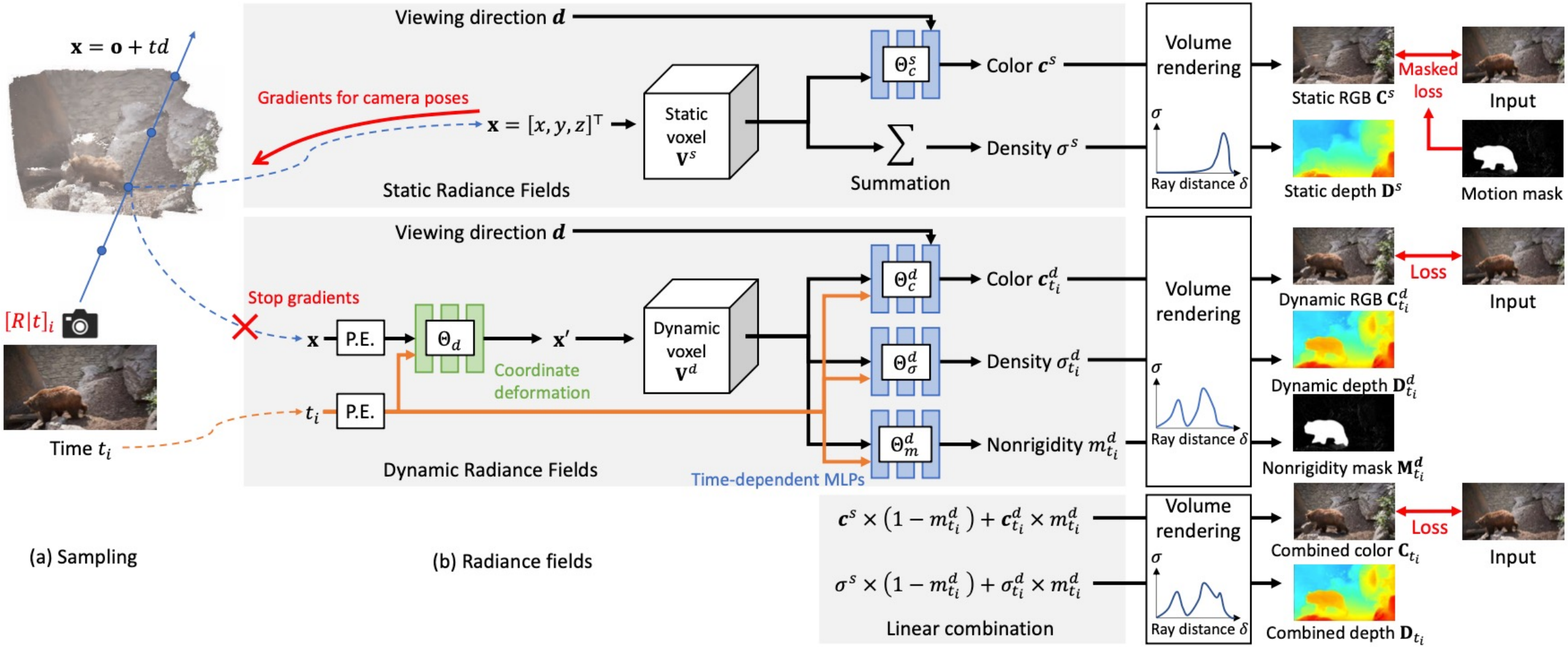
Conclusion of Related work

- D-NeRF:
  - represent time-varying deformations with two modules
    - one that learns the deformation field of the scene between original space and the canonical space
    - another that learns canonical configuration
- Dynamic-NeRF:
  - scene flow based regularization for enforcing temporal consistency
  - jointly training a time-invariant static NeRF and a time-varying Dynamic NeRF, and learn how to blend it

Outline

- Introduction
- Related Work
- Framework
- Method
- Experiment
- Conclusion

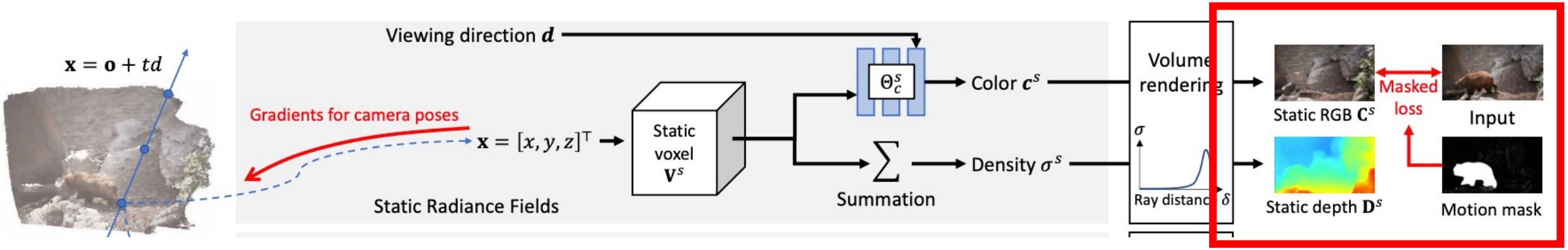
Framework


$\mathbf{x} = \mathbf{o} + td$
$[R|t]_i$
Time t_i
(a) Sampling
Viewing direction $\boldsymbol{d}$
Gradients for camera poses
$\mathbf{x} = [x, y, z]^\top$
Static voxel $\mathbf{V}^s$
Θ_c^s
Color $\boldsymbol{c}^s$
Density σ^s
Summation
Static Radiance Fields
Viewing direction $\boldsymbol{d}$
Stop gradients
$\mathbf{x}$
P.E.
Θ_d
$\mathbf{x}'$
Coordinate deformation
t_i
P.E.
Dynamic voxel $\mathbf{V}^d$
Θ_c^d
Color $\boldsymbol{c}_{t_i}^d$
Θ_σ^d
Density $\sigma_{t_i}^d$
Θ_m^d
Nonrigidity $m_{t_i}^d$
Dynamic Radiance Fields
Time-dependent MLPs
$\boldsymbol{c}^s \times (1 - m_{t_i}^d) + \boldsymbol{c}_{t_i}^d \times m_{t_i}^d$
$\sigma^s \times (1 - m_{t_i}^d) + \sigma_{t_i}^d \times m_{t_i}^d$
Linear combination
(b) Radiance fields
Volume rendering
σ
Ray distance δ
Static RGB $\mathbf{C}^s$
Masked loss
Input
Static depth $\mathbf{D}^s$
Motion mask
Dynamic RGB $\mathbf{C}_{t_i}^d$
Loss
Input
Dynamic depth $\mathbf{D}_{t_i}^d$
Nonrigidity mask $\mathbf{M}_{t_i}^d$
Combined color $\mathbf{C}_{t_i}$
Loss
Input
Combined depth $\mathbf{D}_{t_i}$

Outline

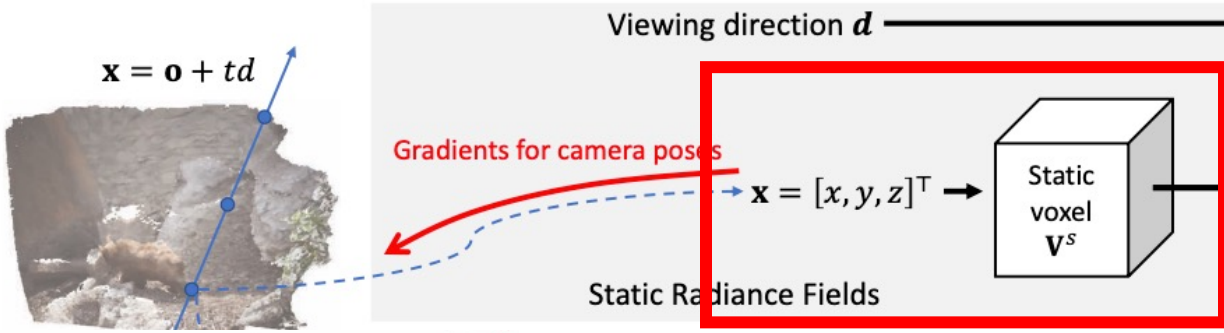
- Introduction
- Related Work
- Framework
- Method
- Experiment
- Conclusion

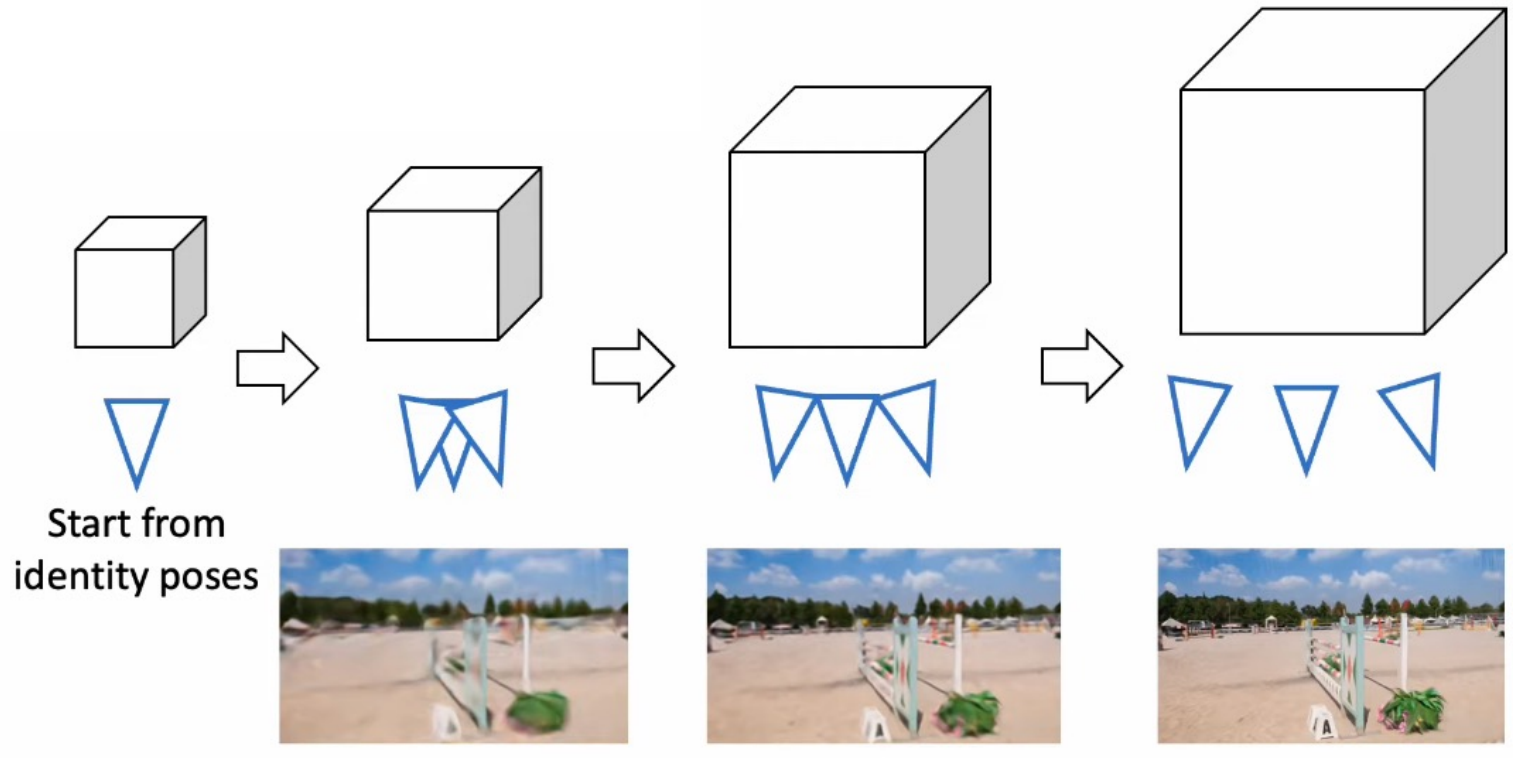
Motion Mask generation



- Excluding dynamic regions helps improve the robustness of camera pose estimation
- Leverage Mask R-CNN
- Epipolar Geometry
  - Estimate the fundamental matrix using the optical flow from consecutive frames
  - Calculate and threshold the Sampson distance (the distance of each pixel to the estimated epipolar line)

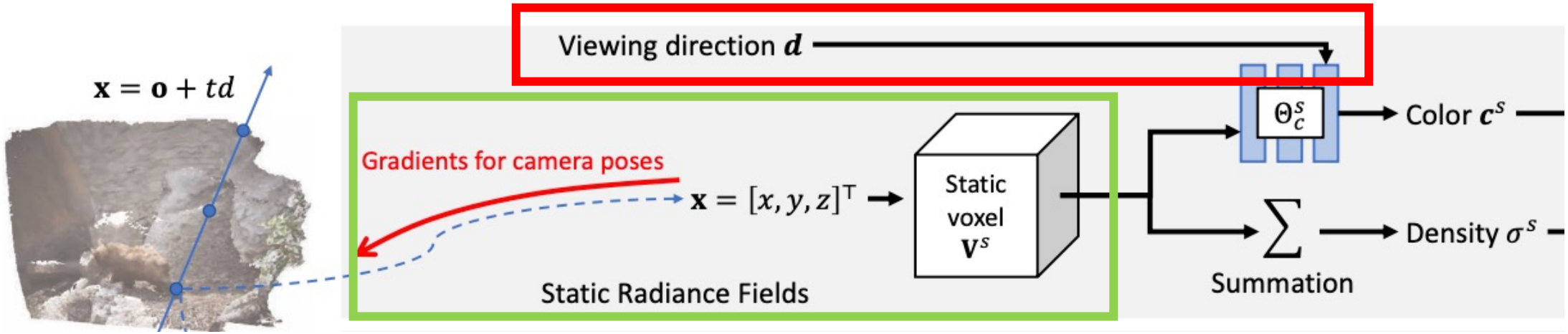
Coarse-to-fine static scene reconstruction





- Optimize, start with a smaller static voxel resolution and progressively increase the voxel resolution during the training.
- This coarse-to-fine strategy is essential to the camera pose estimation as the energy surface will become smoother.

Late viewing direction conditioning



- Fuse the viewing direction only in the last layer of the color MLP
- Without the late viewing direction conditioning, the optimization could minimize the photometric loss by optimizing the MLP and lead to erroneous camera poses and geometry estimation

Photometric Losses

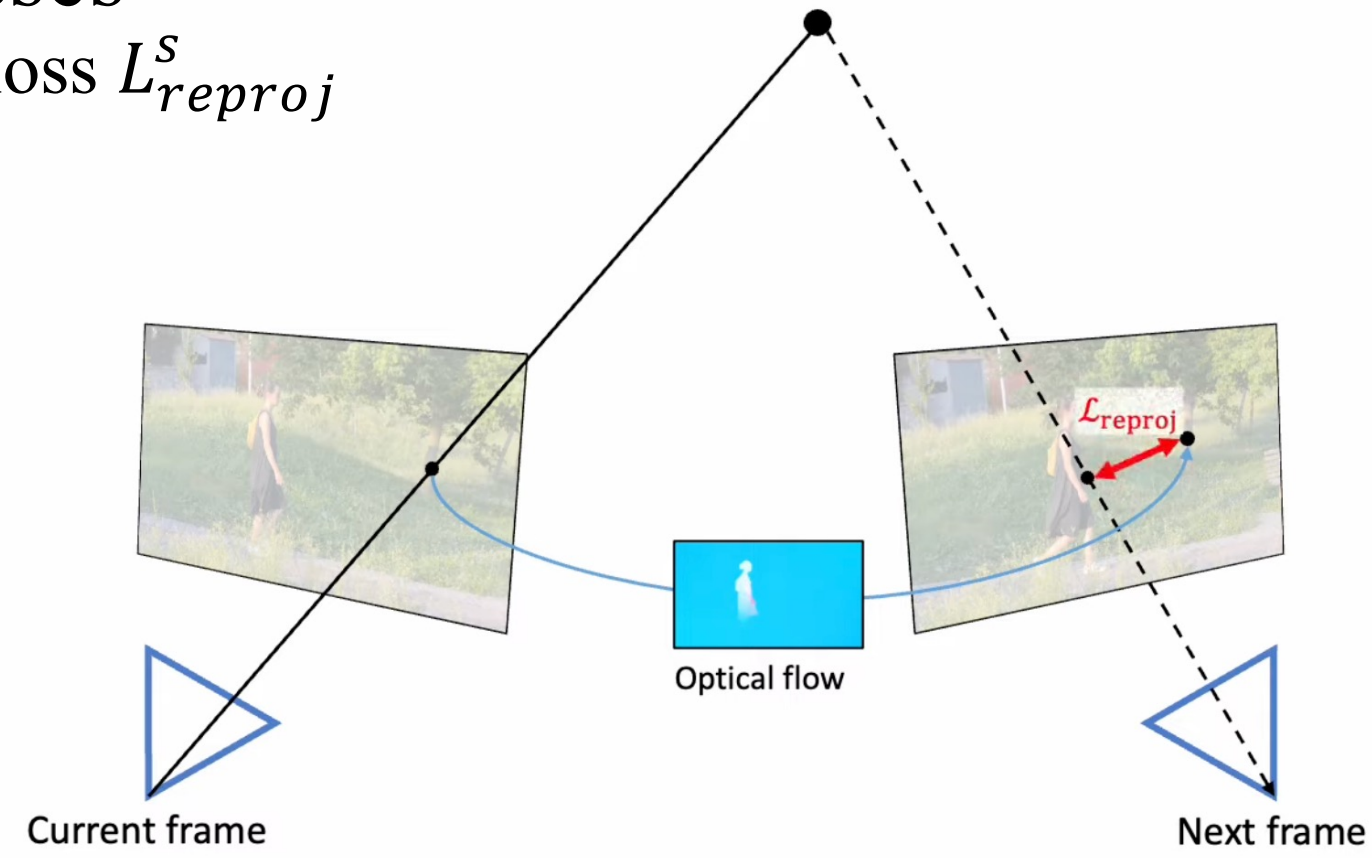
$$\hat{\mathbf{C}}(\mathbf{r}) = \sum_{i=1}^{N} T(i)(1 - \exp(-\sigma(i)\delta(i)))\mathbf{c}(i),$$
$$T(i) = \exp(-\sum_{i-1}^{j} \sigma(j)\delta(j)), \tag{2}$$

$$\mathcal{L}_c^s = \left\| (\hat{\mathbf{C}}^s(\mathbf{r}) - \mathbf{C}(\mathbf{r})) \cdot (1 - \mathbf{M}(\mathbf{r})) \right\|_2^2, \tag{4}$$

term	meaning
M	the motion mask
$\boldsymbol{\delta}$	the distance between two consecutive sample points along the ray
N	the number of samples along each ray
T	accumulated transparency

Auxiliary Losses

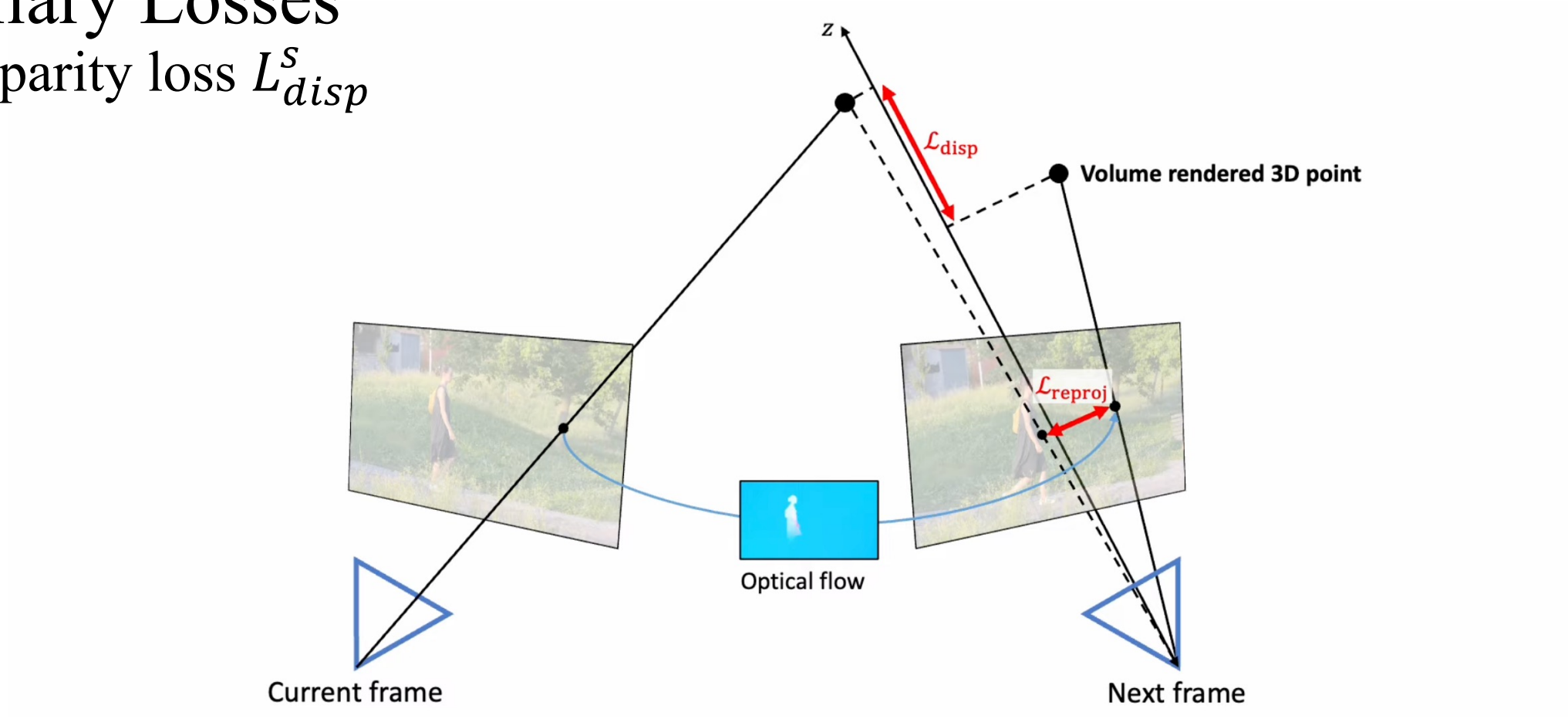
(1) Reprojection loss L^{s}_{reproj}



- We use 2D optical flow estimated by RAFT to guide the training.
- Volume render all the sampled 3D points along a ray to generate a surface point
- Reproject this point onto its neighbor frame and calculate the reprojection error

Auxiliary Losses

(2) Disparity loss L_{disp}^{s}

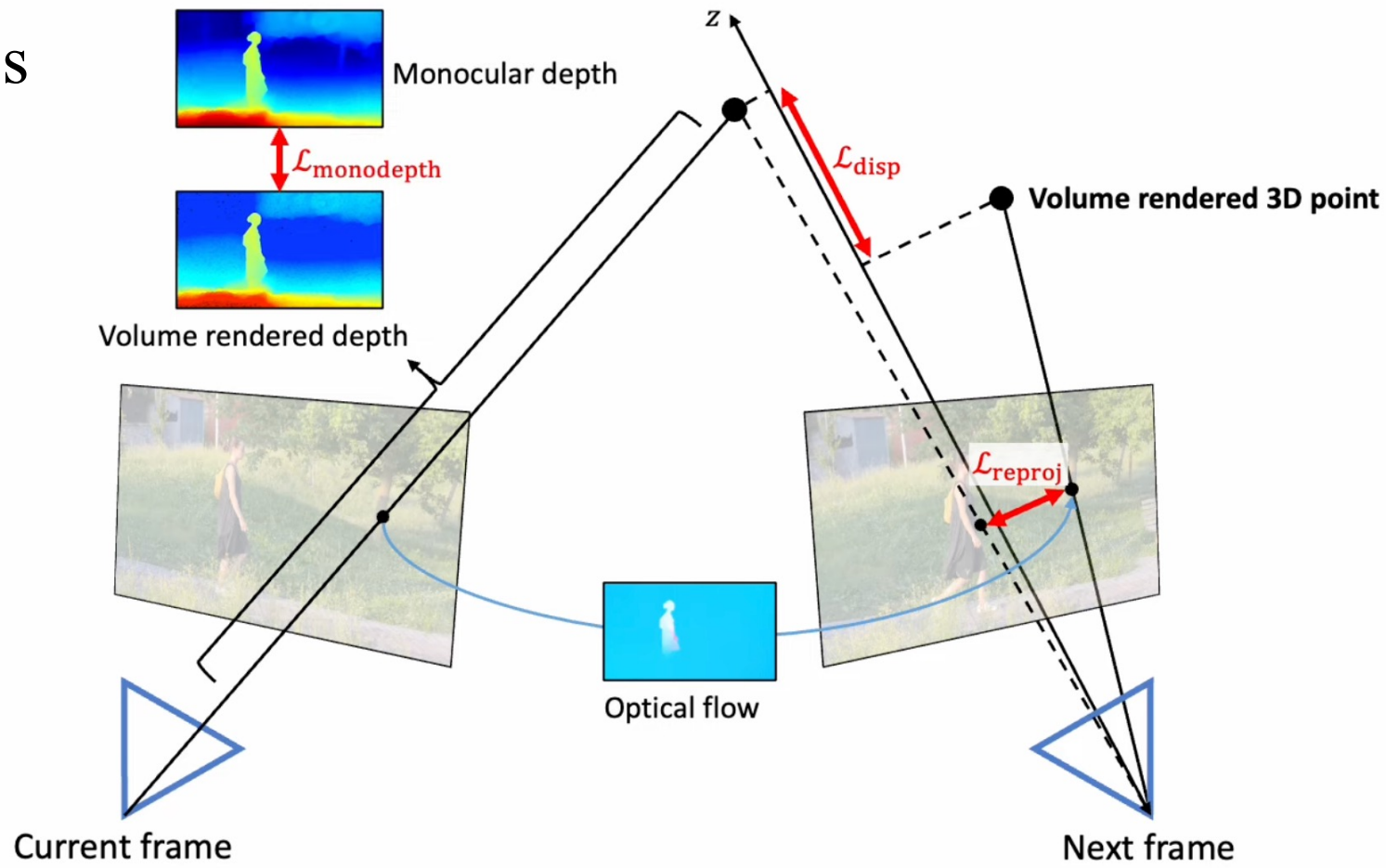


- Regularize the error in the z-direction (in the camera coordinate)
- Volume render the two points into 3D space and calculate the error of the z component
- Care more about the near than the far, we compute this loss in the inverse-depth domain

Auxiliary Losses

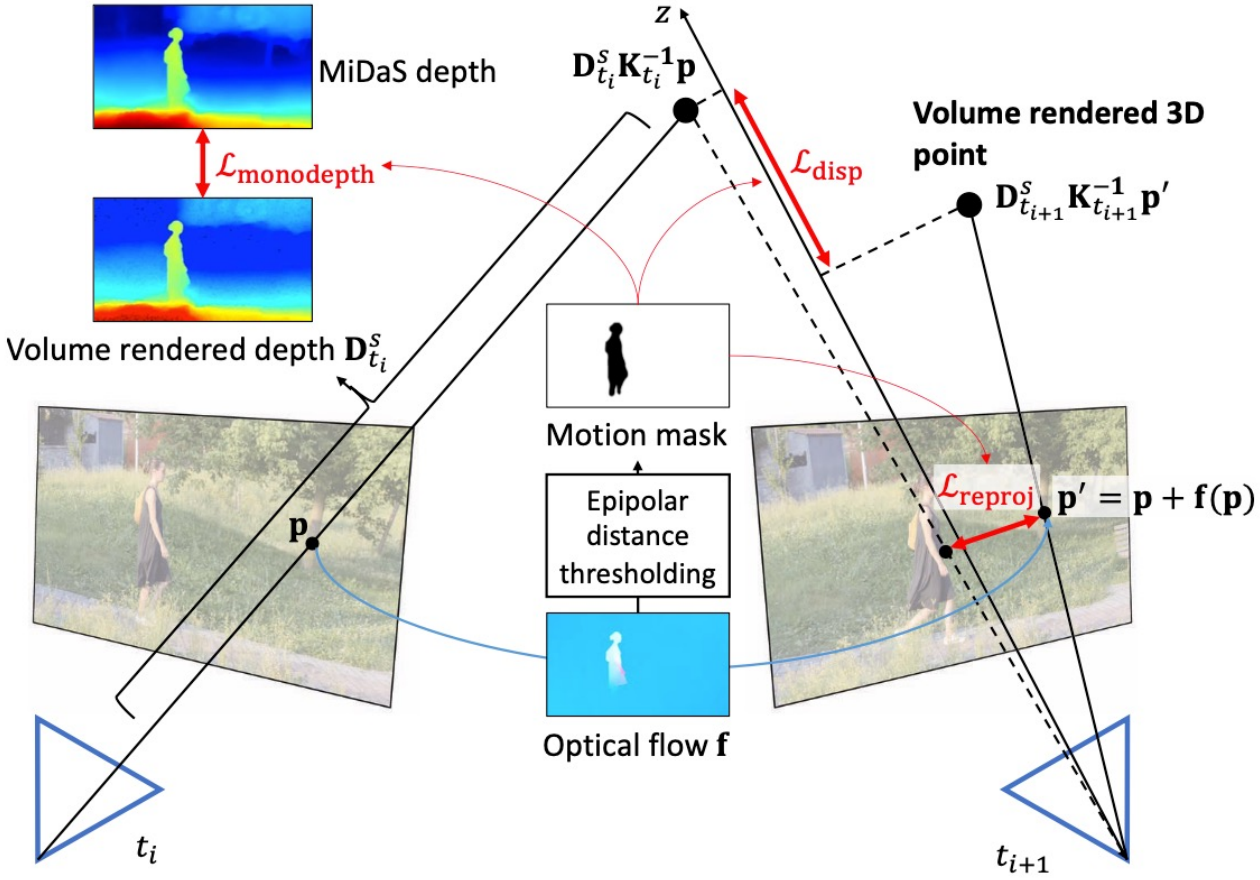
(3) Monocular depth loss

$$L^{s}_{monodepth}$$



- Pre-calculate the depth map using MiDaS
- Enforce the depth order from multiple pixels of the same frame to match the order of a monocular depth map.

Static Radiance Field final Losses



(a) Static radiance field reconstruction and pose estimation

$$\mathcal{L}^s = \mathcal{L}^s_c + \lambda^s_{\text{reproj}}\mathcal{L}^s_{\text{reproj}} + \lambda^s_{\text{disp}}\mathcal{L}^s_{\text{disp}} + \lambda^s_{\text{monodepth}}\mathcal{L}^s_{\text{monodepth}}. \tag{5}$$

The impact of design choices

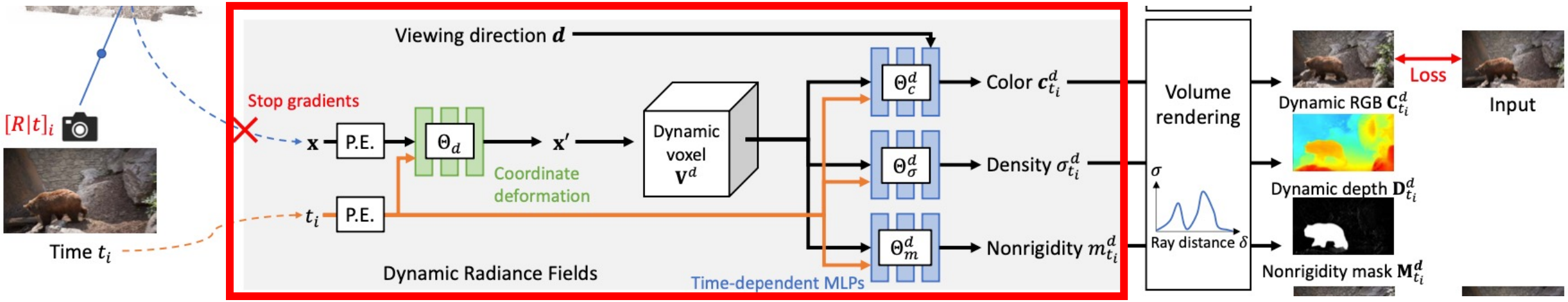
(a) w/o coarse-to-fine

(b) w/o monocular depth prior

(c) w/o late viewing direction conditioning

(d) Full model

Handling temporal information



$$\mathcal{L}_c^d = \left\| \hat{\mathbf{C}}^d(\mathbf{r}) - \mathbf{C}(\mathbf{r}) \right\|_2^2, \qquad (6)$$

Scene flow modeling

$$(S_{i\to i+1}, S_{i\to i-1}) = \mathrm{MLP}_{\theta_{\mathrm{sf}}}(x, y, z, t_i), \qquad (7)$$

term	meaning
$S_{i\to i+1}$	the 3D scene flow of the 3D point (x, y, z) at time t_i to t_{i+1}
$S_{i\to i-1}$	the 3D scene flow of the 3D point (x, y, z) at time t_i to t_{i-1}

- Need auxiliary loss as external priors to better model the dynamic movements
- Similar to the static part, but we need to model the movements of the 3D points

Scene flow modeling training loss

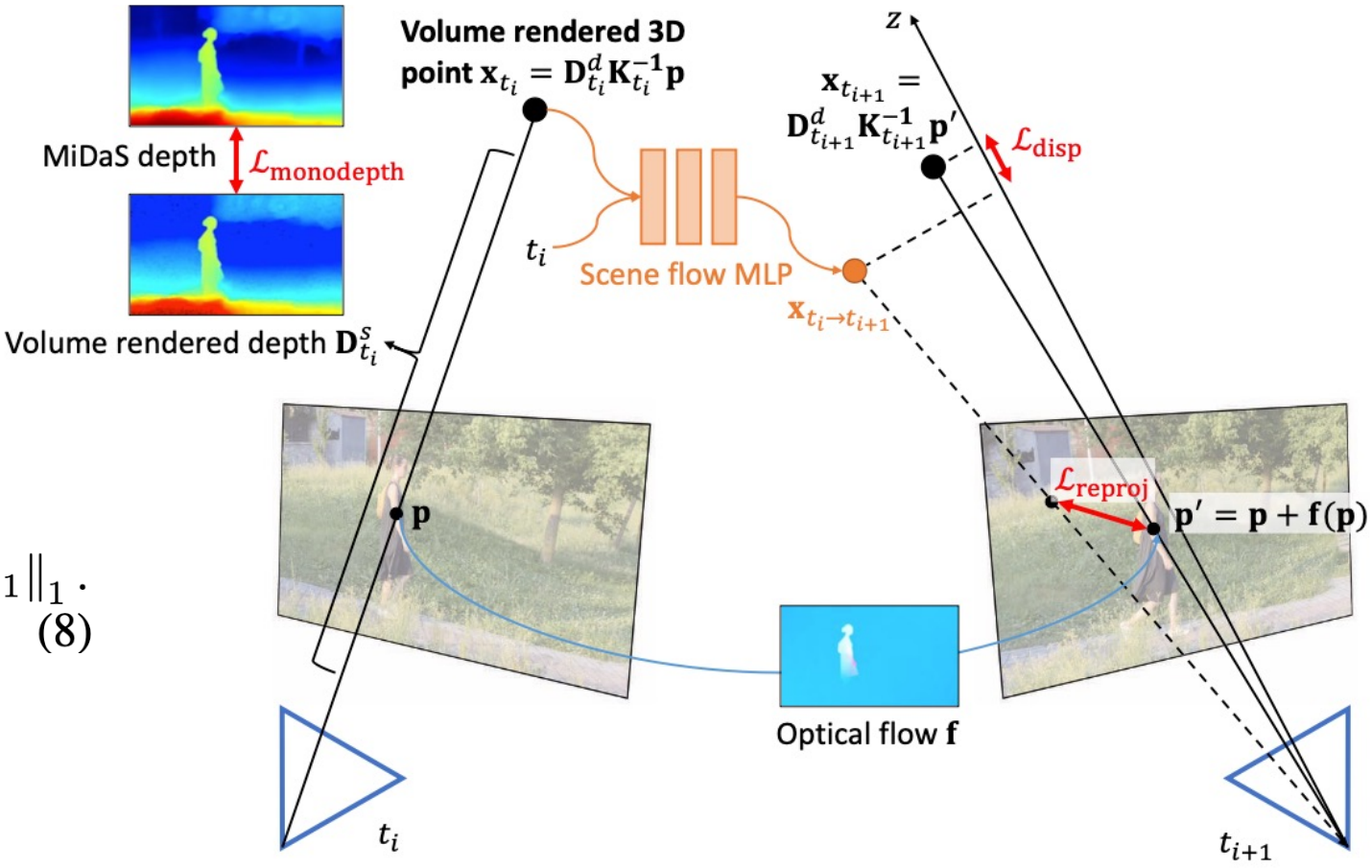
(1) Reprojection loss L^d_{reproj}

(2) Disparity loss L^d_{disp}

(3) Monocular depth loss $L^d_{monodepth}$

- regularize the 3D motion prediction

$$\mathcal{L}_{\text{sf}}^{\text{reg}} = \|S_{i\to i+1} + S_{i\to i-1}\|_1 + \|S_{i\to i+1}\|_1 + \|S_{i\to i-1}\|_1 \,. \tag{8}$$



(b) Dynamic radiance field reconstruction

Other Dynamic part loss

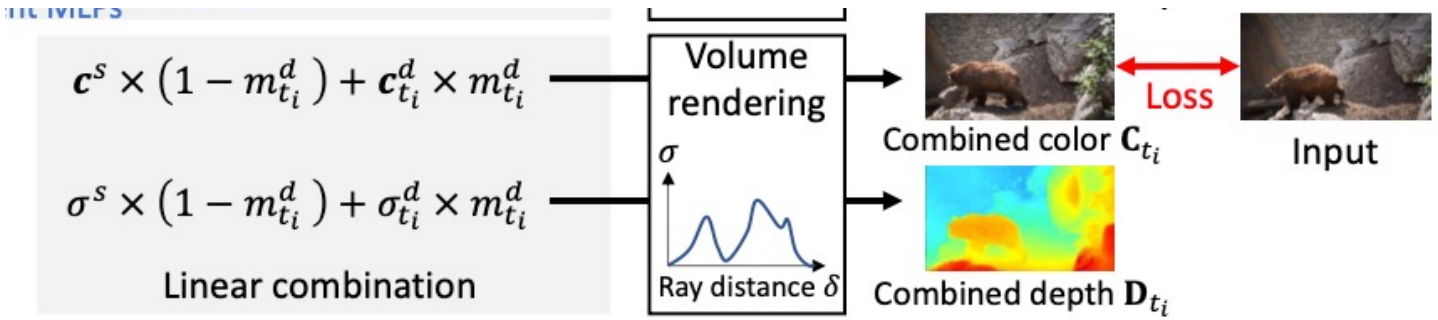
- supervise the nonrigidity mask M_d with motion mask M

$$\mathcal{L}_m^d = \left\| \mathbf{M}^d - \mathbf{M} \right\|_1 . \tag{9}$$

- overall loss of the dynamic part

$$\begin{aligned} \mathcal{L}^d = \mathcal{L}_c^d + \lambda_{\text{reproj}}^d \mathcal{L}_{\text{reproj}}^d + \lambda_{\text{disp}}^d \mathcal{L}_{\text{disp}}^d + \\ \lambda_{\text{monodepth}}^d \mathcal{L}_{\text{monodepth}}^d + \lambda_{\text{sf}}^{\text{reg}} \mathcal{L}_{\text{sf}}^{\text{reg}} + \lambda_m^d \mathcal{L}_m^d . \end{aligned} \tag{10}$$

Total training loss



- linearly compose the static and dynamic parts into the final results

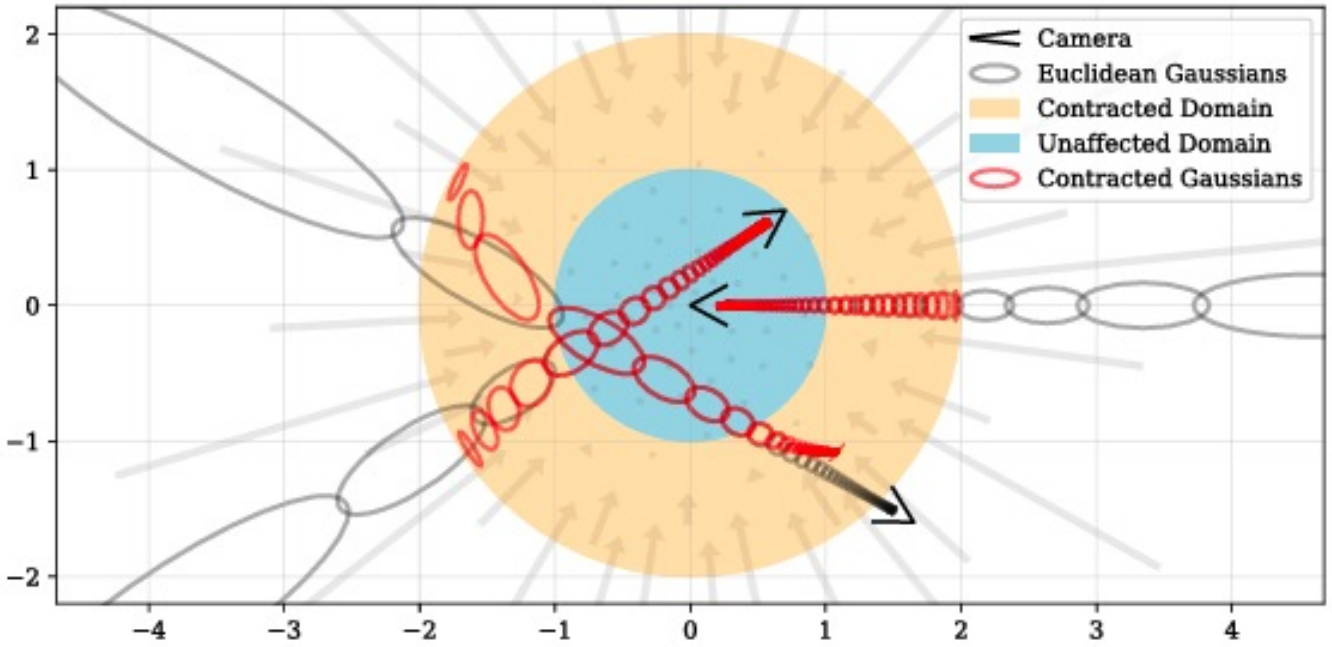
$$\hat{\mathbf{C}}(\mathbf{r}) = \sum_{i=1}^{N} T(i)(m^d(1 - \exp(-\sigma^d(i)\delta(i)))\mathbf{c}^d(i) + (1 - m^d)(1 - \exp(-\sigma^s(i)\delta(i)))\mathbf{c}^s(i)). \tag{11}$$

- total training loss

$$\mathcal{L} = \left\| \hat{\mathbf{C}}(\mathbf{r}) - \mathbf{C}(\mathbf{r}) \right\|_2^2 + \mathcal{L}^s + \mathcal{L}^d. \tag{12}$$

Implementation detail

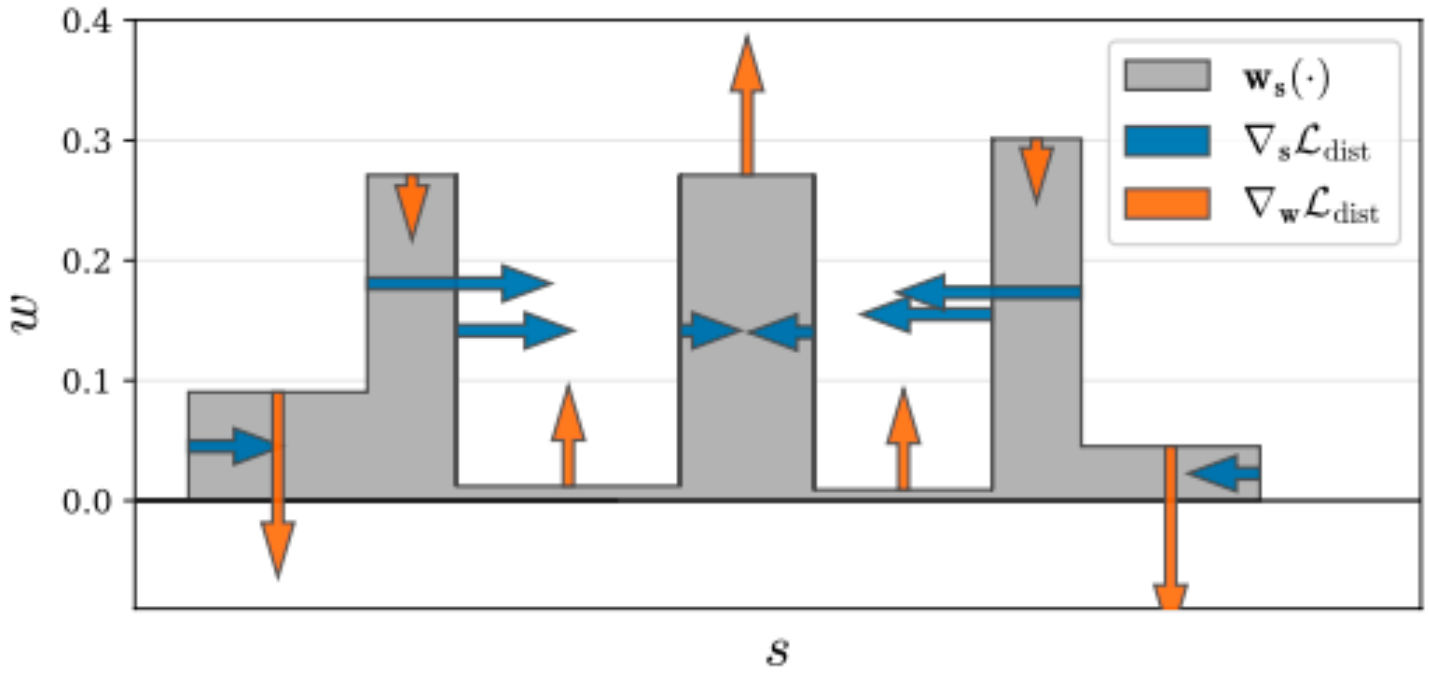
- The training process takes around 28 hours with one NVIDIA V100 GPU
- we parameterize the scenes with **normalized device coordinates (NDC)**
- To handle unbounded scenes in the wild videos, we parameterize the scenes using the **contraction parameterization**.



Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields

Implementation detail

- Distortion loss:
  - suppresses “floaters” (pieces of semi-transparent material floating in space)
  - regularize the distribution of weights across different segments of a ray
  - encourages each ray to be as compact as possible



Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields

Outline

- Introduction
- Related Work
- Framework
- Method
- Experiment
- Conclusion

Evaluation on Camera Poses Estimation

Method	ATE (m)	RPE trans (m)	RPE rot (deg)
R-CVD [35]	0.360	0.154	3.443
DROID-SLAM [70]	0.175	0.084	1.912
ParticleSfM [83]	0.129	**0.031**	**0.535**
NeRF - - [73]	0.433	0.220	3.088
BARF [40]	0.447	0.203	6.353
Ours	**0.089**	0.073	1.313

- We exclude the COLMAP results since it fails to produce poses in 5 out of 14 sequences in MPI Sintel dataset.

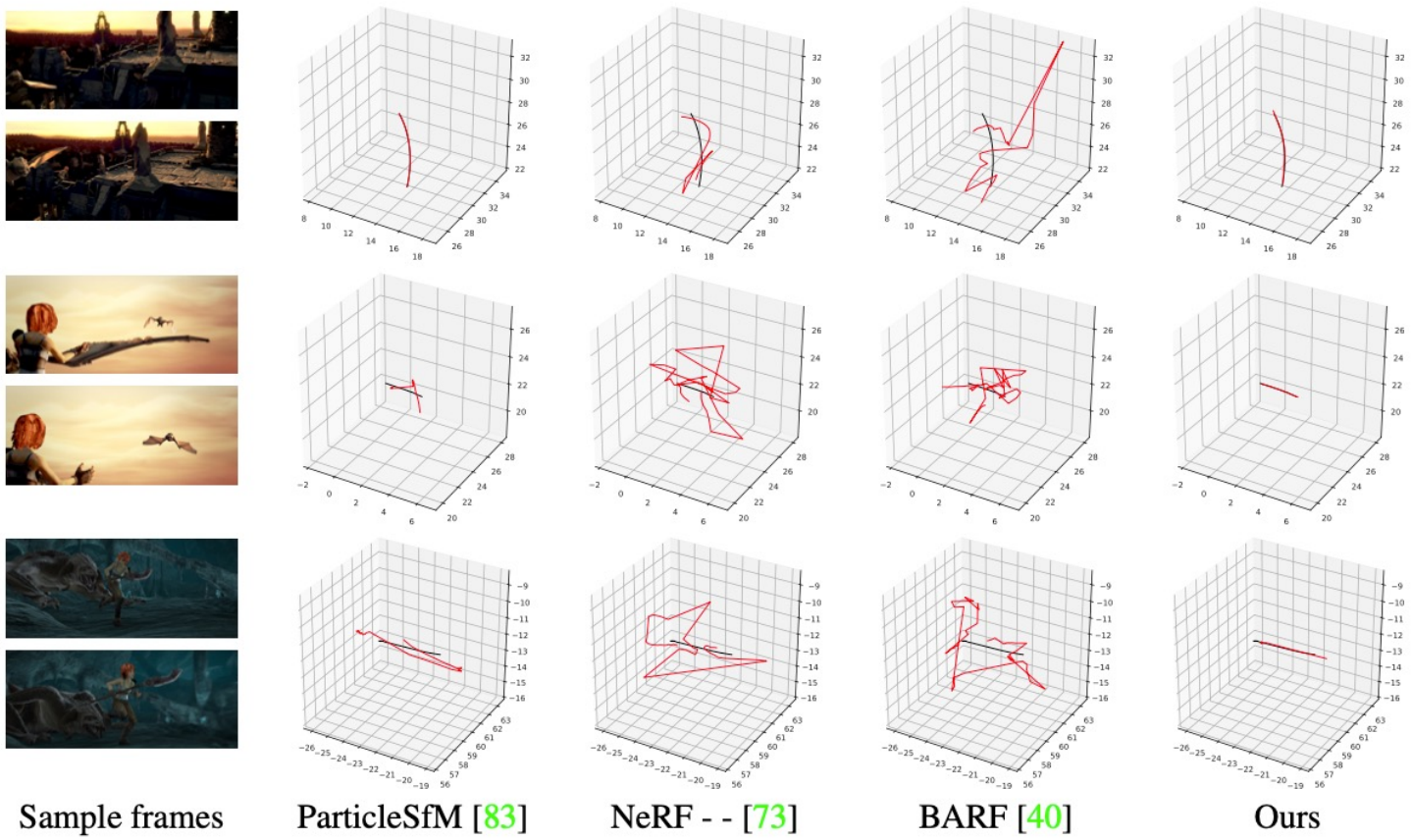
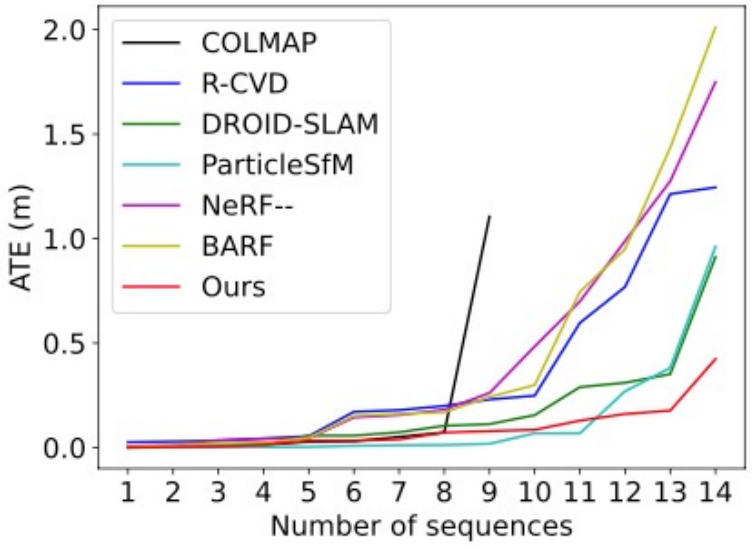
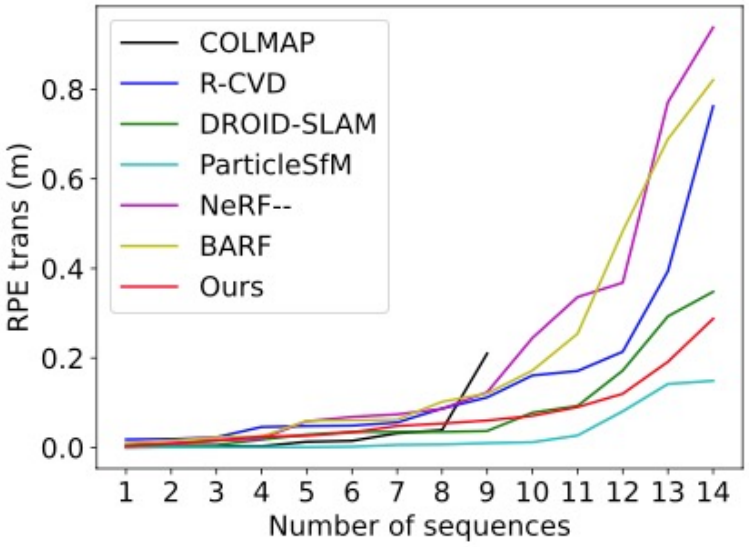


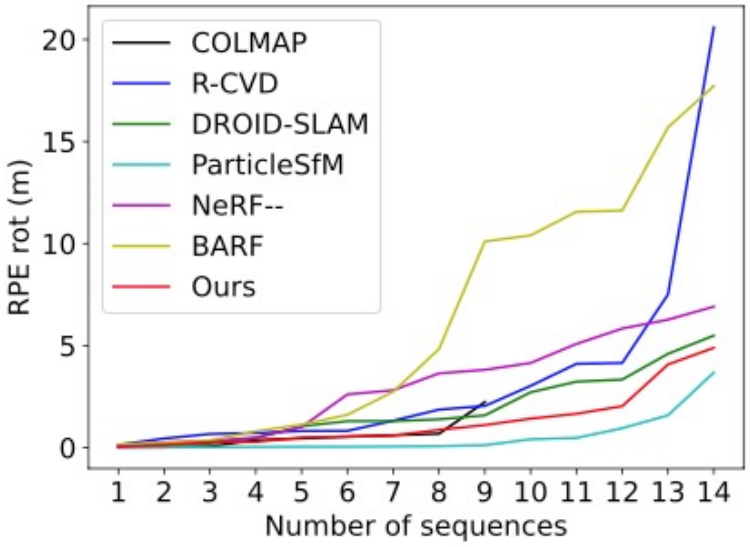
Figure 5. **Qualitative results of moving camera localization on the MPI Sintel dataset.**

Evaluation on Camera Poses Estimation

- The sorted error plots showing both the accuracy and completeness/robustness in the MPI Sintel dataset.

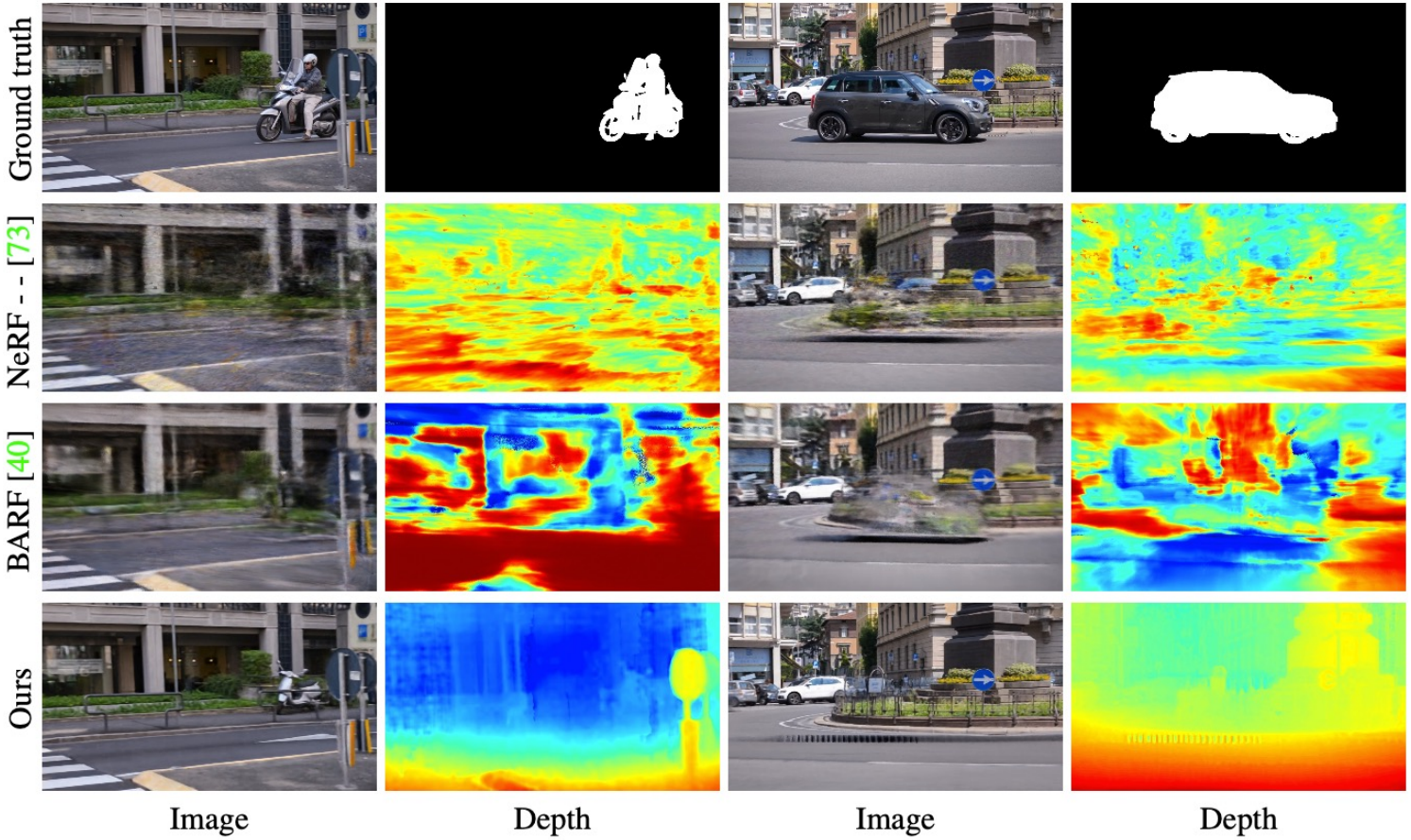






Evaluation on Camera Poses Estimation

- Qualitative results of static view synthesis on the DAVIS dataset from unknown camera poses and ground truth foreground masks.



Evaluation on Dynamic View Synthesis

- We report the average PSNR and LPIPS results with comparisons to existing methods on Dynamic Scene dataset

PSNR ↑ / LPIPS ↓	Jumping	Skating	Truck	Umbrella	Balloon1	Balloon2	Playground	Average
NeRF* [44]	20.99 / 0.305	23.67 / 0.311	22.73 / 0.229	21.29 / 0.440	19.82 / 0.205	24.37 / 0.098	21.07 / 0.165	21.99 / 0.250
D-NeRF [56]	22.36 / 0.193	22.48 / 0.323	24.10 / 0.145	21.47 / 0.264	19.06 / 0.259	20.76 / 0.277	20.18 / 0.164	21.48 / 0.232
NR-NeRF* [71]	20.09 / 0.287	23.95 / 0.227	19.33 / 0.446	19.63 / 0.421	17.39 / 0.348	22.41 / 0.213	15.06 / 0.317	19.69 / 0.323
NSFF* [39]	24.65 / 0.151	29.29 / 0.129	25.96 / 0.167	22.97 / 0.295	21.96 / 0.215	24.27 / 0.222	21.22 / 0.212	24.33 / 0.199
DynamicNeRF* [24]	24.68 / 0.090	**32.66 / 0.035**	28.56 / 0.082	23.26 / 0.137	22.36 / 0.104	**27.06 / 0.049**	24.15 / 0.080	**26.10** / 0.082
HyperNeRF [53]	18.34 / 0.302	21.97 / 0.183	20.61 / 0.205	18.59 / 0.443	13.96 / 0.530	16.57 / 0.411	13.17 / 0.495	17.60 / 0.367
TiNeuVox [20]	20.81 / 0.247	23.32 / 0.152	23.86 / 0.173	20.00 / 0.355	17.30 / 0.353	19.06 / 0.279	13.84 / 0.437	19.74 / 0.285
Ours w/ COLMAP poses	**25.66 / 0.071**	28.68 / 0.040	**29.13 / 0.063**	**24.26 / 0.089**	**22.37 / 0.103**	26.19 / 0.054	**24.96 / 0.048**	25.89 / **0.065**
Ours w/o COLMAP poses	24.27 / 0.100	28.71 / 0.046	28.85 / 0.066	23.25 / 0.104	21.81 / 0.122	25.58 / 0.064	25.20 / 0.052	25.38 / 0.079

- We compare the mPSNR and mSSIM scores with existing methods on the iPhone dataset

mPSNR ↑ / mSSIM ↑	Apple	Block	Paper-windmill	Space-out	Spin	Teddy	Wheel	Average
NSFF [39]	17.54 / 0.750	16.61 / 0.639	17.34 / 0.378	17.79 / 0.622	18.38 / 0.585	13.65 / 0.557	13.82 / 0.458	15.46 / 0.569
Nerfies [52]	17.64 / 0.743	17.54 / 0.670	17.38 / 0.382	17.93 / 0.605	19.20 / 0.561	13.97 / 0.568	13.99 / 0.455	16.45 / 0.569
HyperNeRF [53]	16.47 / 0.754	14.71 / 0.606	14.94 / 0.272	17.65 / 0.636	17.26 / 0.540	12.59 / 0.537	14.59 / 0.511	16.81 / 0.550
T-NeRF [25]	17.43 / 0.728	17.52 / 0.669	17.55 / 0.367	17.71 / 0.591	19.16 / 0.567	13.71 / 0.570	15.65 / 0.548	16.96 / 0.577
Ours	18.73 / 0.722	18.73 / 0.634	16.71 / 0.321	18.56 / 0.594	17.41 / 0.484	14.33 / 0.536	15.20 / 0.449	17.09 / 0.534

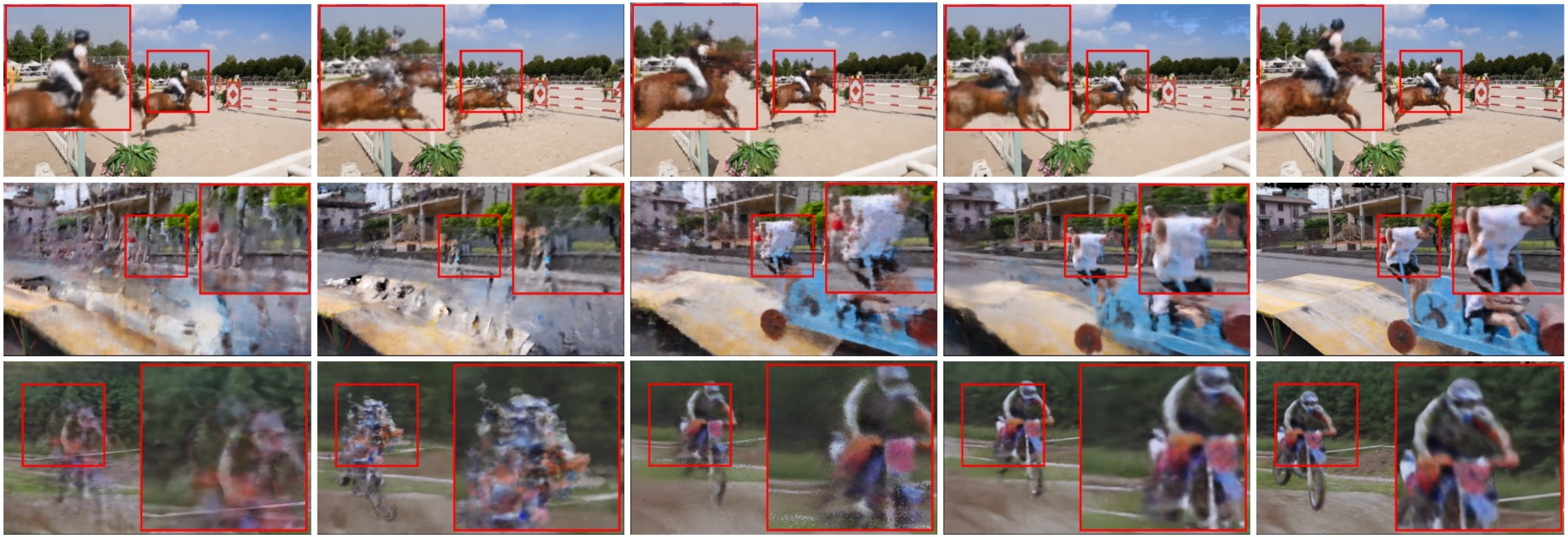
Evaluation on Dynamic View Synthesis

- Compared to other methods, our results are sharper, closer to the ground truth, and contain fewer artifacts.

NSFF [39] | DynamicNeRF [24] | HyperNeRF [53] | TiNeuVox [20] | Ours | Ground truth

Evaluation on Dynamic View Synthesis

- COLMAP fails to estimate the camera poses for 44 out of 50 sequences in the DAVIS dataset
- Run our method and give our camera poses to other methods as input
- Other can reconstruct consistent static scenes but generate artifacts for the dynamic parts



NSFF [39] DynamicNeRF [24] HyperNeRF [53] TiNeuVox [20] Ours

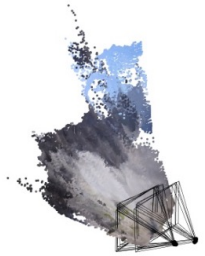
Ablation Study

(a) Pose estimation design choices

	PSNR ↑	SSIM ↑	LPIPS ↓
Ours w/o coarse-to-fine	12.45	0.4829	0.327
Ours w/o late viewing direction fusion	18.34	0.5521	0.263
Ours w/o stopping the dynamic gradients	21.47	0.7392	0.211
Ours	**25.20**	**0.9052**	**0.052**

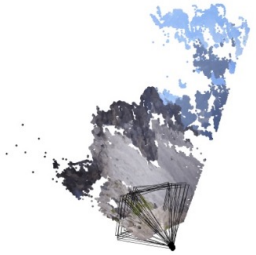
(b) Dynamic reconstruction achitectural designs

Dyn. model	Deform. MLP	Time-depend. MLPs	PSNR ↑	SSIM ↑	LPIPS ↓
			21.34	0.8192	0.161
✓	✓		22.37	0.8317	0.115
✓		✓	23.14	0.8683	0.083
✓	✓	✓	**25.20**	**0.9052**	**0.052**

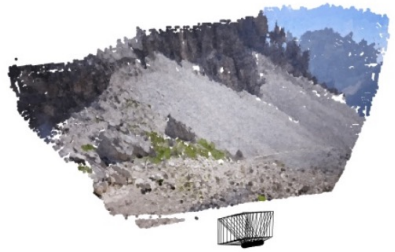


(a) w/o coarse-to-fine

(b) w/o monocular depth prior



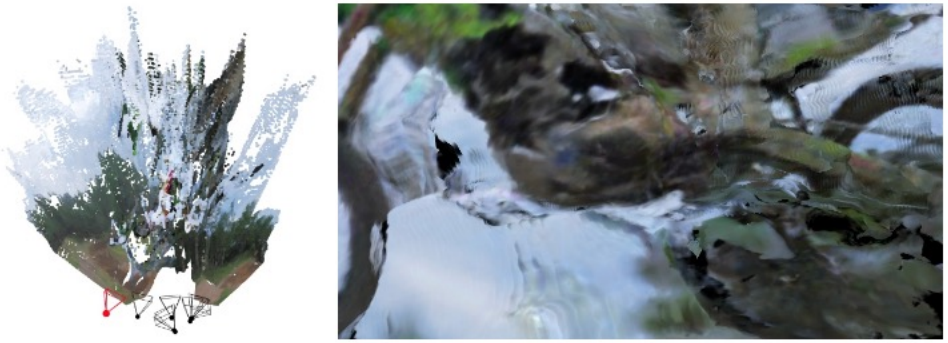
(c) w/o late viewing direction conditioning



(d) Full model

Failure Cases

- (a) In the cases that the camera is moving fast, the flow estimation fails and leads to wrong estimated poses and geometry
- (b) Our method assumes a shared intrinsic over the entire video and thus cannot handle changing focal length well.



(a) Fast moving camera

(b) Changing focal length

Outline

- Introduction
- Related Work
- Framework
- Method
- Experiment
- Conclusion

Conclusion

- Present robust dynamic radiance fields for space-time synthesis of casually captured monocular videos without requiring camera poses as input.

- Demonstrate that our approach can reconstruct accurate dynamic radiance fields from a wide range of challenging videos.

- Quantitative and qualitative evaluations demonstrate the robustness of our method over other state-of-the-art methods on several challenging datasets that typical SfM systems fail to estimate camera poses.