Thursday, August 28, 2025

Depth-Supervised NeRF: Fewer Views and Faster Training for Free

(CVPR 2022)

한국과학기술연구원(KIST)

CVIPL 학생연구원 **김연욱**

한국과학기술연구원
Korea Institute of Science and Technology

Depth-supervised NeRF: Fewer Views and Faster Training for Free

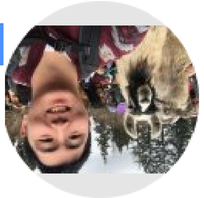
Kangle Deng[1] Andrew Liu[2] Jun-Yan Zhu[1] Deva Ramanan[1,3]
[1]Carnegie Mellon University [2]Google [3]Argo AI

Kangle Deng

Carnegie Mellon University
andrew.cmu.edu의 이메일 확인됨 - 홈페이지
Computer Vision

팔로우

제목	인용	연도
Depth-supervised nerf: Fewer views and faster training for free K Deng, A Liu, JY Zhu, D Ramanan Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern ...	1131	2022
IRC-GAN: Introspective Recurrent Convolutional GAN for Text-to-video Generation. K Deng, T Fei, X Huang, Y Peng International Joint Conferences on Artificial Intelligence (IJCAI) 2019	61	2019
3d-aware conditional image synthesis K Deng, G Yang, D Ramanan, JY Zhu Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern ...	50	2023



Andrew Liu

Google
google.com의 이메일 확인됨 - 홈페이지
3D Computer Vision Generative Models

팔로우

제목	인용	연도
Depth-supervised nerf: Fewer views and faster training for free K Deng, A Liu, JY Zhu, D Ramanan Proceedings of the IEEE/CVF conference on computer vision and pattern ...	1130	2022
Fighting fake news: Image splice detection via learned self-consistency M Huh, A Liu, A Owens, AA Efros Proceedings of the European conference on computer vision (ECCV), 101-117	573	2018
Urban radiance fields K Rematas, A Liu, PP Srinivasan, JT Barron, A Tagliasacchi, ... Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern ...	383	2022

Depth-supervised NeRF: Fewer Views and Faster Training for Free

Kangle Deng[1] Andrew Liu[2] Jun-Yan Zhu[1] Deva Ramanan[1,3]
[1]Carnegie Mellon University [2]Google [3]Argo AI

Jun-Yan Zhu

팔로우

Assistant Professor, Carnegie Mellon University
cs.cmu.edu의 이메일 확인됨 - 홈페이지
Computer Vision Computer Graphics Generative Models Computational Photography

제목	인용	연도
Unpaired image-to-image translation using cycle-consistent adversarial networks JY Zhu, T Park, P Isola, AA Efros Proceedings of the IEEE International Conference on Computer Vision	29090	2017
Image-to-image translation with conditional adversarial networks P Isola, JY Zhu, T Zhou, AA Efros Proceedings of the IEEE Conference on Computer Vision and Pattern ...	28408	2017
High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs TC Wang, MY Liu, JY Zhu, A Tao, J Kautz, B Catanzaro Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition	5669	2018

Deva Ramanan

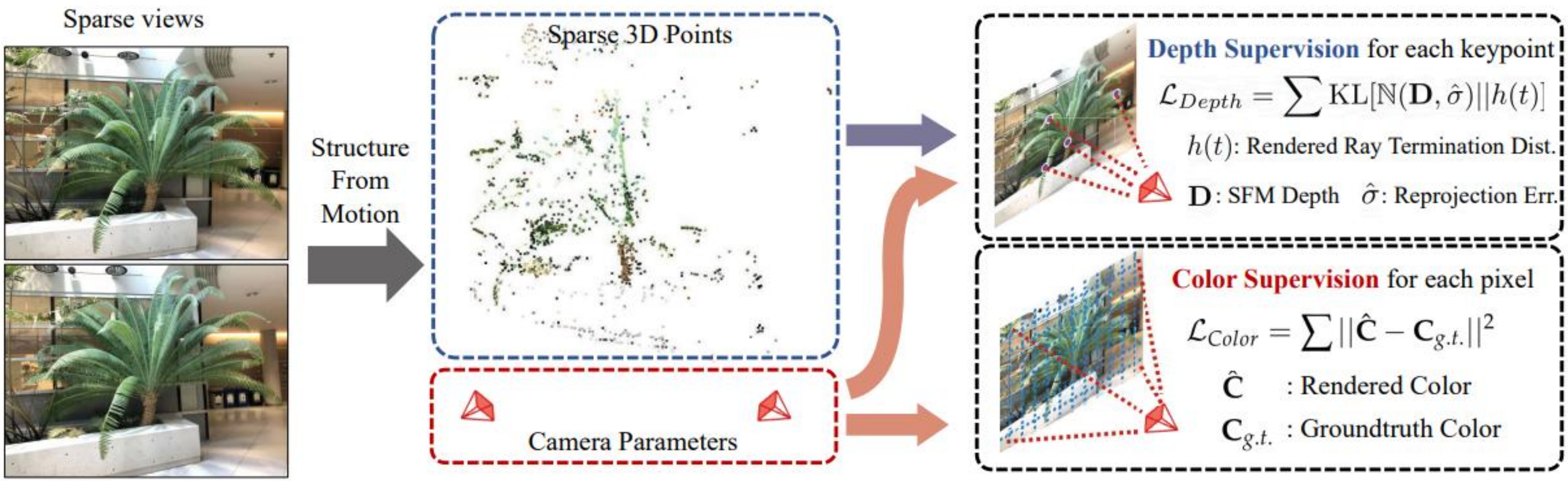
팔로우

Professor, Robotics Institute, Carnegie Mellon University
cs.cmu.edu의 이메일 확인됨 - 홈페이지
Computer Vision Machine Learning

제목	인용	연도
Microsoft coco: Common objects in context TY Lin, M Maire, S Belongie, J Hays, P Perona, D Ramanan, P Dollár, ... European conference on computer vision, 740-755	62768	2014
Object detection with discriminatively trained part-based models PF Felzenszwalb, RB Girshick, D McAllester, D Ramanan IEEE transactions on pattern analysis and machine intelligence 32 (9), 1627-1645	13612	2009

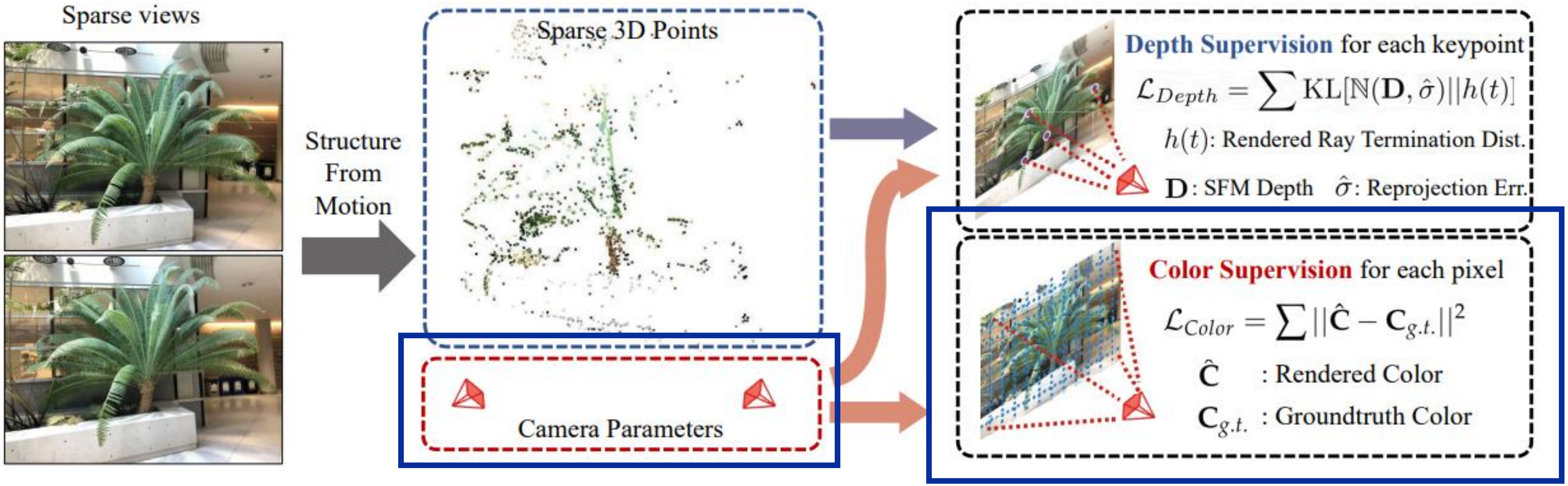
Depth-supervised NeRF

Overview



Depth-supervised NeRF

Overview



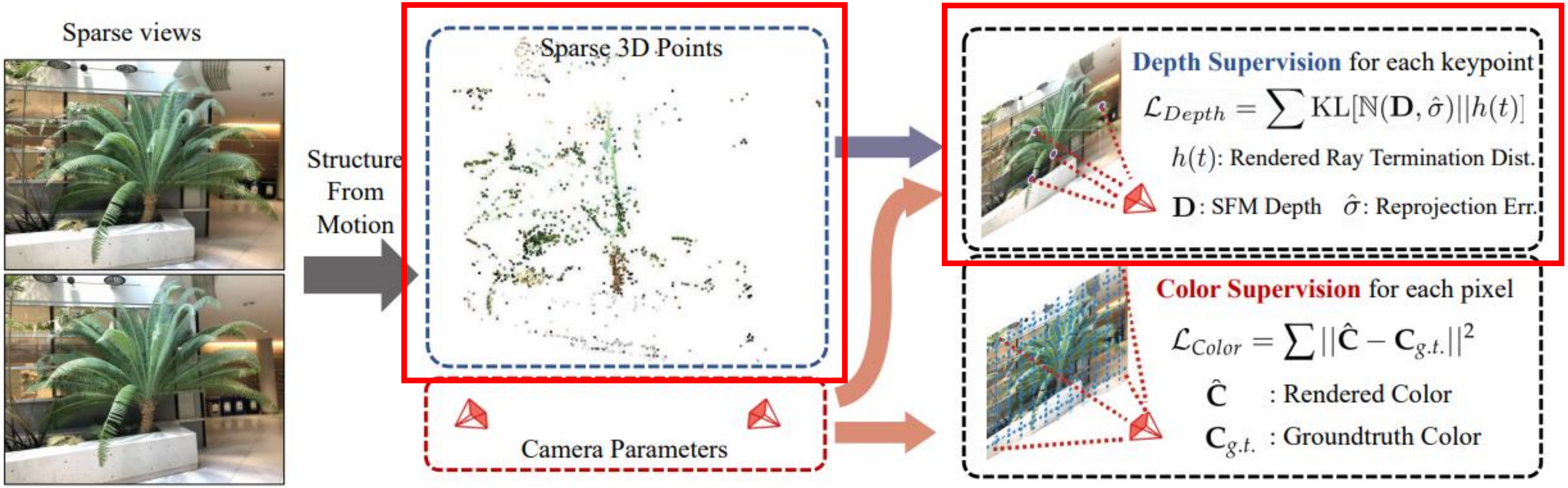
Vanilla NeRF

Depth-supervised NeRF

Overview

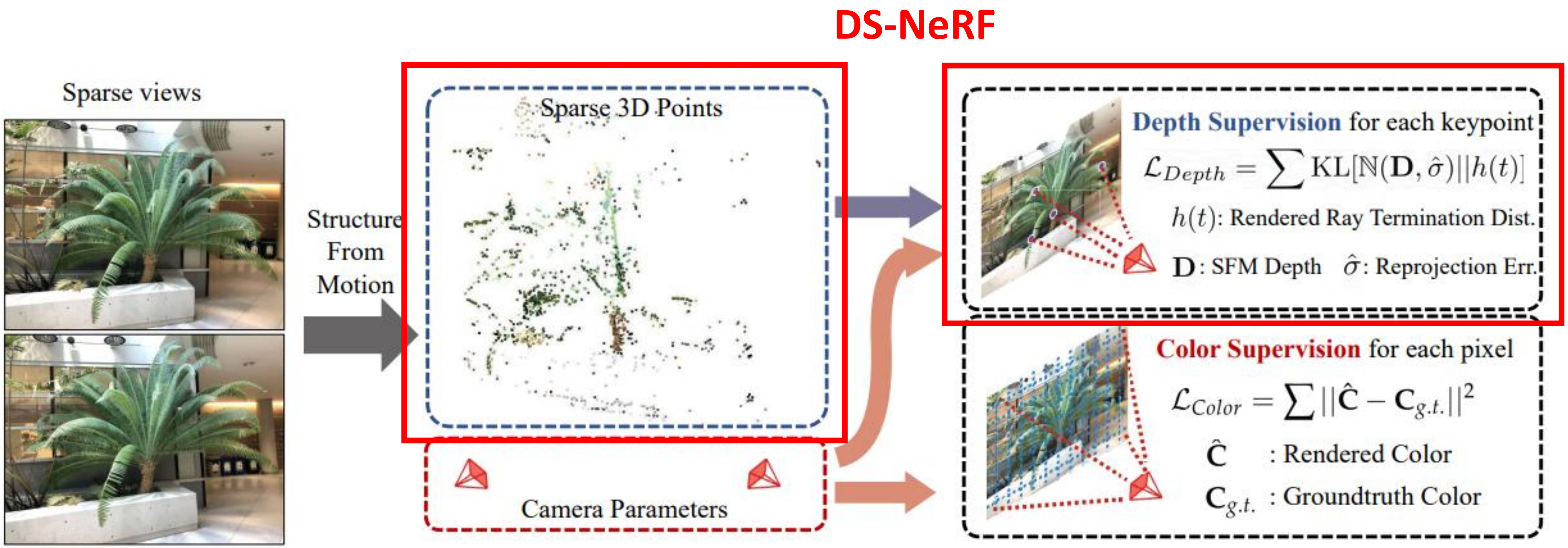
DS-NeRF

adding L_depth



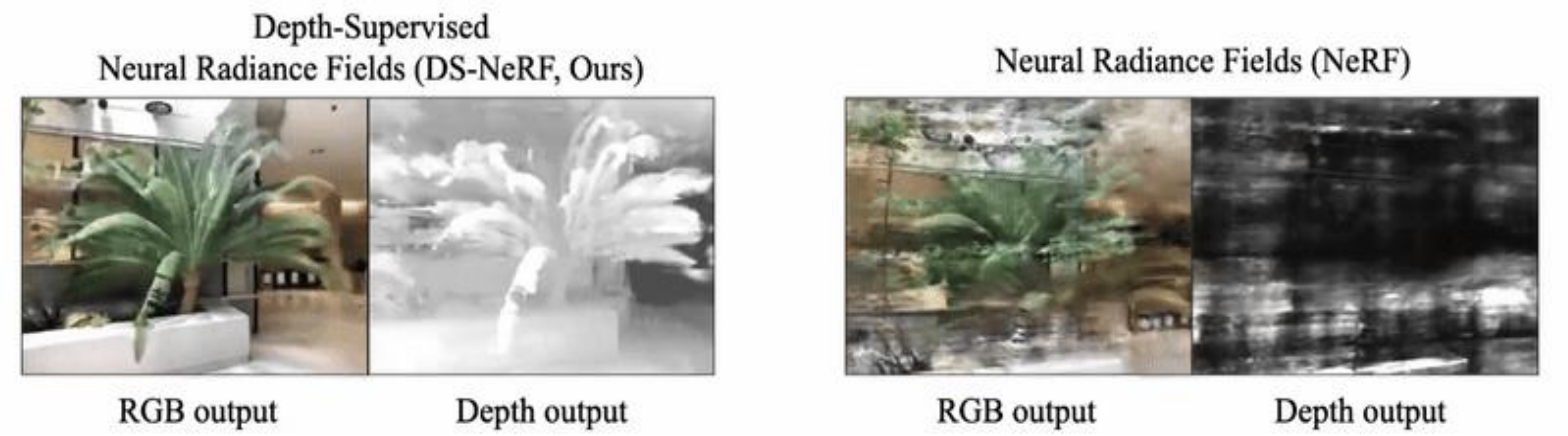
Depth-supervised NeRF

Overview



=> Fewer input views, Faster training time for Free

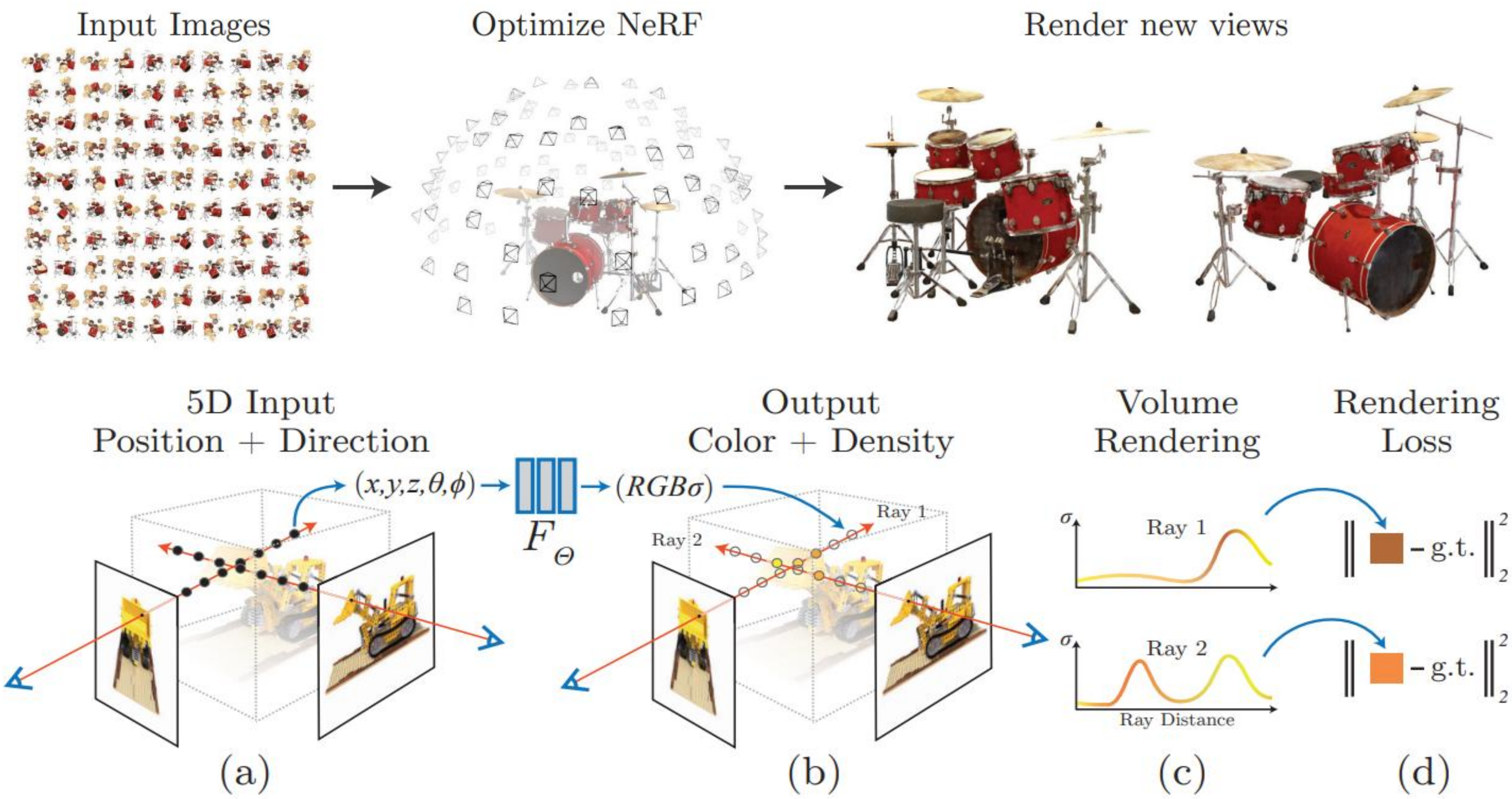
Depth-supervised NeRF



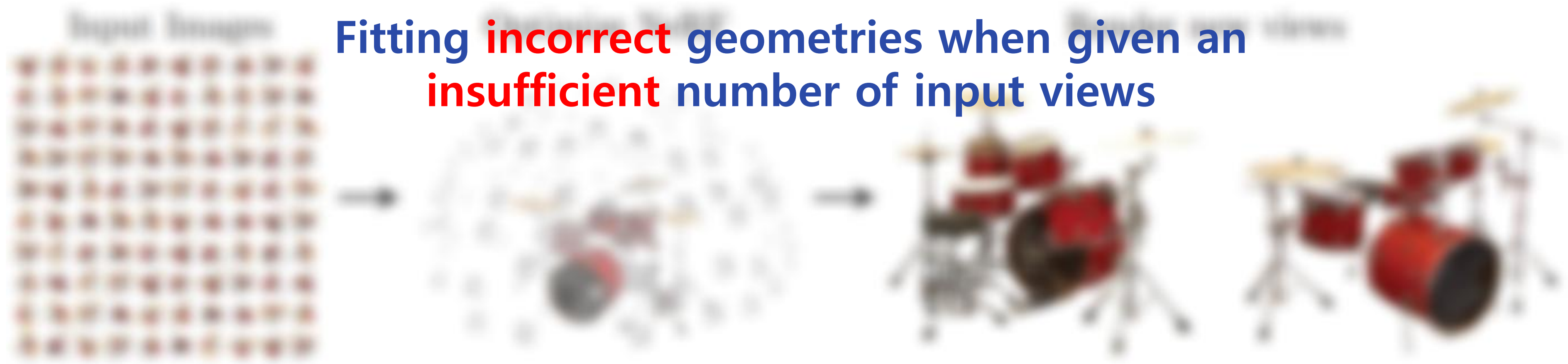
DS-NeRF vs NeRF when two input views

Motivation

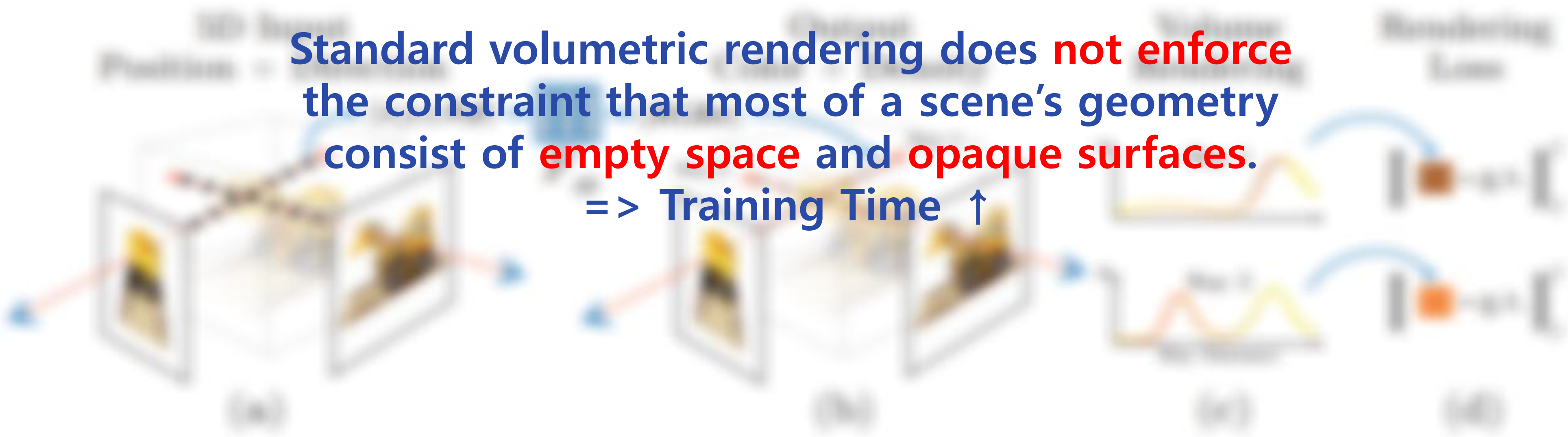
Vanilla NeRF's Problem



Vanilla NeRF's Problem



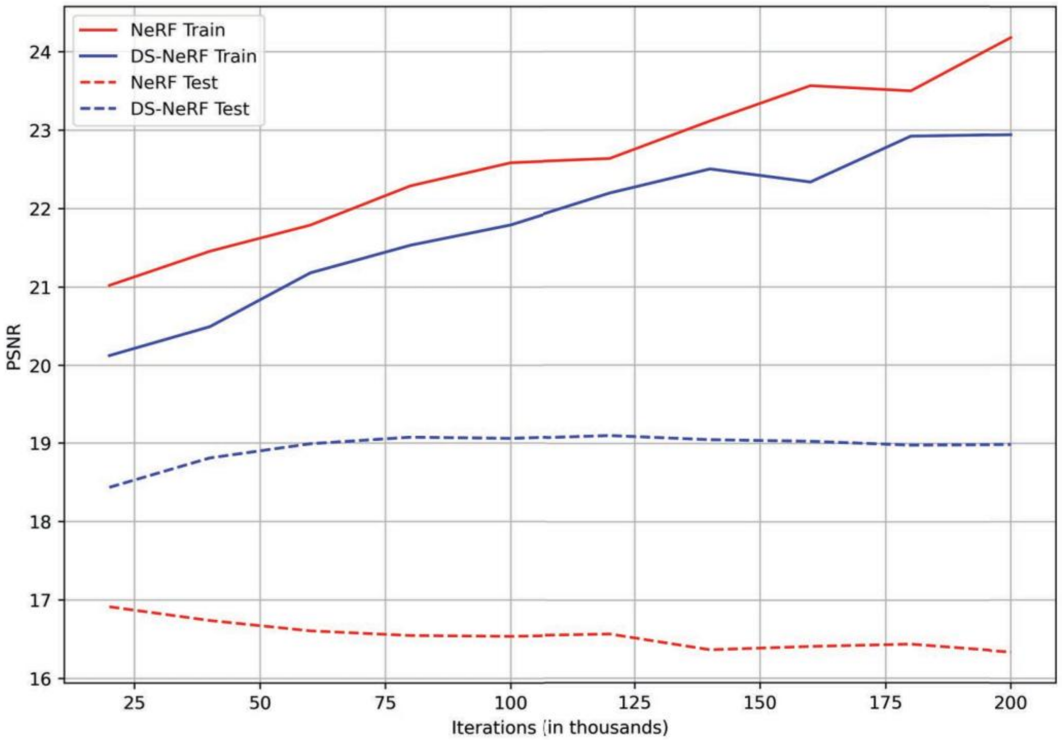
Fitting incorrect geometries when given an insufficient number of input views

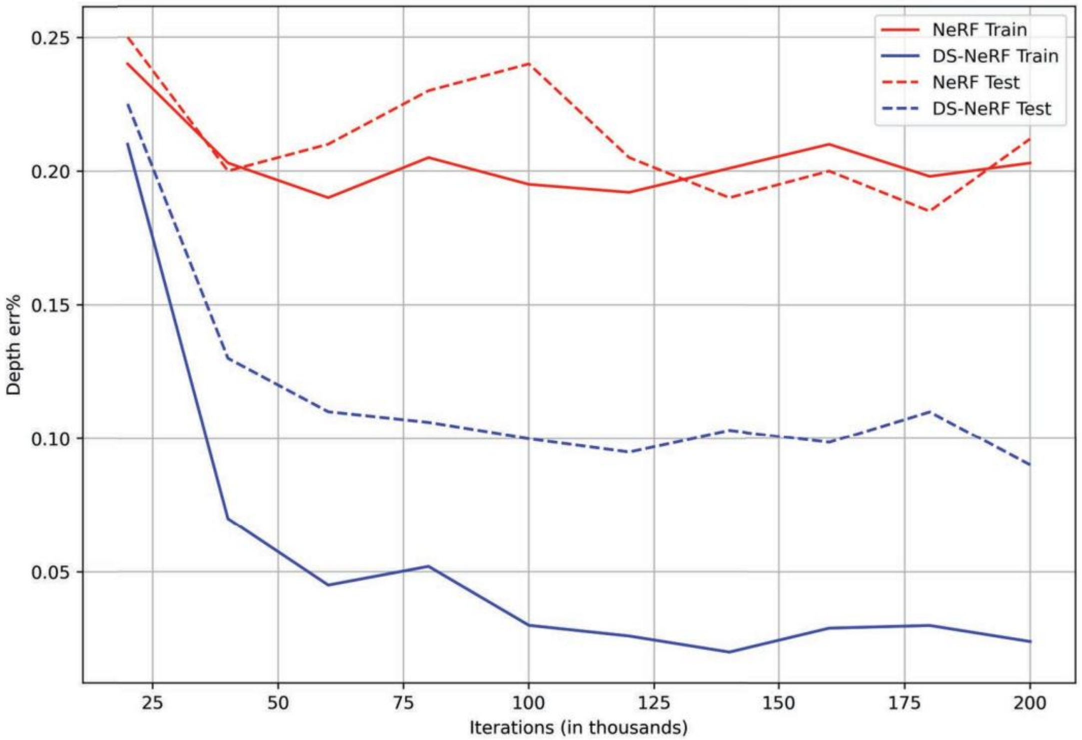


Standard volumetric rendering does not enforce the constraint that most of a scene's geometry consist of empty space and opaque surfaces.

=> Training Time ↑

Vanilla NeRF's Problem





NeRF is susceptible to overfitting when given few training views.

Structure From Motion (COLMAP)

COLMAP Output

cameras.txt

This file contains the intrinsic parameters of all reconstructed cameras in the dataset using one line per camera, e.g.:

```
# Camera list with one line of data per camera:
#   CAMERA_ID, MODEL, WIDTH, HEIGHT, PARAMS[]
# Number of cameras: 3
1 SIMPLE_PINHOLE 3072 2304 2559.81 1536 1152
2 PINHOLE 3072 2304 2560.56 2560.56 1536 1152
3 SIMPLE_RADIAL 3072 2304 2559.69 1536 1152 -0.0218531
```

points3D.txt

This file contains the information of all reconstructed 3D points in the dataset using one line per point, e.g.:

```
# 3D point list with one line of data per point:
#   POINT3D_ID, X, Y, Z, R, G, B, ERROR, TRACK[] as (IMAGE_ID, POINT2D_IDX)
# Number of points: 3, mean track length: 3.3334
63390 1.67241 0.292931 0.609726 115 121 122 1.33927 16 6542 15 7345 6 6714 14 7227
63376 2.01848 0.108877 -0.0260841 102 209 250 1.73449 16 6519 15 7322 14 7212 8 3991
63371 1.71102 0.28566 0.53475 245 251 249 0.612829 118 4140 117 4473
```

images.txt

This file contains the pose and keypoints of all reconstructed images in the dataset using two lines per image, e.g.:

```
# Image list with two lines of data per image:
#   IMAGE_ID, QW, QX, QY, QZ, TX, TY, TZ, CAMERA_ID, NAME
#   POINTS2D[] as (X, Y, POINT3D_ID)
# Number of images: 2, mean observations per image: 2
1 0.851773 0.0165051 0.503764 -0.142941 -0.737434 1.02973 3.74354 1 P1180141.JPG
2362.39 248.498 58396 1784.7 268.254 59027 1784.7 268.254 -1
2 0.851773 0.0165051 0.503764 -0.142941 -0.737434 1.02973 3.74354 1 P1180142.JPG
1190.83 663.957 23056 1258.77 640.354 59070
```

rigs.txt

This file contains the configured rigs and sensors, e.g.:

```
# Rig calib list with one line of data per calib:
#   RIG_ID, NUM_SENSORS, REF_SENSOR_TYPE, REF_SENSOR_ID, SENSORS[] as (SENSOR_TYPE, SENSOR_ID, HAS_POSE,
# Number of rigs: 1
1 2 CAMERA 1 CAMERA 2 1 -0.9999701516465348 -0.0011120266840749639 -0.0075347911527510894 0.00129851258
2 1 CAMERA 3
```

Structure From Motion (COLMAP)

COLMAP Output

cameras.txt

This file contains the intrinsic parameters of all reconstructed cameras in the dataset using one line per camera, e.g.:

```
# Camera list with one line of data per camera:
#   CAMERA_ID, MODEL, WIDTH, HEIGHT, PARAMS[]
# Number of cameras: 3
1 SIMPLE_PINHOLE 3072 2304 2559.81 1536 1152
2 PINHOLE 3072 2304 2560.56 2560.56 1536 1152
3 SIMPLE_RADIAL 3072 2304 2559.69 1536 1152 -0.0218531
```

images.txt

This file contains the pose and keypoints of all reconstructed images in the dataset using two lines per image, e.g.:

```
# Image list with two lines of data per image:
#   IMAGE_ID, QW, QX, QY, QZ, TX, TY, TZ, CAMERA_ID, NAME
#   POINTS2D[] as (X, Y, POINT3D_ID)
# Number of images: 2, mean observations per image: 2
1 0.851773 0.0165051 0.503764 -0.142941 -0.737434 1.02973 3.74354 1 P1180141.JPG
2362.39 248.498 58396 1784.7 268.254 59027 1784.7 268.254 -1
2 0.851773 0.0165051 0.503764 -0.142941 -0.737434 1.02973 3.74354 1 P1180142.JPG
1190.83 663.957 23056 1258.77 640.354 59070
```

points3D.txt

This file contains the information of all reconstructed 3D points in the dataset using one line per point, e.g.:

```
# 3D point list with one line of data per point:
#   POINT3D_ID, X, Y, Z, R, G, B, ERROR, TRACK[] as (IMAGE_ID, POINT2D_IDX)
# Number of points: 3, mean track length: 3.3334
63390 1.67241 0.292931 0.609726 115 121 122 1.33927 16 6542 15 7345 6 6714 14 7227
63376 2.01848 0.108877 -0.0260841 102 209 250 1.73449 16 6519 15 7322 14 7212 8 3991
63371 1.71102 0.28566 0.53475 245 251 249 0.612829 118 4140 117 4473
```

rigs.txt

This file contains the configured rigs and sensors, e.g.:

```
# Rig calib list with one line of data per calib:
#   RIG_ID, NUM_SENSORS, REF_SENSOR_TYPE, REF_SENSOR_ID, SENSORS[] as (SENSOR_TYPE, SENSOR_ID, HAS_POSE,
# Number of rigs: 1
1 2 CAMERA 1 CAMERA 2 1 -0.9999701516465348 -0.0011120266840749639 -0.0075347911527510894 0.00129851258
2 1 CAMERA 3
```

Structure From Motion (COLMAP)

COLMAP Output

cameras.txt

This file contains the intrinsic parameters of all reconstructed cameras in the dataset using one line per camera, e.g.:

```
# Camera list with one line of data per camera:
#   CAMERA_ID, MODEL, WIDTH, HEIGHT, PARAMS[]
# Number of cameras: 3
1 SIMPLE_PINHOLE 3072 2304 2559.81 1536 1152
2 PINHOLE 3072 2304 2560.56 2560.56 1536 1152
3 SIMPLE_RADIAL 3072 2304 2559.69 1536 1152 -0.0218531
```

images.txt

This file contains the pose and keypoints of all reconstructed images in the dataset using two lines per image, e.g.:

```
# Image list with two lines of data per image:
#   IMAGE_ID, QW, QX, QY, QZ, TX, TY, TZ, CAMERA_ID, NAME
#   POINTS2D[] as (X, Y, POINT3D_ID)
# Number of images: 2, mean observations per image: 2
1 0.851773 0.0165051 0.503764 -0.142941 -0.737434 1.02973 3.74354 1 P1180141.JPG
2362.39 248.498 58396 1784.7 268.254 59027 1784.7 268.254 -1
2 0.851773 0.0165051 0.503764 -0.142941 -0.737434 1.02973 3.74354 1 P1180142.JPG
1190.83 663.957 23056 1258.77 640.354 59070
```

Sparse 3D Point Cloud(의 Z값 = Depth)

points3D.txt

This file contains the information of all reconstructed 3D points in the dataset using one line per point, e.g.:

재투영오차

```
# 3D point list with one line of data per point:
#   POINT3D_ID, X, Y, Z, R, G, B, ERROR, TRACK[] as (IMAGE_ID, POINT2D_IDX)
# Number of points: 3, mean track length: 3.3334
63390 1.67241 0.292931 0.609726 115 121 122 1.33927 16 6542 15 7345 6 6714 14 7227
63376 2.01848 0.108877 -0.0260841 102 209 250 1.73449 16 6519 15 7322 14 7212 8 3991
63371 1.71102 0.28566 0.53475 245 251 249 0.612829 118 4140 117 4473
```

rigs.txt

This file contains the configured rigs and sensors, e.g.:

```
# Rig calib list with one line of data per calib:
#   RIG_ID, NUM_SENSORS, REF_SENSOR_TYPE, REF_SENSOR_ID, SENSORS[] as (SENSOR_TYPE, SENSOR_ID, HAS_POSE,
# Number of rigs: 1
1 2 CAMERA 1 CAMERA 2 1 -0.9999701516465348 -0.0011120266840749639 -0.0075347911527510894 0.00129851258
2 1 CAMERA 3
```

Structure From Motion (COLMAP)

COLMAP Output

cameras.txt

This file contains the intrinsic parameters of all reconstructed cameras in the dataset using one line per camera, e.g.:

```
# Camera list with one line of data per camera:
#   CAMERA_ID, MODEL, WIDTH, HEIGHT, PARAMS[]
# Number of cameras: 3
1 SIMPLE_PINHOLE 3072 2304 2559.81 1536 1152
2 PINHOLE 3072 2304 2560.56 2560.56 1536 1152
3 SIMPLE_RADIAL 3072 2304 2559.69 1536 1152 -0.0218531
```

images.txt

This file contains the pose and keypoints of all reconstructed images in the dataset using two lines per image, e.g.:

```
# Image list with two lines of data per image:
#   IMAGE_ID, QW, QX, QY, QZ, TX, TY, TZ, CAMERA_ID, NAME
#   POINTS2D[] as (X, Y, POINT3D_ID)
# Number of images: 2, mean observations per image: 2
1 0.851773 0.0165051 0.503764 -0.142941 -0.737434 1.02973 3.74354 1 P1180141.JPG
2362.39 248.498 58396 1784.7 268.254 59027 1784.7 268.254 -1
2 0.851773 0.0165051 0.503764 -0.142941 -0.737434 1.02973 3.74354 1 P1180142.JPG
1190.83 663.957 23056 1258.77 640.354 59070
```

for Free

points3D.txt

This file contains the information of all reconstructed 3D points in the dataset using one line per point, e.g.:

```
# 3D point list with one line of data per point:
#   POINT3D_ID, X, Y, Z, R, G, B, ERROR, TRACK[] as (IMAGE_ID, POINT2D_IDX)
# Number of points: 3, mean track length: 3.3334
63390 1.67241 0.292931 0.609726 115 121 122 1.33927 16 6542 15 7345 6 6714 14 7227
63376 2.01848 0.108877 -0.0260841 102 209 250 1.73449 16 6519 15 7322 14 7212 8 3991
63371 1.71102 0.28566 0.53475 245 251 249 0.612829 118 4140 117 4473
```

rigs.txt

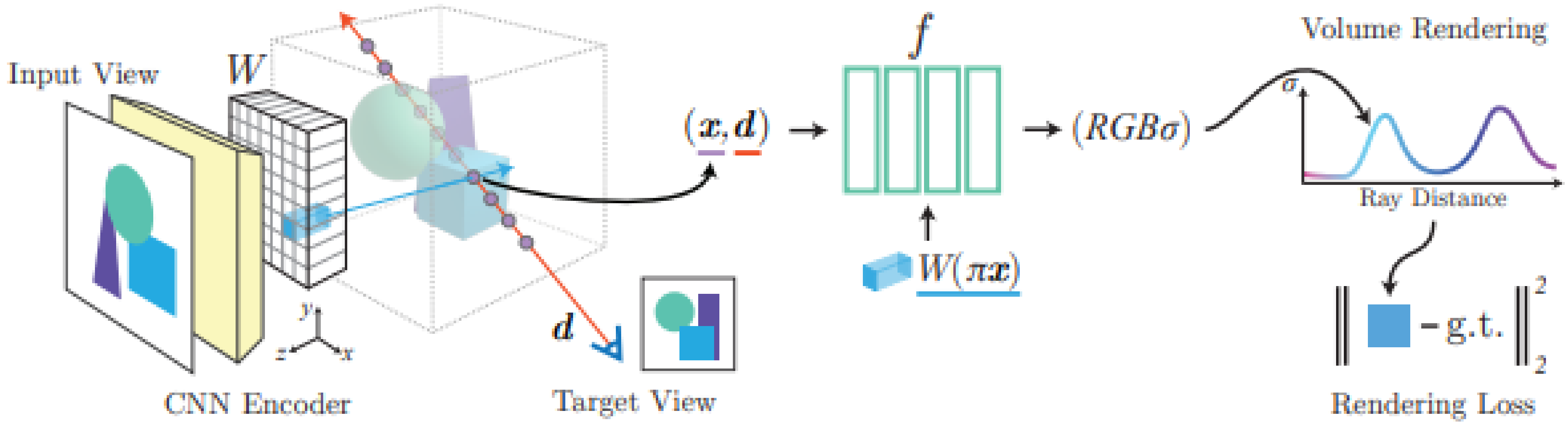
This file contains the configured rigs and sensors, e.g.:

```
# Rig calib list with one line of data per calib:
#   RIG_ID, NUM_SENSORS, REF_SENSOR_TYPE, REF_SENSOR_ID, SENSORS[] as (SENSOR_TYPE, SENSOR_ID, HAS_POSE,
# Number of rigs: 1
1 2 CAMERA 1 CAMERA 2 1 -0.9999701516465348 -0.0011120266840749639 -0.0075347911527510894 0.00129851258
2 1 CAMERA 3
```

Related Work

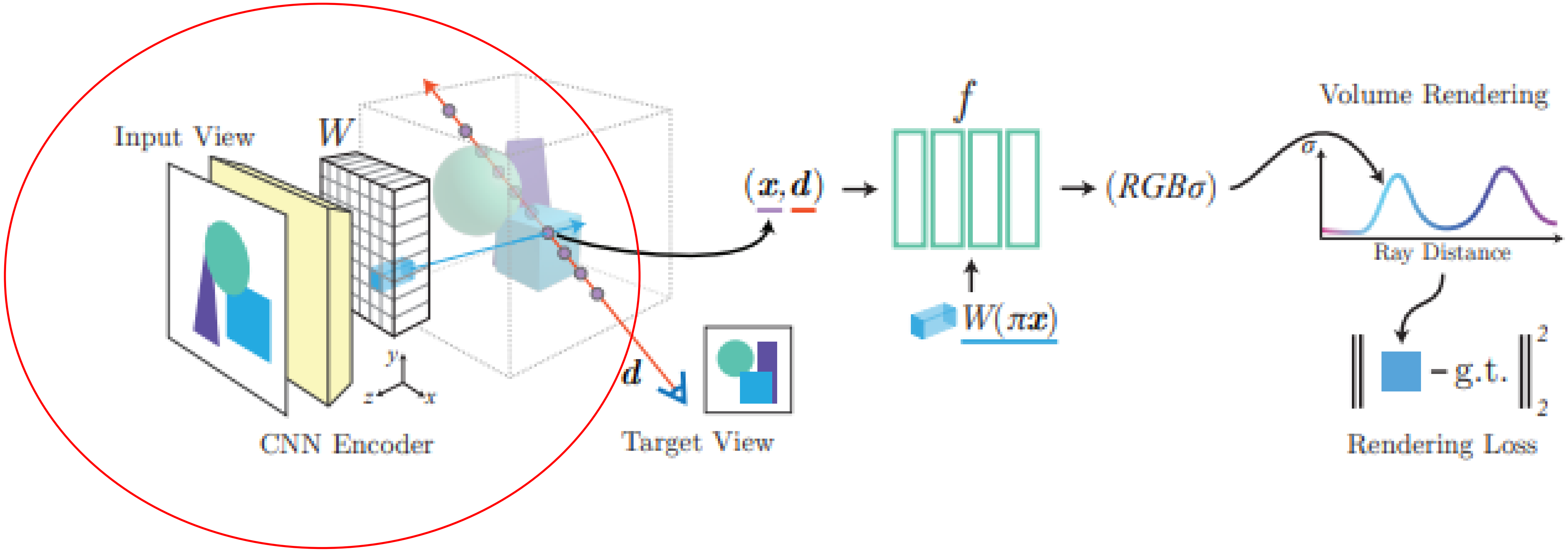
Related Work

PixelNeRF



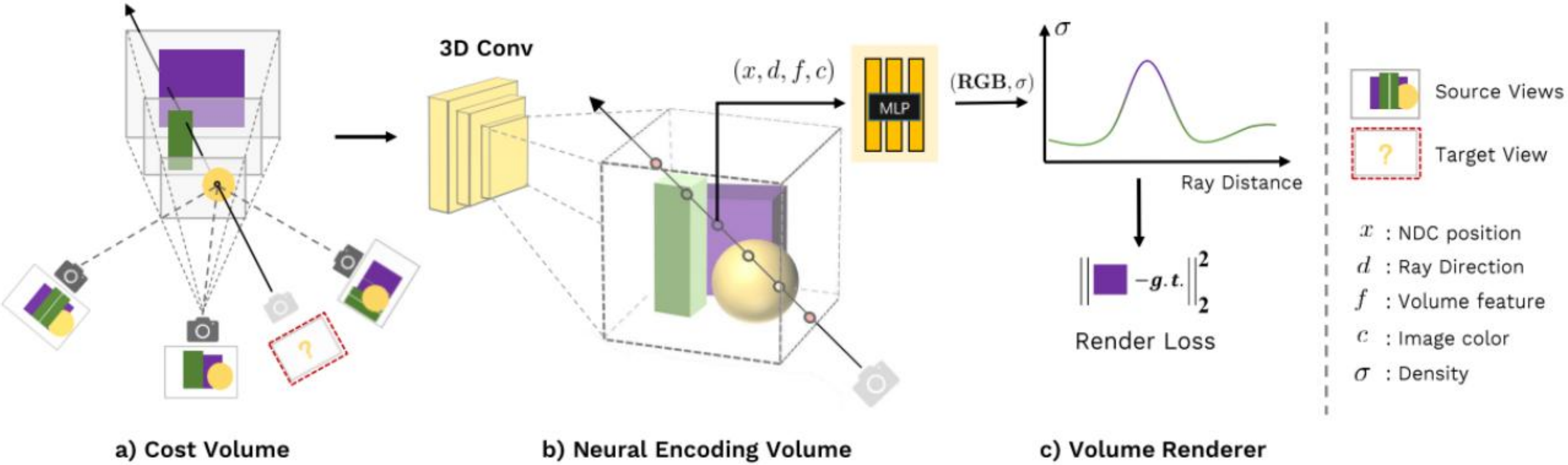
Related Work

PixelNeRF



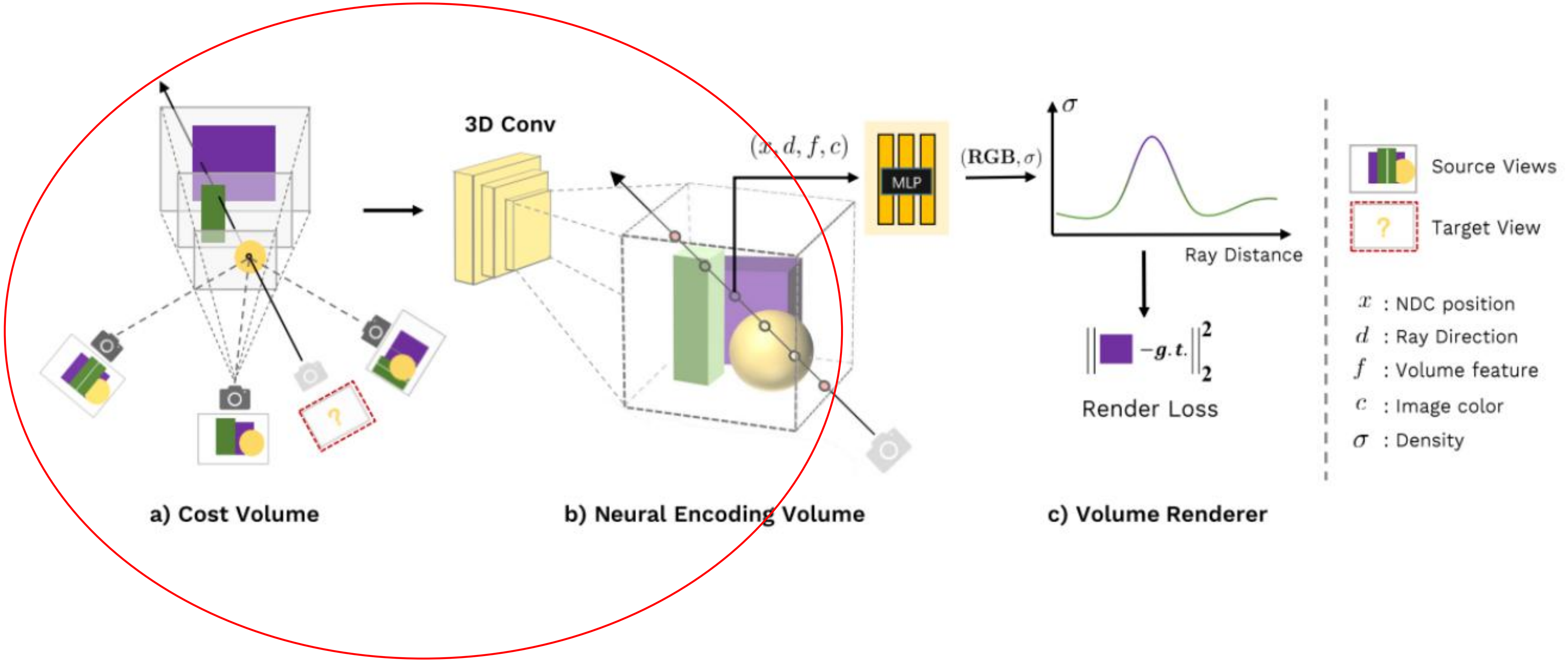
Related Work

MVS-NeRF



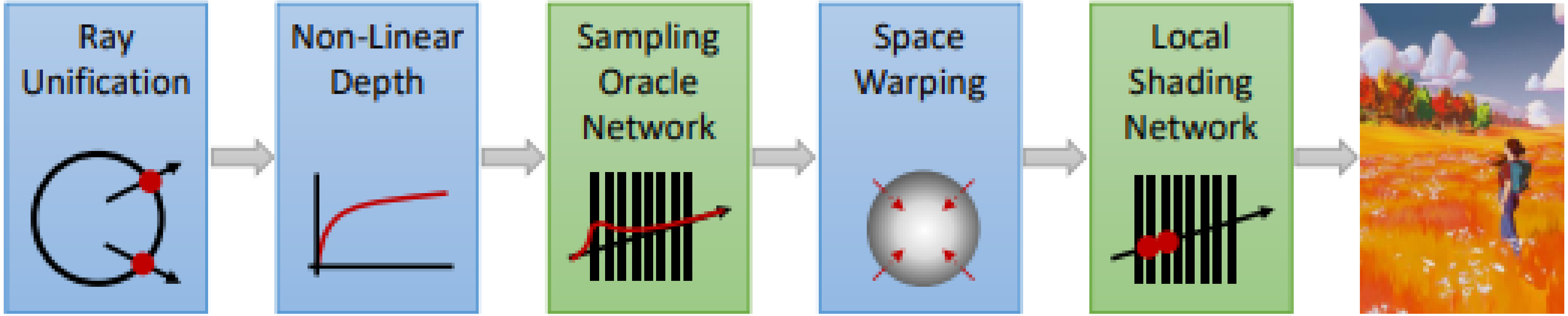
Related Work

MVS-NeRF



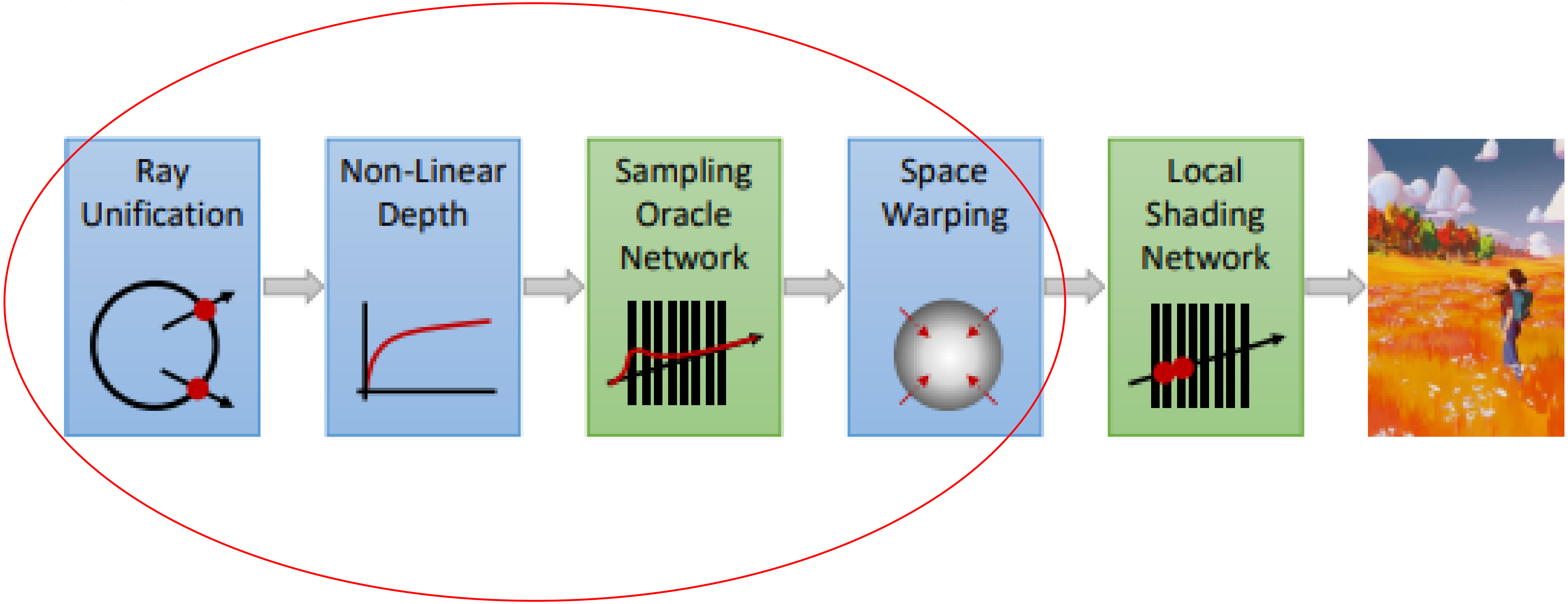
Related Work

DONeRF



Related Work

DONeRF



Related Work

These are not Free.

But, DS-NeRF is Free.

Depth-Supervised Ray Termination

Depth-Supervised Ray Termination

- DS-NeRF does not propose a new network architecture.
- It introduces an additional form of supervision to make the training of existing NeRF models more effective.
- Just add a new L_depth to the original Loss function.

=> It can be applied to other NeRFs

$$\mathcal{L} = \mathcal{L}_{\text{Color}} + \lambda_D \mathcal{L}_{\text{Depth}}$$

Depth-Supervised Ray Termination

$$\mathcal{L} = \underline{\mathcal{L}_{\text{Color}}} + \lambda_D \mathcal{L}_{\text{Depth}}$$

Same as NeRF

$$\hat{\mathbf{C}} = \int_0^\infty T(t)\sigma(t)\mathbf{c}(t)dt,$$

$$\hat{\mathbf{C}} = \int_0^\infty h(t)\mathbf{c}(t)dt = \mathbb{E}_{h(t)}[\mathbf{c}(t)].$$

$$\mathcal{L}_{\text{Color}} = \mathbb{E}_{\mathbf{r}\in\mathcal{R}(\mathbf{P})} \left\| \hat{\mathbf{C}}(\mathbf{r}) - \mathbf{C}(\mathbf{r}) \right\|_2^2 .$$

Depth-Supervised Ray Termination

NeRF Loss

$$\mathcal{L} = \sum_{\mathbf{r} \in \mathcal{R}} \left[\left\| \hat{C}_c(\mathbf{r}) - C(\mathbf{r}) \right\|_2^2 + \left\| \hat{C}_f(\mathbf{r}) - C(\mathbf{r}) \right\|_2^2 \right]$$

Some Difference

DS-NeRF Color_Loss

$$\mathcal{L}_{\text{Color}} = \mathbb{E}_{\mathbf{r} \in \mathcal{R}(\mathbf{P})} \left\| \hat{\mathbf{C}}(\mathbf{r}) - \mathbf{C}(\mathbf{r}) \right\|_2^2 .$$

Depth-Supervised Ray Termination

NeRF Loss

$$\mathcal{L} = \sum_{\mathbf{r}\in\mathcal{R}} \left[\underline{\left\| \hat{C}_c(\mathbf{r}) - C(\mathbf{r}) \right\|_2^2 + \left\| \hat{C}_f(\mathbf{r}) - C(\mathbf{r}) \right\|_2^2} \right]$$

SUM

=

DS-NeRF Color_Loss

$$\mathcal{L}_{\text{Color}} = \mathbb{E}_{\mathbf{r}\in\mathcal{R}(\mathbf{P})} \underline{\left\| \hat{\mathbf{C}}(\mathbf{r}) - \mathbf{C}(\mathbf{r}) \right\|_2^2} .$$

Average

Depth-Supervised Ray Termination

hyper-parameter

$$\mathcal{L} = \mathcal{L}_{\mathrm{Color}} + \boxed{\lambda_D} \underline{\mathcal{L}_{\mathrm{Depth}}}$$

our objective is to minimize the KL divergence between the rendered ray distribution $h_{ij}(t)$ of $\mathbf{x}_i$'s image coordinates and the noisy depth distribution:

Depth-Supervised Ray Termination

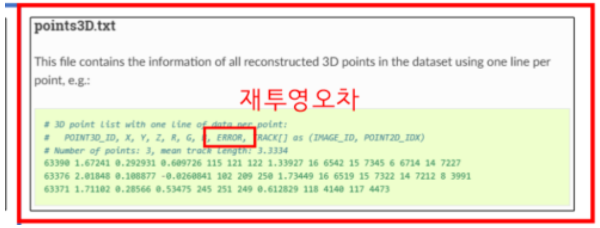
$$\mathcal{L} = \mathcal{L}_{\text{Color}} + \lambda_D \mathcal{L}_{\text{Depth}}$$

our objective is to minimize the KL divergence between the rendered ray distribution $h_{ij}(t)$ of $\mathbf{x}_i$'s image coordinates and the noisy depth distribution:

$$h(t) = T(t)\sigma(t)$$

$$\mathbb{N}(\mathbf{D}_{ij}, \hat{\sigma}_i)$$

Sparse 3D Point Cloud(의 Z값 = Depth)



points3D.txt

This file contains the information of all reconstructed 3D points in the dataset using one line per point, e.g.:

재투영오차

```
# 3D point list with one line of data per point:
#   POINT3D_ID, X, Y, Z, R, G, B, ERROR, TRACK[] as (IMAGE_ID, POINT2D_IDX)
# Number of points: 3, mean track length: 3.3334
63390 1.67241 0.292931 0.609726 115 121 122 1.33927 16 6542 15 7345 6 6714 14 7227
63376 2.01848 0.108877 -0.0260841 102 209 250 1.73449 16 6519 15 7322 14 7212 8 3991
63371 1.71102 0.28566 0.53475 245 251 249 0.612829 118 4140 117 4473
```

Depth-Supervised Ray Termination

Dirac delta

$$\mathbb{E}_{\mathbb{D}_{ij}} \mathbf{KL}[\delta(t - \mathbb{D}_{ij})||h_{ij}(t)] = \mathbf{KL}[\mathbb{N}(\mathbf{D}_{ij}, \hat{\sigma}_i)||h_{ij}(t)] + const.$$

i : 3D KeyPoint Number
j: Camera(image) Number

Using definition of KL Divergence

$$\mathrm{KL}[p||q] = \int p(x) \log \frac{p(x)}{q(x)} dx$$

$$\begin{aligned} \mathcal{L}_{Depth} &= \mathbb{E}_{x_i \in X_j} \left[-\int \log h(t) \exp\left(-\frac{(t - \mathbf{D}_{ij})^2}{2\hat{\sigma}_i^2}\right) dt \right] \\ &\approx \mathbb{E}_{x_i \in X_j} \left[-\sum_k \log h_k \exp\left(-\frac{(t_k - \mathbf{D}_{ij})^2}{2\hat{\sigma}_i^2}\right) \Delta t_k \right]. \end{aligned}$$

Experiment

Experiment

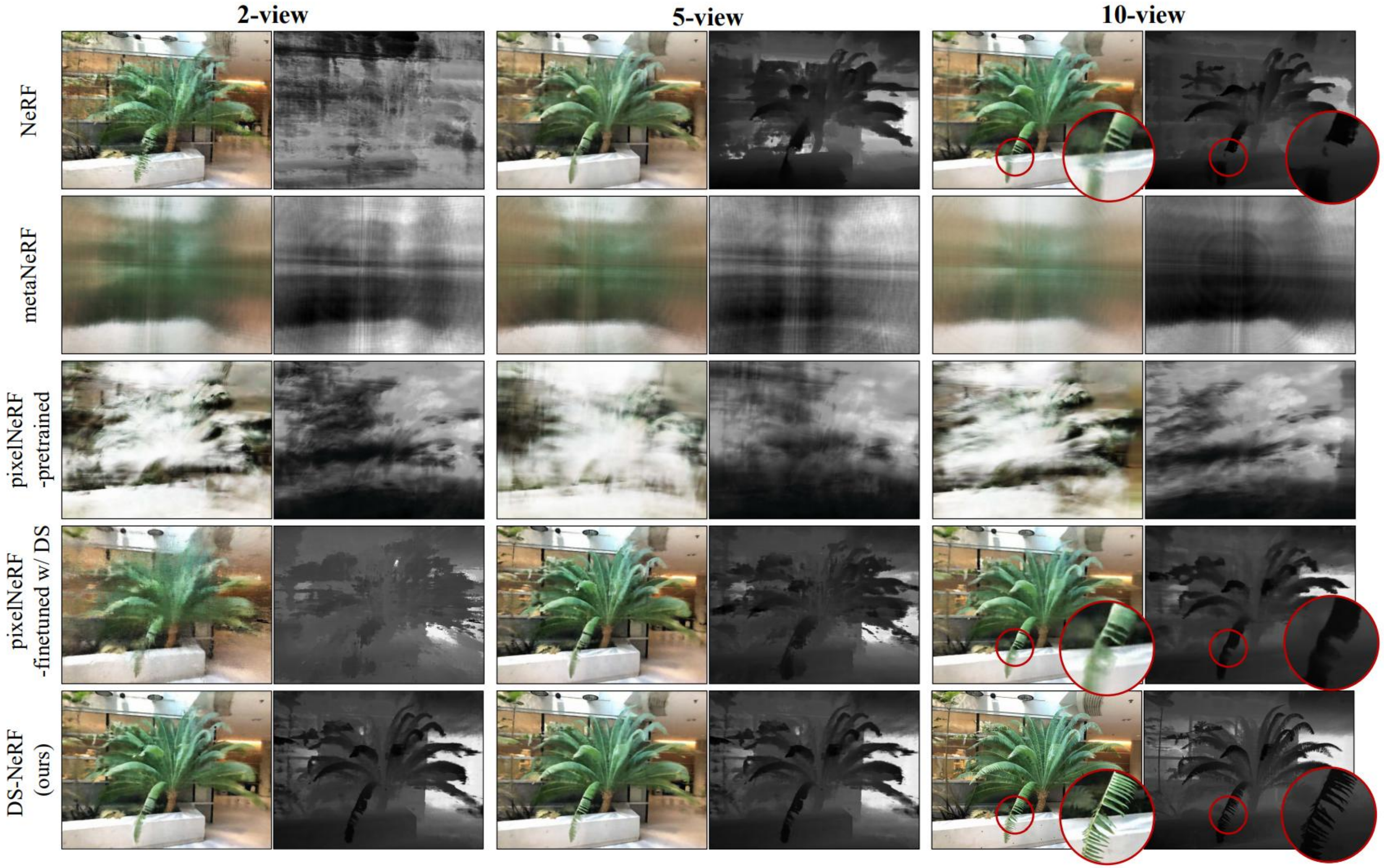
DataSet

DTU MVS Dataset

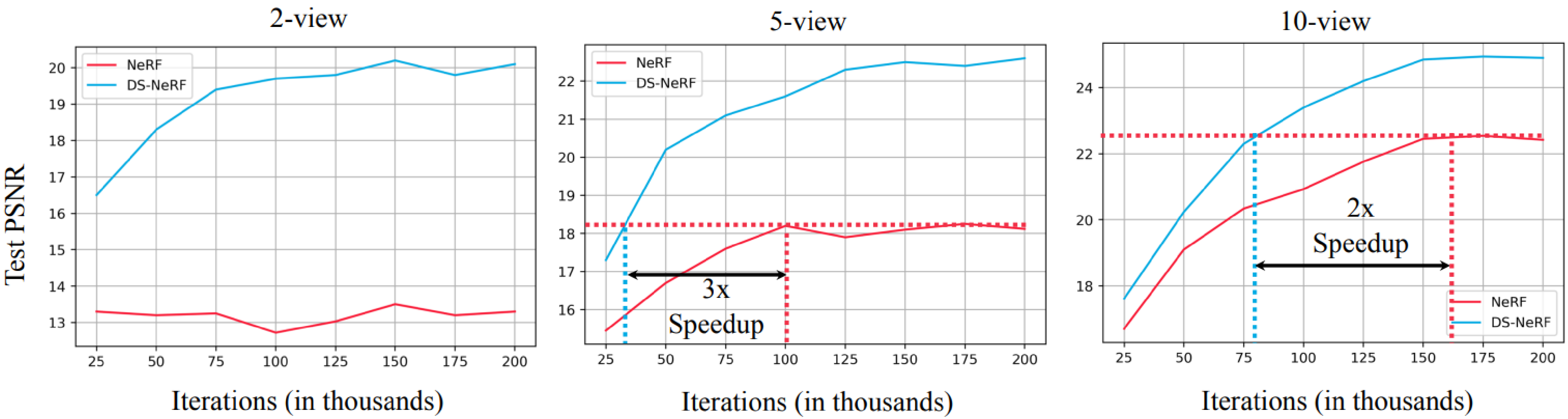
NeRF Real-world Data

Redwood-3dscan

Experiment


2-view
5-view
10-view
NeRF
metaNeRF
pixelNeRF
-pretrained
pixelNeRF
-finetuned w/ DS
DS-NeRF
(ours)

Experiment



2~3 times faster

Experiment

DTU [8]	PSNR↑			SSIM↑			LPIPS↓		
	3-view	6-view	9-view	3-view	6-view	9-view	3-view	6-view	9-view
NeRF	9.9	18.6	22.1	0.37	0.72	**0.82**	0.62	0.35	0.26
metaNeRF-DTU	18.2	18.8	20.2	0.60	0.61	0.67	0.40	0.41	0.35
pixelNeRF-DTU	**19.3**	20.4	21.1	**0.70**	0.73	0.76	**0.39**	0.36	0.34
DS-NeRF									
MSE	16.5	20.5	22.2	0.54	0.73	0.77	0.48	0.31	0.26
KL divergence	16.9	**20.6**	**22.3**	0.57	**0.75**	0.81	0.45	**0.29**	**0.24**

Contribution

- Addressing NeRF's Limitations: Fewer Views & Faster Training
- Proposing a Depth Supervision Loss for Free
- High Compatibility