MVSNeRF: Fast Generalizable Radiance Field Reconstruction from Multi-View Stereo

ICCV 2021

Anpei Chen
Zexiang Xu
Fuqiang Zhao

29. September 2021

ShanghaiTech University
Adobe Research
University of California, San Diego

Outline

Introduction

- Original NeRF
- Research topic for improving NeRF
- Generalization NeRF to use less input

Method

- MVSNeRF
  - Planar Homography
  - Wrong formula in paper
  - Fine tuning technique
- PixelNeRF
- IBRNet

Evaluation

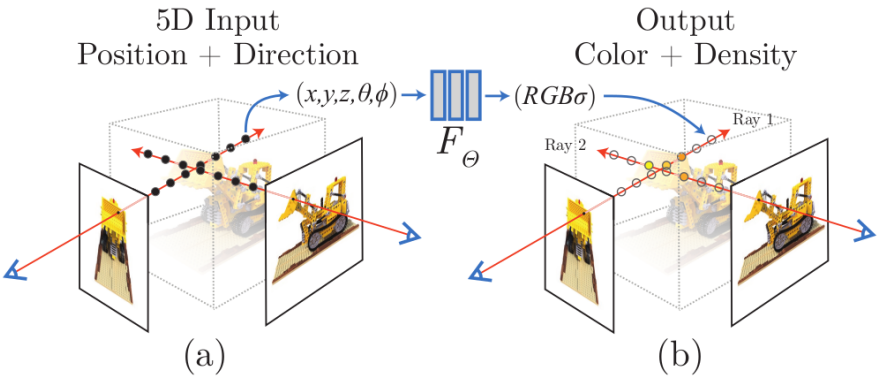
- Some helpful materials

Introduction

Original NeRF

Representing Scenes as Neural Radiance Fields for View Synthesis, Mildenhall et al, ECCV 2020(best-paper)

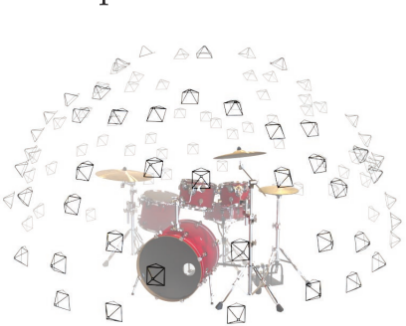
- Feeding location and viewing direction to the MLP
- Using volume rendering techniques(*) to composite these output values into an image



* Ray Tracing Volume Density, Kajiya et al, SIGGRAPH 1984

Research topic for improving NeRF

- Faster Training
- Faster Inference
- Unconstrained Images
- Deformable/Editing neural radiance
- Video Synthesis In 2D/3D
- Pose Estimation
- Lighting / Illumination / Reflectance
- Generalization ← today's topic
- Neural Field Textures



Generalization NeRF to use less input

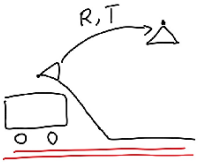
Common in MVSNeRF/PixelNeRF/IBRNet:

- Obtain the latent image features of nearby views through a feature extractor(include 2D/3D Conv) for scene understanding
- Query the colors and densities of sampled point(*) along the rays, using volume rendering techniques to composite these output values into images
  - (*)Adopt linear hierarchical sampling
- Simultaneously optimize two scaled network(fine/coarse)
- Can train on one scene first, and then get better result through fast fine-tuning

Method

Planar Homography

If the scene captured by two cameras is on the same plane, the relationship between the two cameras is called Planar Homography.



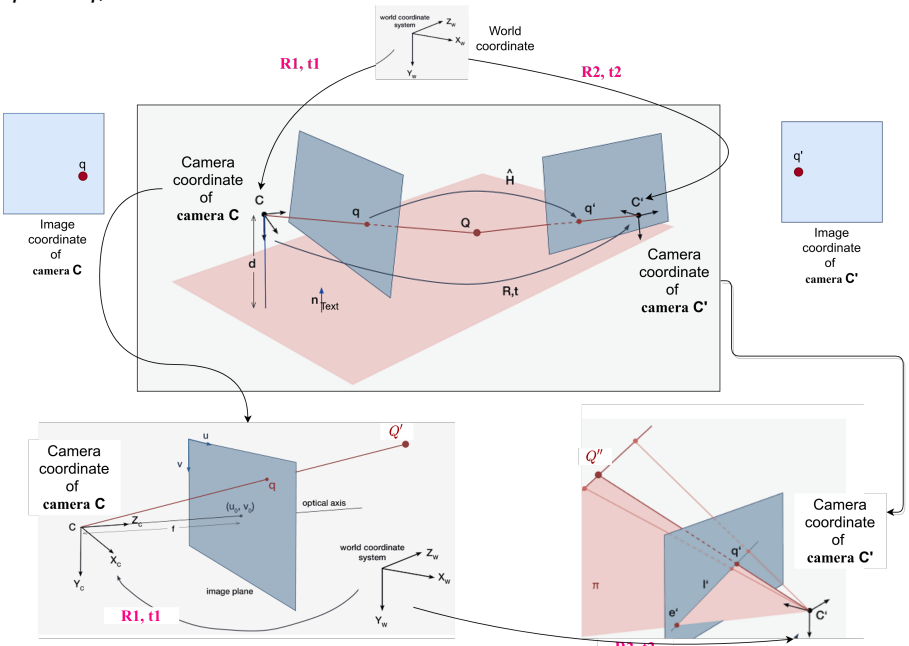




- We could derive a homography matrix H to transform the different points on the image plane of one camera to another camera's image plane

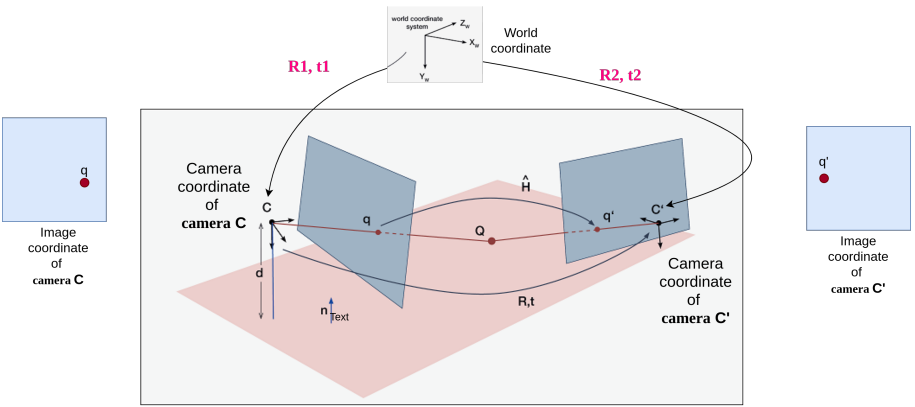
Planar Homography (cont.)

$q' = Hq$, how to derive H?



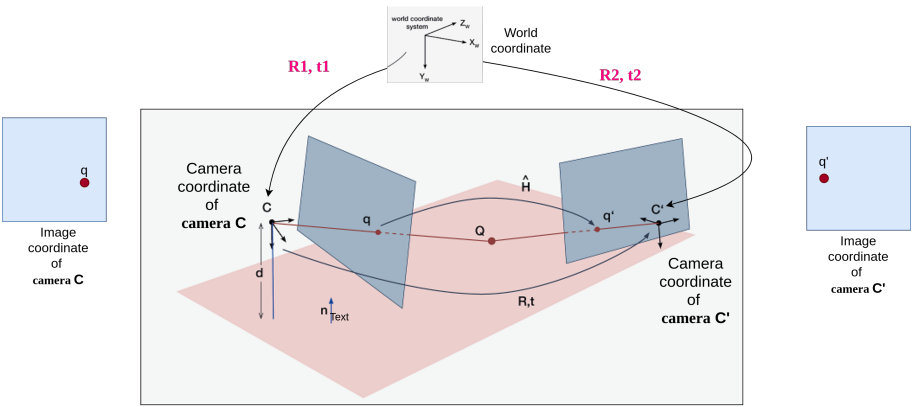
Planar Homography (cont.)

- Q': Q in camera coordinate of C (3D)
  - $q = K_1 Q'$ (K_1 : intrinsic camera matrix of camera C)
- Q'': Q in camera coordinate of C' (3D)
  - $q' = K_i Q''$ (K_i : intrinsic camera matrix of camera C')
- $Q'' = RQ' + t$ (translate C to C', R, t is actually unknown)
- The distance from camera C to the plane is $d = n^T Q' \rightarrow 1 = \frac{1}{d} n^T Q'$



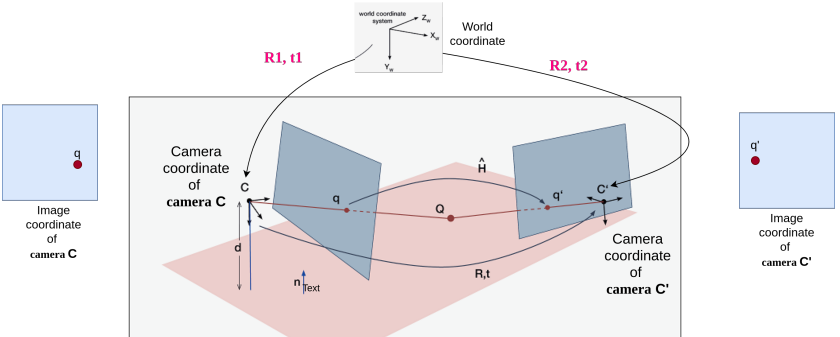
Planar Homography (cont.)

- Q': Q in camera coordinate of C (3D)
  - $q = K_1 Q'$ (K_1 : intrinsic camera matrix of camera C)
- Q'': Q in camera coordinate of C' (3D)
  - $q' = K_i Q''$ (K_i : intrinsic camera matrix of camera C')
- $Q'' = RQ' + t$ (translate C to C', R, t is actually unknown)
  $\rightarrow Q'' = RQ' + 1 \times t = RQ' + \frac{1}{d} n^T Q' t = (R + \frac{1}{d} t n^T) Q'$
- The distance from camera C to the plane is $d = n^T Q' \rightarrow 1 = \frac{1}{d} n^T Q'$



Planar Homography (cont.)

- Q': Q in camera coordinate of C (3D)
  - $q = K_1 Q'$ (K_1 : intrinsic camera matrix of camera C) $\rightarrow Q' = K_1^{-1} q$
- Q'': Q in camera coordinate of C' (3D)
  - $q' = K_i Q''$ (K_i : intrinsic camera matrix of camera C') $\rightarrow Q'' = K_i^{-1} q'$
- $Q'' = RQ' + t$ (translate C to C', R, t is actually unknown)
  $\rightarrow Q'' = RQ' + 1 \times t = RQ' + \frac{1}{d} n^T Q' t = (R + \frac{1}{d} t n^T) Q'$
  $\rightarrow Q'' = K_i^{-1} q' = (R + \frac{1}{d} t n^T) Q' = (R + \frac{1}{d} t n^T) K_1^{-1} q$
  $\rightarrow q' = K_i (R + \frac{1}{d} t n^T) K_1^{-1} q$
  $\rightarrow H = K_i (R + \frac{1}{d} t n^T) K_1^{-1}$
  (later we need to substitute R, t with known parameter R_1, t_1, R_2, t_2)



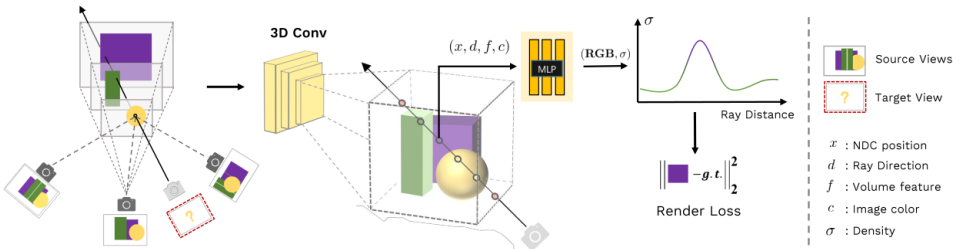
MVSNeRF

- A Novel neural rendering model for view synthesis
- Can construct neural radiance field within fewer input images
  - 3 posed images needed in implementation (compared to $\sim$ 100 densely captured images needed in original NeRF)
- Achieve original NeRF's 9.5h training result through less than 10min fine tuning on specified scene
- Leverage Multiple View Stereo (MVS in paper title) techniques and Convolutions for geometry scene reasoning



a) Source views b) MVS-NeRF no fine-tuning c) MVS-NeRF 6 min fine-tuning d) NeRF 5.1h optimization

MVSNeRF


3D Conv
(x, d, f, c)
MLP
$(\mathbf{RGB}, \sigma)$
σ
Ray Distance
$\|-g.t.\|_2^2$
Render Loss
Source Views
Target View
x : NDC position
d : Ray Direction
f : Volume feature
c : Image color
σ : Density
a) Cost Volume
b) Neural Encoding Volume
c) Volume Renderer

MVSNeRF

Step-1: Cost volume construction

- Extract image feature $F_i = T(I_i)$
- Warping feature maps of source views
  ~~$H_i(z) = K_i \cdot (R_i \cdot R_1^T + \frac{(t_1 - t_i) \cdot n_1^T}{z}) \cdot K_1^{-1}$~~
  (I don't know how author derived this formula)
  $H_i(z) = K_i \cdot (R_i \cdot R_1^T + \frac{-R_1^T t_1 + R_i^T t_i}{z} n_1^T) \cdot K_1^{-1}$
  (BTW, releasing code match correct formula above)

How to derive homographic warping formula in section 3.1 #23

Closed **murumura** opened this issue 6 days ago · 4 comments

murumura commented 6 days ago

Hi, thank you for sharing the cool work.
I am wondering how to derive the matrix warping formula in section 3.1 i.e

$$\mathcal{H}_i(z) = K_i \cdot (R_i \cdot R_1^T + \frac{(t_1 - t_i) \cdot n_1^T}{z}) \cdot K_1^{-1}$$

Because according to my own derivation, the result is

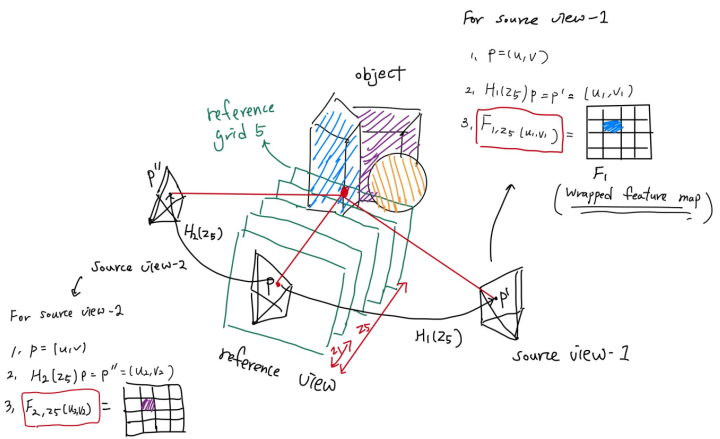
$$H_i(z) = K_i \cdot (R_i \cdot R_1^T + \frac{-R_1^T t_1 + R_i^T t_i}{z} n_1^T) \cdot K_1^{-1}$$

Are there any derivation premises that I have not considered? Thanks in advance

MVSNeRF

Step-1: Cost volume construction

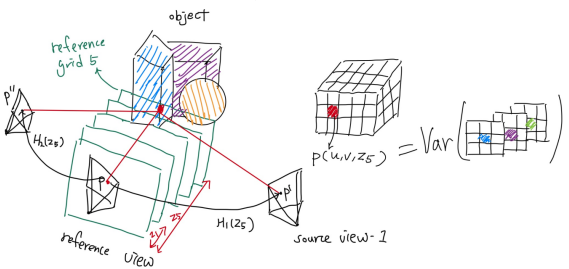
- Extract image feature $F_i = T(I_i)$ (including reference view)
- Warping feature maps of source views then project them to the feature map of reference view
  - $H_i(z) = K_i \cdot (R_i \cdot R_1^T + \frac{-R_1^T t_1 + R_i^T t_i}{z} n_1^T) \cdot K_1^{-1}$
  - $F_{i,z}(u, v) = F_i(H_i(z)[u, v, 1])$



MVSNeRF

Step-1: Cost volume construction

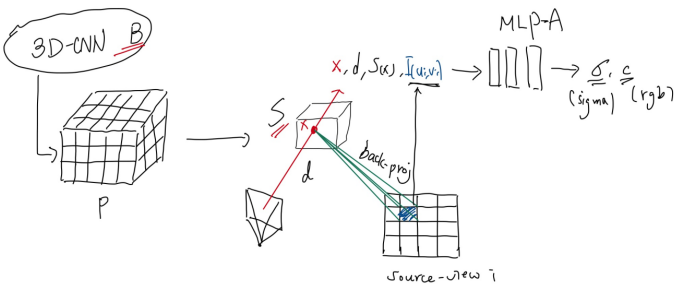
- Extract image feature $F_i = T(I_i)$ (including reference view)
- Warping feature maps of source views then project them to the feature map of reference view
  - $H_i(z) = K_i \cdot (R_i \cdot R_1^T + \frac{-R_1^T t_1 + R_i^T t_i}{z} n_1^T) \cdot K_1^{-1}$
  - $F_{i,z}(u, v) = F_i(H_i(z)[u, v, 1])$
- Build cost volume P by warped feature maps at reference view through variance-based metric
  - 'cost' in semantic denote matching cost!
  - variance-based cost volume encodes the image appearance variations across different input views
  - $P(u, v, z) = Var(F_{i,z}(u, v))$



MVSNeRF

Step-2: Radiance field reconstruction

- convert the built cost volume into a radiance field for realistic view synthesis
- Train a deep 3D CNN B to transform the built image-feature cost volume into a scene encoding volume S
  - The 3D CNN B is a 3D UNet with downsampling and upsampling convolutional layers and skip connections
- Regressing volume properties by an MLP network A through scene encoding volume S
  - $\sigma, c = A(x, d, f, I(u_i, v_i))$
  - point features $f = S(x)$



MVSNeRF

Step-3: Volume rendering and end-to-end training

- Volume rendering : pixel's color is computed by ray marching and accumulating radiance (different from original NeRF adopt!)

Rendering model for ray: $r(t) = o + td,\ t_1 \leq t \leq t_n$

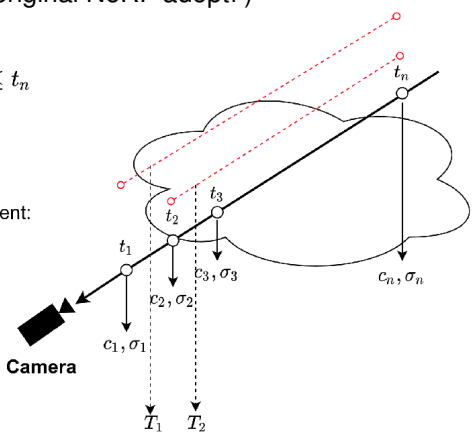
$$\hat{C}(r) = \sum_{i=1}^{N} T_i(1 - e^{(-\sigma_i)})c_i = \sum_{i=1}^{N} T_i \alpha_i c_i$$

How much light is blocked before reaching the segment:

$$T_i = e^{-\sum_{j=1}^{i-1} \sigma_j} = \prod_{j=1}^{i-1}(1 - \alpha_j)$$

Opacity of volume

$$\alpha_i = 1 - e^{(-\sigma_i)}$$



MVSNeRF

Step-3: Volume rendering and end-to-end training

- Volume rendering : pixel's color is computed by ray marching and accumulating radiance (without considering distance between sampled point while original NeRF does!)

Rendering model for ray: $r(t) = o + td,\ t_1 \leq t \leq t_n$

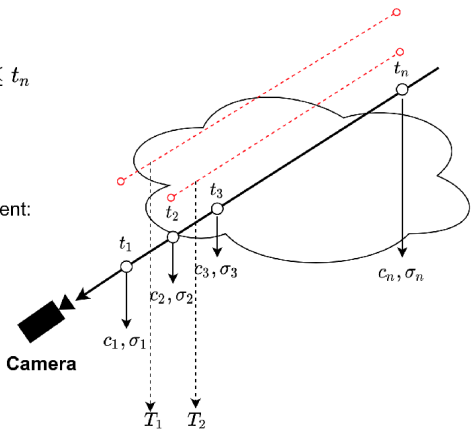
$$\hat{C}(r) = \sum_{i=1}^{N} T_i (1 - e^{(-\sigma_i)}) c_i = \sum_{i=1}^{N} T_i \alpha_i c_i$$

How much light is blocked before reaching the segment:

$$T_i = e^{-\sum_{j=1}^{i-1} \sigma_j} = \prod_{j=1}^{i-1} (1 - \alpha_j)$$

Opacity of volume

$$\alpha_i = 1 - e^{(-\sigma_i)}$$



MVSNeRF

Step-3: Volume rendering and end-to-end training

- End to End training : $\mathcal{L} = \sum_{r \in R}[||\hat{C}_{gt}(r) - C(r)||_2^2$

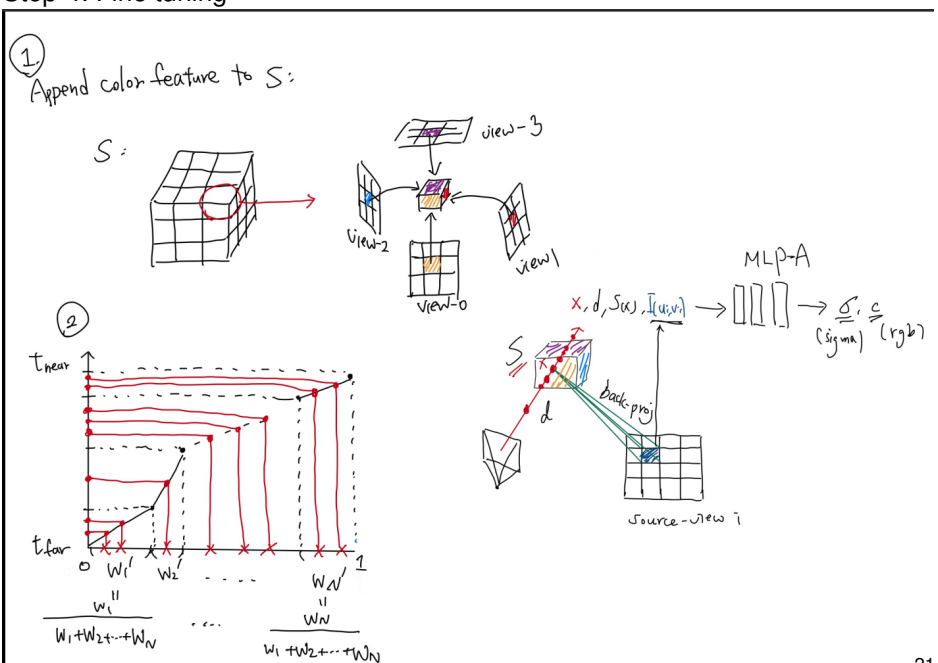
MVSNeRF

Step-4: Fine tuning

- Optimize encoded volume S
  - Build a color volume
  - Appending color volume as additional channels to the encoded volume S
- Optimize MLP A
  - Build a density feature by executing an additional MLP prediction and only retrieve sigma values
  - Using density feature as weight perform important sampling (Higher density, denser sample)
    - $w_i = T_i(1 - \exp(\sigma_i \delta_i))$
  - Volume rendering on these important points

MVSNeRF

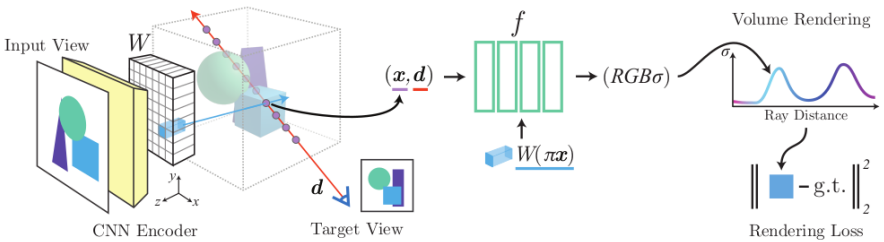
Step-4: Fine tuning



pixelNeRF: Neural Radiance Fields from One or Few Images

Alex Yu Vickie Ye Matthew Tancik Angjoo Kanazawa
UC Berkeley

PixelNeRF

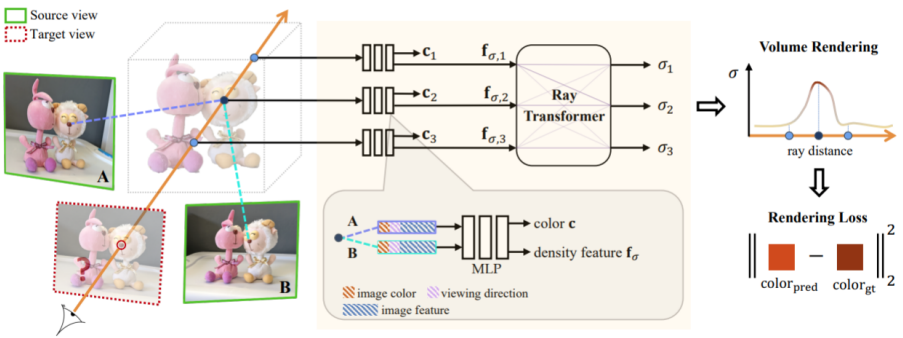

Input View
W
CNN Encoder
d
Target View
(x, d)
f
W(πx)
(RGBσ)
Volume Rendering
σ
Ray Distance
- g.t.
Rendering Loss

IBRNet: Learning Multi-View Image-Based Rendering

Qianqian Wang[1,2] Zhicheng Wang[1] Kyle Genova[1,3] Pratul Srinivasan[1] Howard Zhou[1]
Jonathan T. Barron[1] Ricardo Martin-Brualla[1] Noah Snavely[1,2] Thomas Funkhouser[1,3]

[1]Google Research [2]Cornell Tech, Cornell University [3]Princeton University

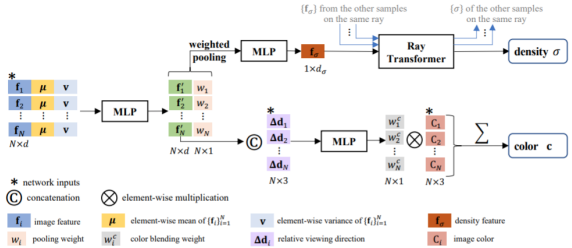
IBRNet



View selection and feature extraction

- Select working set by checking by checking that the viewing direction and the target view are similar
- Extract the image feature f_i of the source views through Residual U-Net (all source views share the same weights)

IBRNet (cont.)

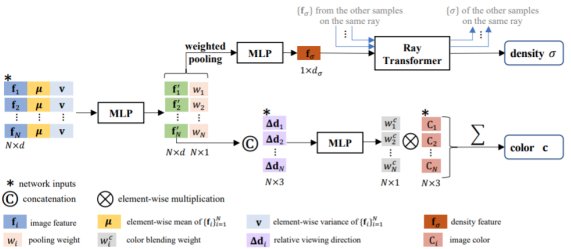


Volume density prediction

- Aggregate the multi-view features at (x, d) to obtain a density feature f_σ
  - compute multi-view aware feature $f_i^{'}$ and weight vector w_i by calculate per-element mean and variance of f_i and a shared MLP
  - density feature $f_\sigma = MLP\ (w_1 f_1^{'}, w_2 f_2^{'}, ..., w_N f_N^{'})$
  - $f_\sigma(x_1), ..., f_\sigma(x_M) = MultiHead(Q, K, V)$
    - treat sampled point $x_1, ..., x_M$ as input sequence
    - $Q/K/V = (f_\sigma, f_\sigma, f_\sigma)$

```
# [n_rays, n_samples, 16]
globalfeat, _ = self.ray_attention(globalfeat, globalfeat, globalfeat, mask=(num_valid_obs > 1).floa
sigma = self.out_geometry_fc(globalfeat)  # [n_rays, n_samples, 1]
sigma_out = sigma.masked_fill(num_valid_obs < 1, 0.)  # set the sigma of invalid point to zero
```

IBRNet (cont.)



Color prediction

- obtain color at a 5D point by predicting blending weights w_i^c for the image colors C_i
  - blending weight $w_i^c = MLP(x), x = \{f_i^{'}, d - d_i\}$

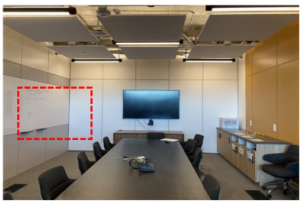
Evaluation

Depth Prediction

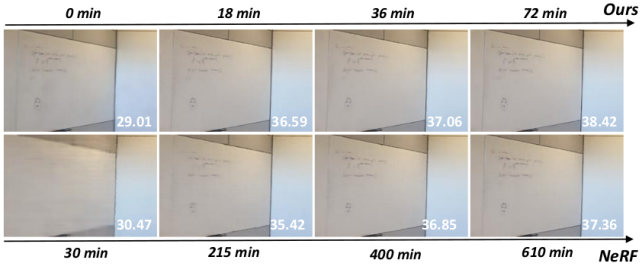
Method	Abs err↓	Acc (0.01)↑	Acc (0.05)↑
MVSNet	**0.018**/ –	0.603/ –	**0.955**/ –
PixelNeRF	0.245/0.239	0.037/0.039	0.176/0.187
IBRNet	1.69/1.62	0.000/0.000	0.000/0.001
Ours	0.023/**0.035**	**0.746**/**0.717**	0.913/**0.866**

- Evaluate our unsupervised depth reconstruction on the DTU dataset

Novel view synthesis

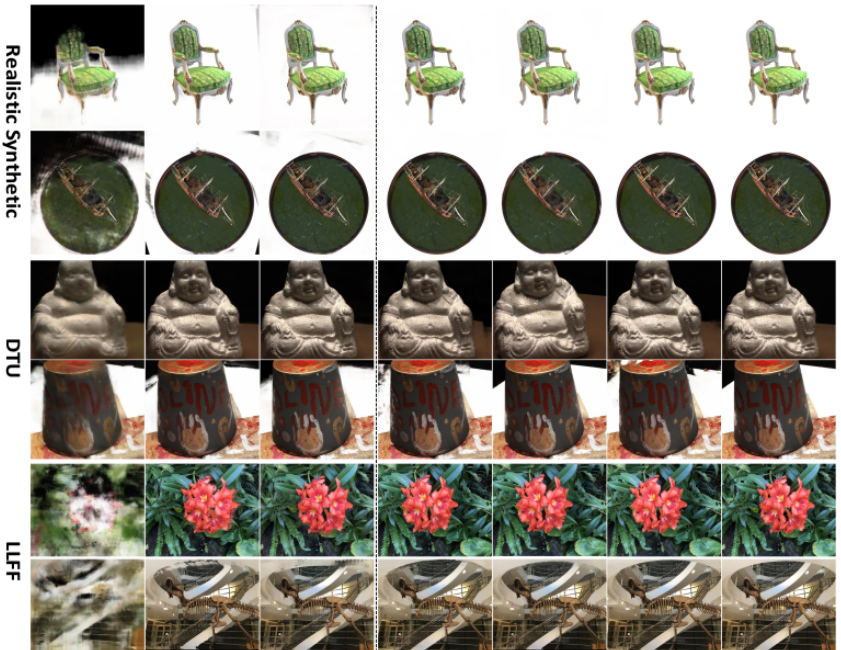


Ours $_{ft\text{-}1.25h}$



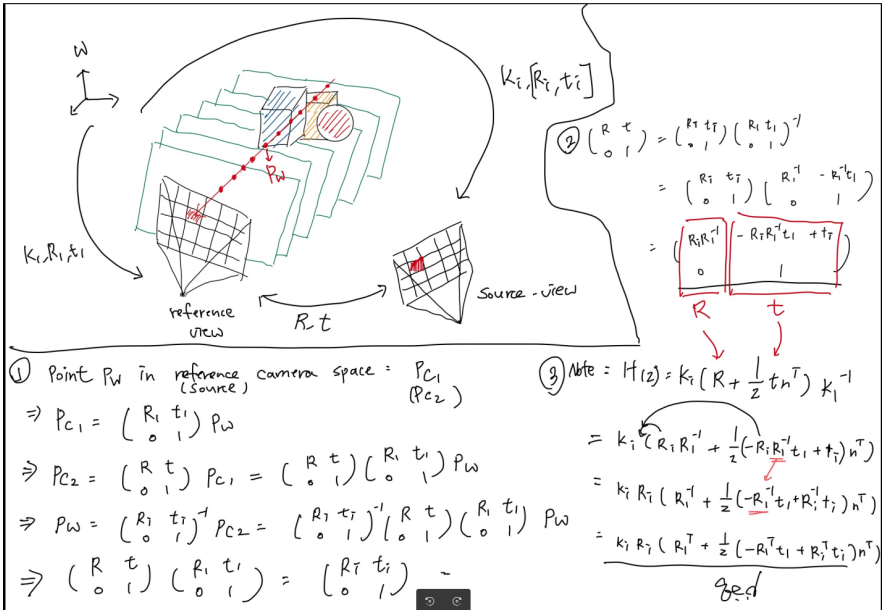
Method	Settings	Synthetic Data (NeRF [31])			Real Data (DTU [20] / Forward-Facing [33])		
		PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
PixelNeRF [54]	No per-scene optimization	7.39	0.658	0.411	19.31/11.24	0.789/0.486	0.382/0.671
IBRNet [46]		22.44	0.874	0.195	26.04/21.79	0.917/0.786	0.190/0.279
Ours		**23.62**	**0.897**	**0.176**	**26.63/21.93**	**0.931/0.795**	**0.168/0.252**
NeRF$_{10.2h}$ [34]	Per-scene optimization	**30.63**	**0.962**	**0.093**	27.01/**25.97**	0.902/0.870	0.263/ 0.236
IBRNet$_{ft-1.0h}$ [46]		25.62	0.939	0.110	**31.35**/24.88	**0.956**/0.861	**0.131/0.189**
Ours$_{ft-15min}$		27.07	0.931	0.168	28.50/25.45	0.933/**0.877**	0.179/0.192

Novel view synthesis(cont.)



End (proof on next slide)

Derivation Homography wrapping formula



② $\begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} R_i & t_i \\ 0 & 1 \end{pmatrix} \begin{pmatrix} R_1 & t_1 \\ 0 & 1 \end{pmatrix}^{-1}$

$= \begin{pmatrix} R_i & t_i \\ 0 & 1 \end{pmatrix} \begin{pmatrix} R_1^{-1} & -R_1^{-1}t_1 \\ 0 & 1 \end{pmatrix}$

$= \begin{pmatrix} R_i R_1^{-1} & -R_i R_1^{-1} t_1 + t_i \\ 0 & 1 \end{pmatrix}$

① Point P_W in reference camera space (source) = P_{C_1} (P_{C_2})

$\Rightarrow P_{C_1} = \begin{pmatrix} R_1 & t_1 \\ 0 & 1 \end{pmatrix} P_W$

$\Rightarrow P_{C_2} = \begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix} P_{C_1} = \begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix} \begin{pmatrix} R_1 & t_1 \\ 0 & 1 \end{pmatrix} P_W$

$\Rightarrow P_W = \begin{pmatrix} R_1 & t_1 \\ 0 & 1 \end{pmatrix}^{-1} P_{C_2} = \begin{pmatrix} R_1 & t_1 \\ 0 & 1 \end{pmatrix}^{-1} \begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix} \begin{pmatrix} R_1 & t_1 \\ 0 & 1 \end{pmatrix} P_W$

$\Rightarrow \begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix} \begin{pmatrix} R_1 & t_1 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} R_i & t_i \\ 0 & 1 \end{pmatrix}$ ~

③ Note : $H(z) = K_i \left(R + \frac{1}{z} t n^T \right) K_1^{-1}$

$= K_i \left(R_i R_1^{-1} + \frac{1}{z} (-R_i R_1^{-1} t_1 + t_i) n^T \right)$

$= K_i R_i \left(R_1^{-1} + \frac{1}{z} (-R_i^{-1} t_1 + R_i^{-1} t_i) n^T \right)$

$= K_i R_i \left(R_1^T + \frac{1}{z} (-R_1^T t_1 + R_i^T t_i) n^T \right)$

ged