NeRF for dynamic scene

2023 여름 세미나

Sogang University
Vision & Display Systems Lab, Dept. of Electronic Engineering

Presented By
염수웅

DynIBaR
Neural Dynamic Image-Based Rendering

Zhengqi Li[1], Qianqian Wang[1,2], Forrester Cole[1], Richard Tucker[1], Noah Snavely[1]

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

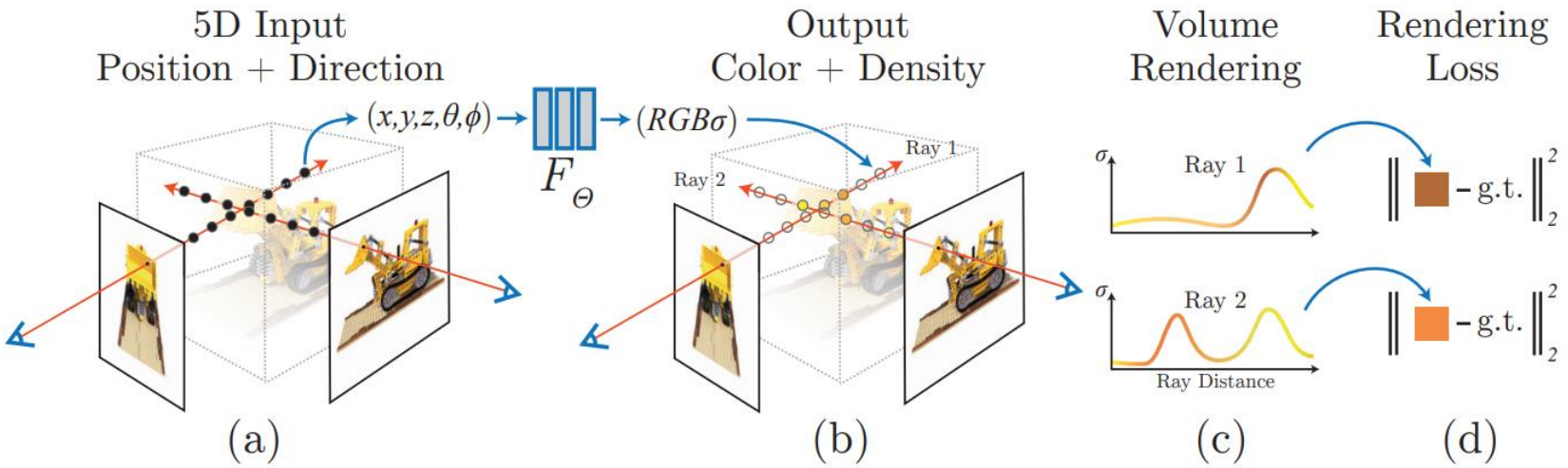
CVPR 2023 (Best Paper Honorable Mention)

Outline

- What is NeRF?
  - NeRF
- Prior Knoledge
  - Dynamic Scene Novel View Synthesis → Neural Scene Flow Fields
  - Image Based Rendering → IBRNet
  - Combining the two representation → NeRF in the Wild
- DynIBaR: Neural Dynamic Image-Based Rendering

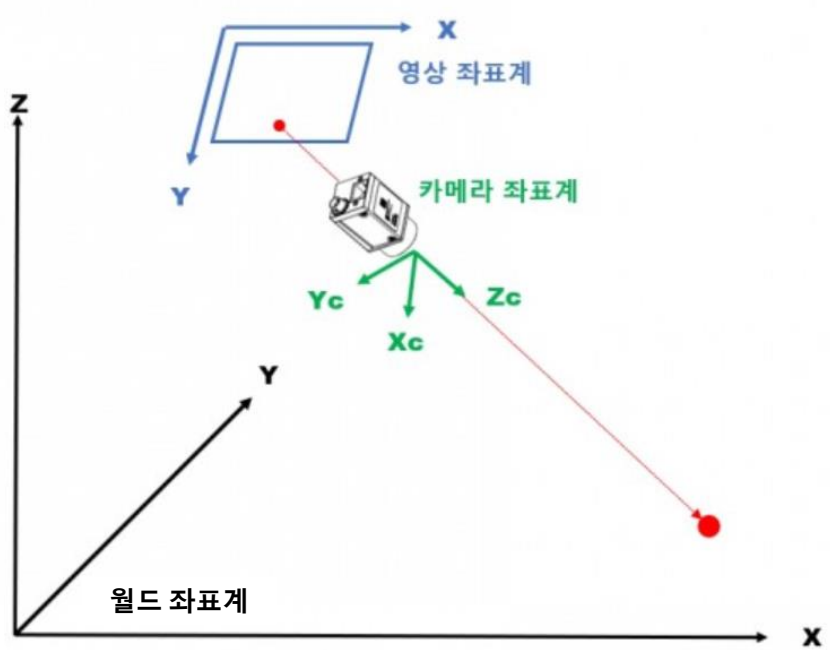
NeRF[ECCV 20]

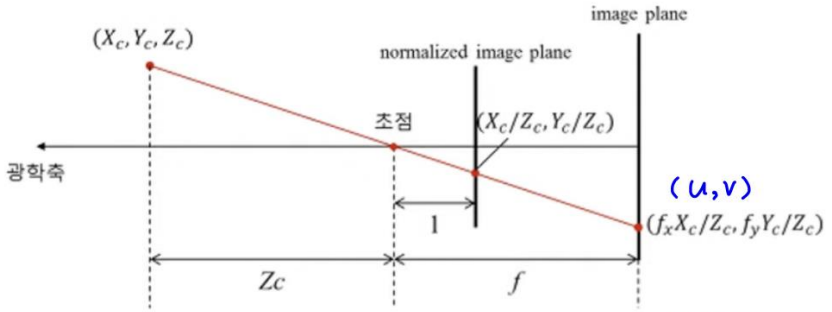
- Over view of NeRF
  - 다수의 이미지 통해 3D 구조 학습 후 이미지 렌더링
    - Input: image
      - pixel의 ray 위에 존재하는 point와 camera direction을 MLP에 쿼리
    - Output: imag
      - MLP의 output color, density를 이용한 volume rendering



NeRF[ECCV 20]

- Calculate ray
  - 카메라의 내/외부 파라미터 활용





2D Image Coordinate

(skew_c_x : 비대칭계수) ↗ 카메라는 skew error 거의없음

3D World Coordinates

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = A[R|t] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

2D Image Coordinates ⇓ Image Plane

Intrinsic properties (Optical Centre, scaling) ⇓ 내부파라미터 (Intrinsic)

C_x, C_y : principle point (주점) ⇒ 카메라 렌즈의 중심

f_x, f_y : focal Length

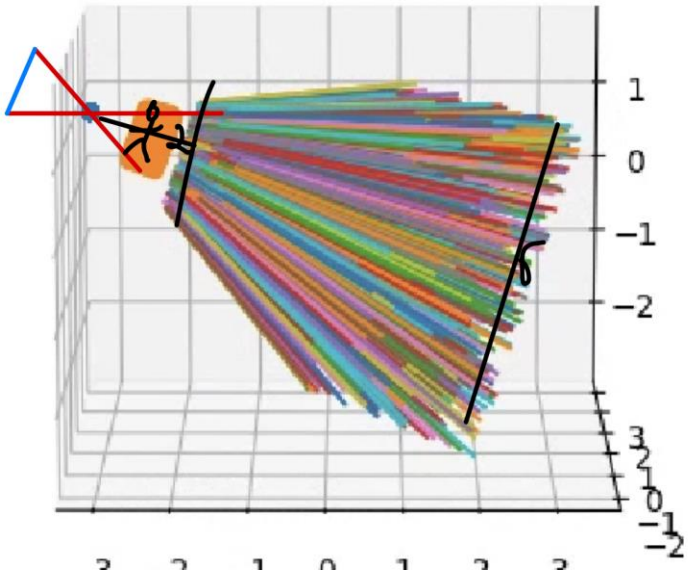
Extrinsic properties (Camera Rotation and translation) ⇓ 외부파라미터 (Extrinsic)

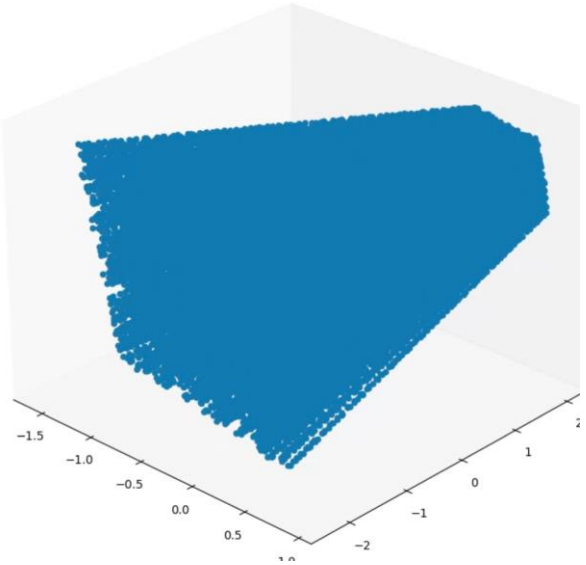
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

3D World Coordinates

NeRF[ECCV 20]

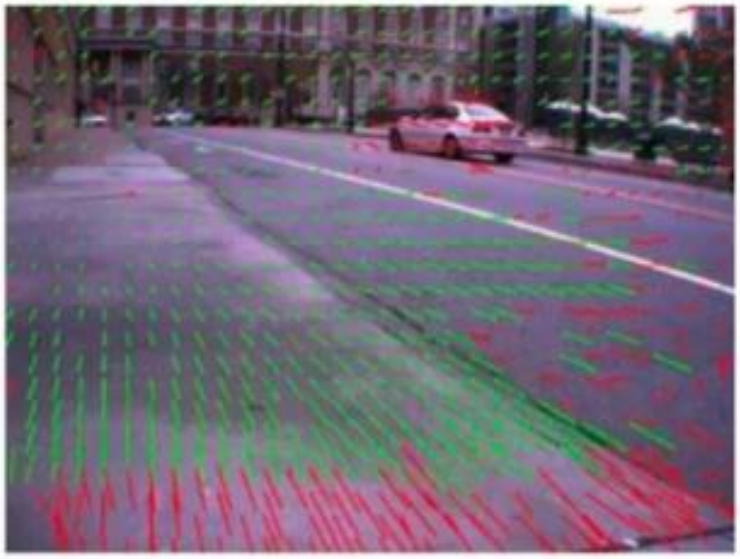
- Calculate ray
  - Ray
    - Ray란 각 pixel들로부터 실제상(real space) 방향으로 뻗어 나가는 빛을 모델링
      - $r(t) = o + td$
        - ✓o: 카메라의 위치(translation), d: viewing direction
        - ✓$d \rightarrow$ normalized plane에서 real space 방향의 벡터에 rotation vector를 곱한 값





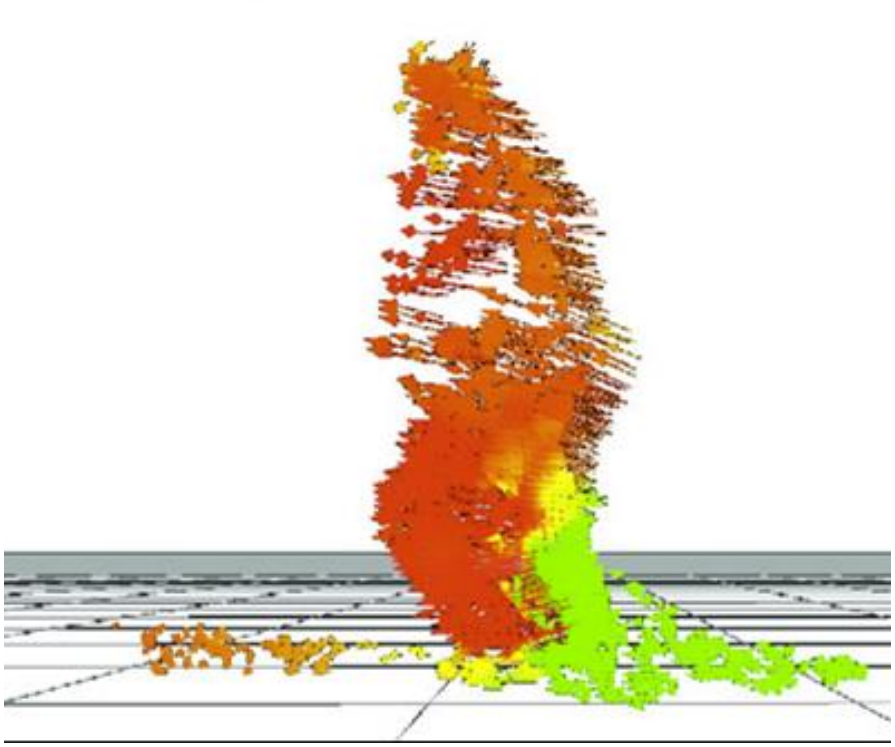
Neural Scene Flow Fields[ICCV 21]

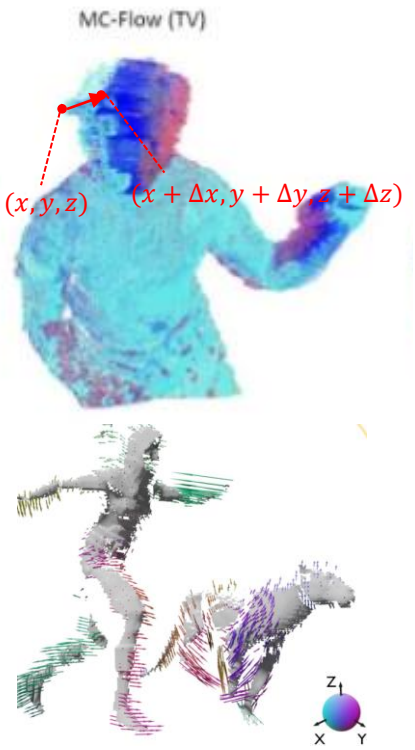
- NSFF
  - 한 개의 비디오만 가지고 다양한 view synthesis + moving object 대응 가능
    - 비디오에 있는 각 프레임들을 NeRF에서 사용하는 N개의 이미지처럼 사용
    - 프레임 내 물체의 움직임 존재, 카메라의 움직임 존재
  - Optical Flow
    - 연속한 두 프레임 사이에서 각 픽셀의 motion을 나타내는 vector map



Neural Scene Flow Fields[ICCV 21]

- NSFF
  - Scene Flow
    - 3차원 공간상의 어떤 점이 다음 프레임에서 x,y,z 방향에서 어떤 방향으로 얼마나 움직였는지를 나타내는 벡터들의 집합





Neural Scene Flow Fields[ICCV 21]

- NSFF
  - NeRF vs NSFF

$$(c, \sigma) = F_\Theta(\mathrm{x},\mathrm{d})$$

$$\hat{C}(r) = \int_{t_n}^{t_f} T(t)\sigma(r(t))c(r(t),\mathrm{d})dt$$

(Volume Rendering)

$$T(t) = \exp(-\int_{t_n}^{t} \sigma(r(s))ds)$$

(Transmittance)

$$L_{static} = \sum_r \left\| \hat{C}(\mathrm{r}) - C(\mathrm{r}) \right\|_2^2$$

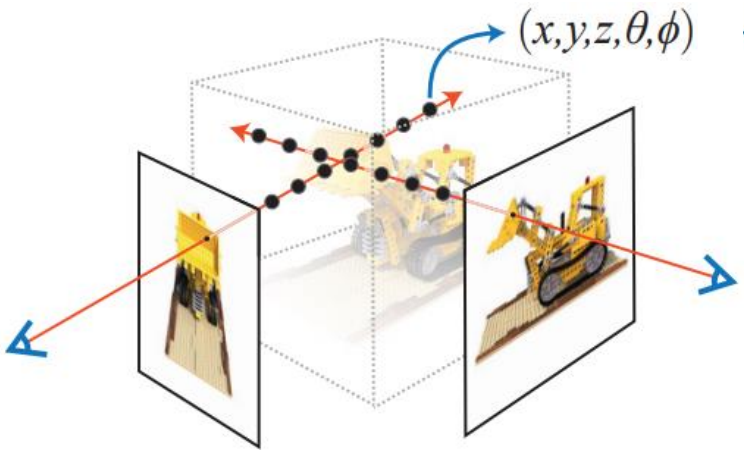
$$(c_i, \sigma_i, \mathcal{F}_i, \mathcal{W}_i) = F_\Theta{}^{dy}(\mathrm{x},\mathrm{d},i)$$

$$\hat{C}_{j\to i}(r_i) = \int_{t_n}^{t_f} T_j(t)\sigma_j(r_{i\to j}(t))c_j(r_{i\to j}(t),\mathrm{d}_i)dt$$

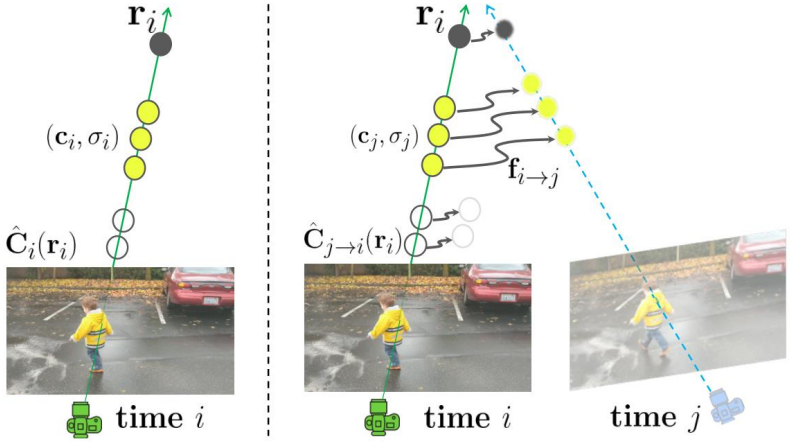
(Volume Rendering)

$$r_{i\to j}(t) = r_i(t) + f_{i\to j}(r_i(t))$$

$$L_{pho} = \sum_{r_i} \sum_{j\in\mathcal{N}} \left\| \hat{C}_{j\to i}(r_i) - C_i(r_i) \right\|_2^2$$



<NeRF>



<NSFF>

Neural Scene Flow Fields[ICCV 21]

- NSFF
  - i 시점 기준으로 j 시간의 프레임에 있는 점들의 color, density 고려
    - 추정된 scene flow를 이용하여 i 시간 프레임의 point warping
    - j 시간에서 warpin된 point들의 color와 density 값을 volume rendering 진행
      - 시간은 i 기준이며, color와 density는 j시간의 프레임에서 얻는 것
    - Scene flow fields를 통해 dynamic 객체의 motion 고려 가능
      - 시간에 대한 일관성을 가질 수 있음

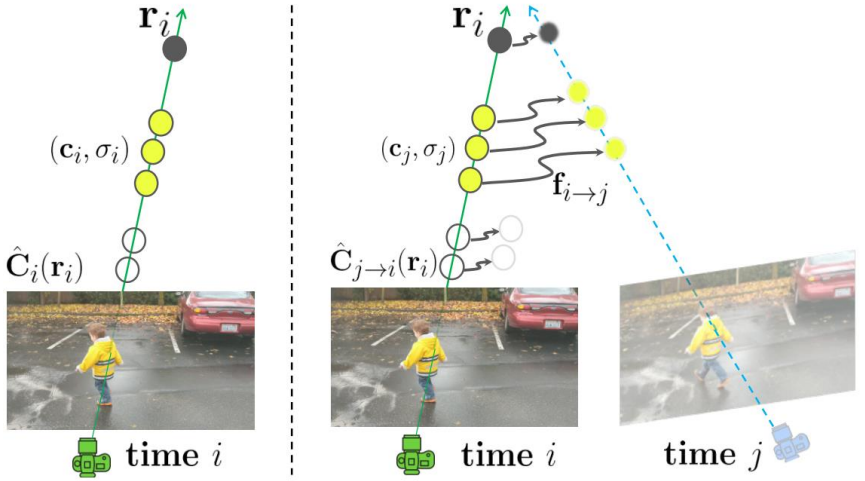
$$(c_i, \sigma_i, \mathcal{F}_i, \mathcal{W}_i) = {F_\Theta}^{dy}(\mathrm{x,d},i)$$

$$\hat{C}_{j\to i}(r_i) = \int_{t_n}^{t_f} T_j(t)\sigma_j(r_{i\to j}(t))c_j(r_{i\to j}(t),\mathrm{d}_i)dt$$

(Volume Rendering)

$$r_{i\to j}(t) = r_i(t) + f_{i\to j}(r_i(t))$$

$$L_{pho}=\sum_{r_i}\sum_{j\epsilon\mathcal{N}}\left\|\hat{C}_{j\to i}(r_i) - C_i(r_i)\right\|_2^2$$



Neural Scene Flow Fields[ICCV 21]

- NSFF
  - Disocculusion weight
    - I시간에 있었던 장면이 j시간에서 없을 수 있음(occulusion)
      - Occulusion으로 인해 photometric loss 최적화 어려움 발생
      - Occulusion에 대한 penaly 항을 추가하여 극복

✓ $\widehat{W}_{j\to i}(r_i) = \int_{t_n}^{t_f} T_j(t)\sigma_j\left(r_{i\to j}(t)\right) w_{i\to j}\left(r_i(t)\right)dt$

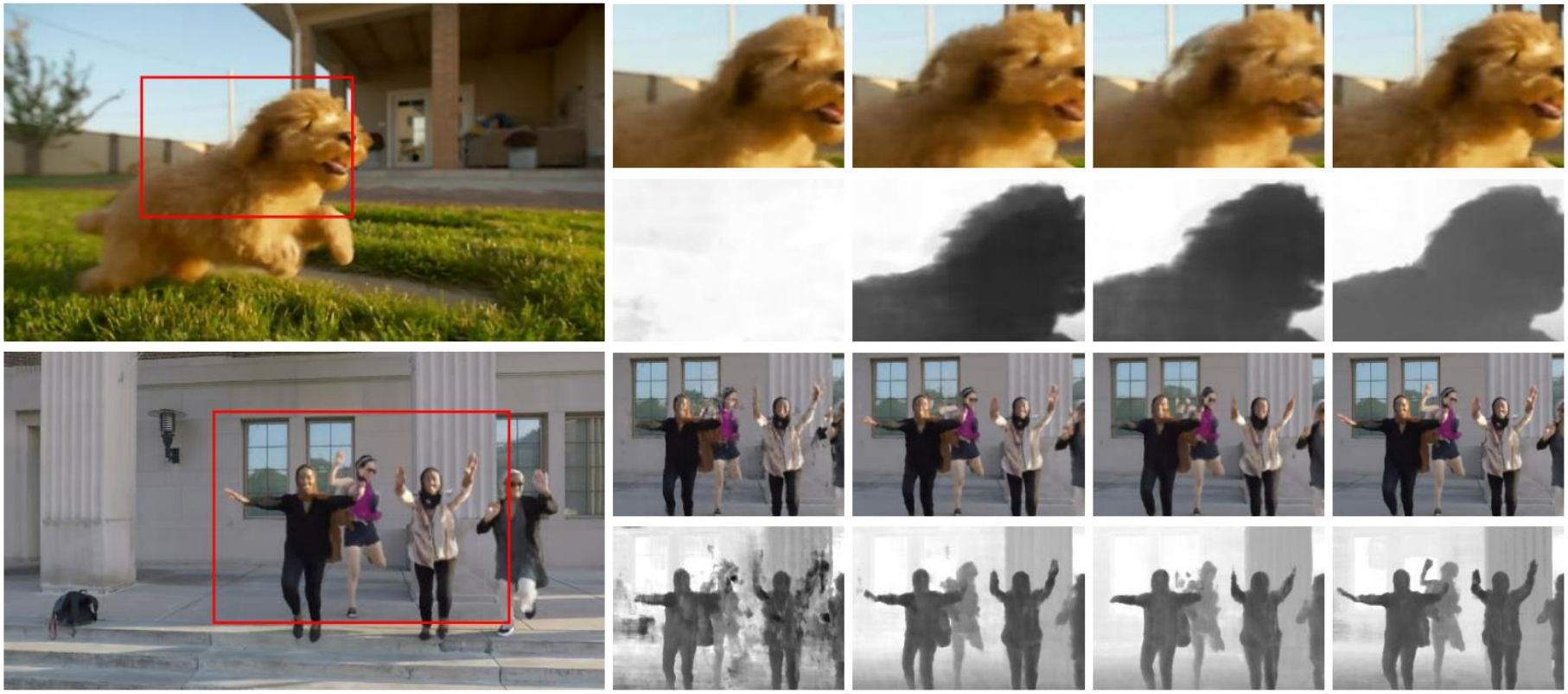
- $w_{i\to j} \to 0\sim1$
- MLP에서 추정된 disocculusion weight fields를 volume rendering 진행

앞의 term이 0이 되는것을 방지하기 위한 L1 regularization

✓ $L_{pho} = \sum_{r_i}\sum_{j\in\mathcal{N}(i)} \widehat{W}_{j\to i}(r_i)\left\|\hat{C}_{j\to i}(r_i) - C_i(r_i)\right\|_2^2 + \beta_w \sum_{\mathrm{x}_i}\left\|w_{i\to j}(\mathrm{x}_i) - 1\right\|_1$

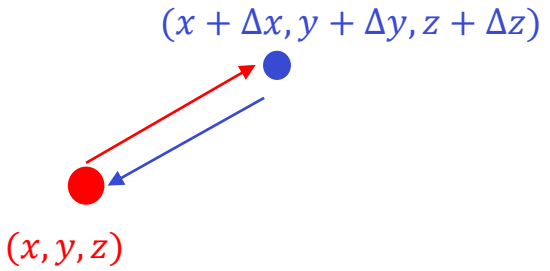
Neural Scene Flow Fields[ICCV 21]

- NSFF
  - Disocculusion weight

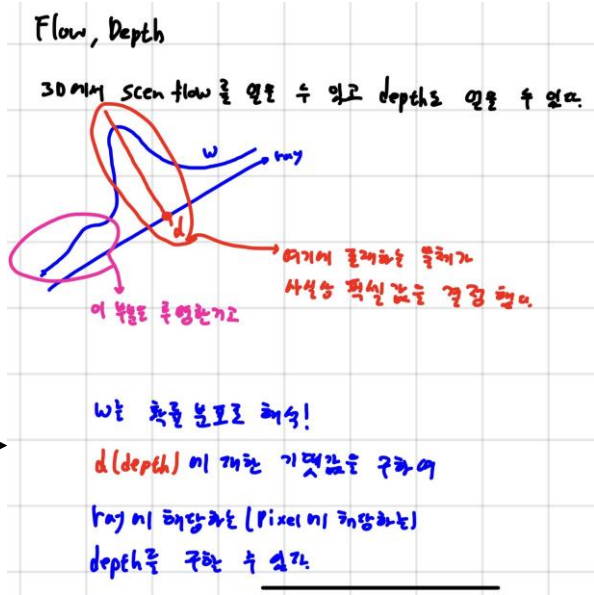


(a) Our rendered views (b) w/o $\mathcal{L}_{\text{data}}$ (c) w/o $\mathcal{L}_{\text{cyc}}$ (d) w/o $\mathcal{W}_i$ (e) Full

Neural Scene Flow Fields[ICCV 21]

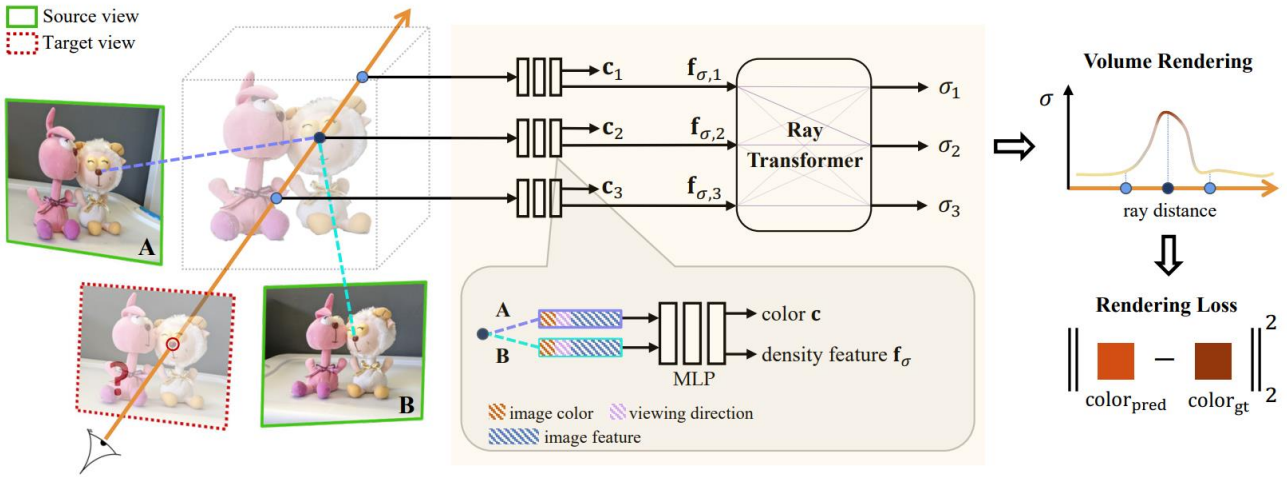


- NSFF
  - Scene Flow Priors
    - Smoothness Term
      - 근처에 있는 Flow 끼리는 운동 방향이나 크기가 비슷해야함
        - ✓한 ray 의 point에 대한 근처에 있는 ray의 point들의 flow 유사
        - ✓Ray에 따라 Scene Flow를 미분한 결과의 크기를 Regularization Term으로 추가
    - Cycle Consistency Termx
      - Scene Flow를 따라 도달한 지점에서 Back Flow를 구하면 출발 위치로 돌아와야 함
      - $L_{cyc} = \sum_{x_i} \sum_{j \in \mathcal{N}(i)} w_{i \to j} \left\| f_{i \to j}(x_i) + f_{j \to i}(x_{i \to j}) \right\|_1$
  - Data Driven Priors
    - Flow: Flow Net v2를 통해 구한 Optical Flow를 Supervision
      - $L_{geo} = \sum_{r_i} \sum_{j \in \mathcal{N}(i)} \left\| \hat{p}_{i \to j}(r_i) - p_{i \to j}(r_i) \right\|_1$
    - Depth: Ranftl et al. 을 통해 구한 Detph를 Supervision
      - $L_z = \sum_{r_i} \left\| \hat{Z}_i^*(r_i) - Z_i^*(r_i) \right\|_1$



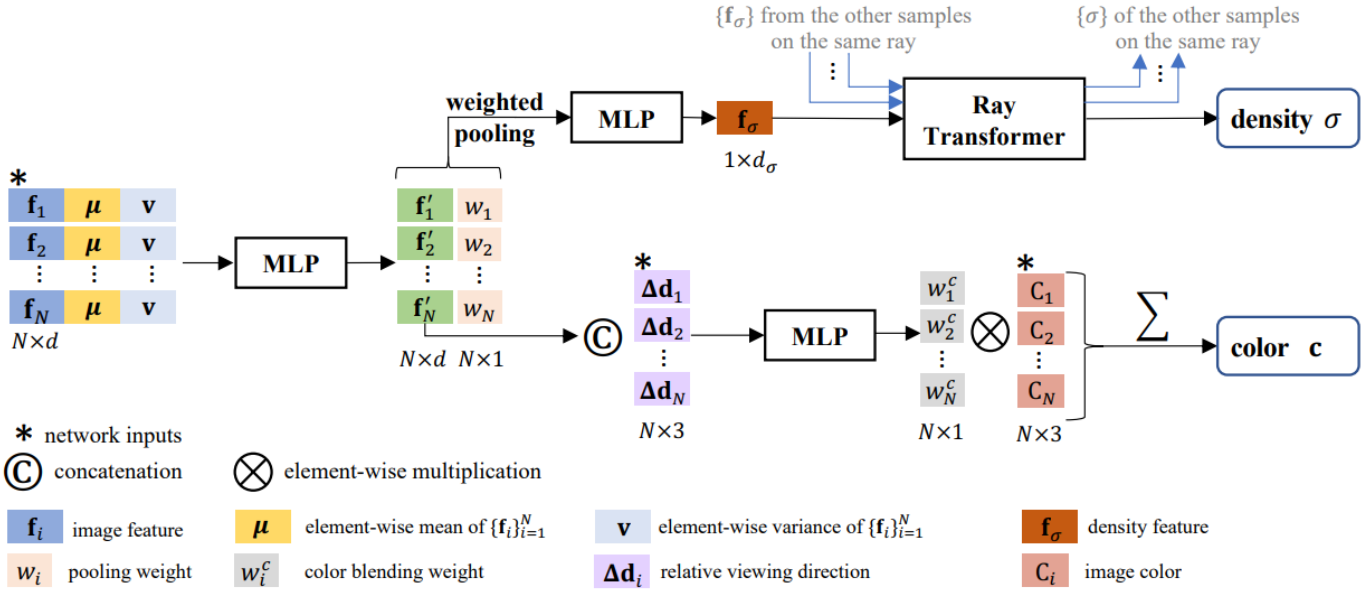
IBRNet[CVPR 21]

- IBRNet: Learning Multi-View Image-Based Rendering
  - Neural Rendering 이전의 Image Based Rendering (IBR)
    - 초기 IBR방법은 참조 이미지들에 가중치를 주어 새로운 뷰 합성
      - 참조 이미지들은 픽셀을 가중치에 따라 섞어 새로운 뷰 생성하는 방법 사용
  - IBRNet
    - 기존의 IBR 방법과 NeRF에서 사용되는 neural rendering방법 결합
      - Multi-view Image-Based Rendering 방법을 사용하여 새로운 뷰 합성
        - ✓MLP와 ray transformer를 사용하며 volume rendering을 사용하여 렌더링 진행



IBRNet[CVPR 21]

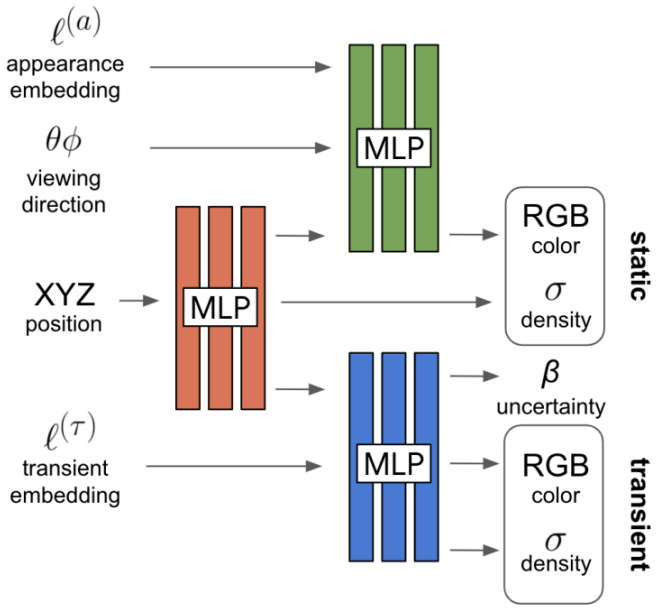
- IBRNet: Learning Multi-View Image-Based Rendering
  - IBRNet
    - 기존의 IBR 방법과 NeRF에서 사용되는 neural rendering방법 결합
      - Epipolar geometry를 사용하여 참조 이미지에서 대응되는 feature 찾음
        - ✓ray상의 point에 대응되는 참조 이미지의 feature는 epipolar geometry로 구함
      - Unet 기반의 CNN 네트워크를 이용하여 이미지의 feature및 색상 정보 추출



NeRF in the Wild[CVPR 21]

- NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collection
  - Wild한 이미지를 사용하여 복잡한 장면의 새로운 시점 합성
    - 가변 조명과 일시적으로 가려지는 물체로 인해 발생하는 문제 해결
      - Static 요소와 transient요소를 따로 렌더링
      - 따로 렌더링한 요소들을 결합하여 최종 이미지 생성

(a) Photos (b) Renderings



NeRF in the Wild[CVPR 21]

- NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collection
  - Static 요소와 transient요소를 따로 렌더링
    - 따로 렌더링한 요소들을 결합하여 최종 이미지 생성
      - I_{static} and $I_{transient}$
    - Uncertainty를 이용하여 두 이미지의 가중치 계산
      - w_{static} and $w_{transient}$
      - Uncertainty가 높은 영역은 가중치가 작고 낮은 영역은 커짐
    - 렌더링된 두 이미지를 가중평균하여 최종 이미지 생성
      - $I_{final} = (w_{static} \times I_{static} + w_{transient} \times I_{transient})/(w_{static} + w_{transient})$

(a) Static (b) Transient (c) Composite (d) Image (e) Uncertainty

DynIBaR:Neural Dynamic Image Based Rendering

- Introduction
  - Dynamic Scene Rendering
    - 단안 카메라로 촬영해 습득한 비디오에 대한 새로운 시점 합성
      - 습득된 비디오내에 static scene과 motion이 있는 객체가 동시에 존재
      - 고정된 카메라가 아닌 움직이는 카메라를 이용하여 대상을 촬영
    - ECCV[2020]에 NeRF가 발표된 이후 2021년부터 dynamic scene에 대한 논문 시작
      - NSFF와 같이 motion 고려한 radiance fields에 대한 연구
        - ✓제한된 카메라 움직임과 짧은 비디오에 대해서만 좋은 성능 보임
          - 제약이 없는 카메라 움직임과 긴 비디오에 대한 확장이 어려움
      - HyperNeRF와 같이 Canonical Space를 구성하여 dynamic scene을 고려하는 연구
        - ✓Canonical model을 구성하는 방법의 경우 제어된 카메라 경로와 객체 중심적인 장면에 대해서는 잘 작동하지만, 복잡한 객체 움직임을 가진 장면은 어려움
    - DynIBaR의 경우 dynamic scene에 neural rendering과 image based rendering을 모두 고려하여 NSFF나 HyperNeRF에서 발생하는 문제를 해결하는 고성능 모델임

DynIBaR:Neural Dynamic Image Based Rendering

- Introduction
  - Problem setting
    - Long time duration
      - 비디오의 길이가 비교적 긴 경우(약 30s)에도 새로운 시점 합성
    - Unbounded scenes
      - 경계가 뚜렷하게 정해져 있지 않은 장면에 대해 수행 가능 목표
        - ✓장면의 크기나 범위가 무한정일 수 있음
    - Uncontrolled camera trajectories
      - 카메라의 움직임이 미리 계획되지 않고, 자유롭게 움직이는 경우에 대해 수행
    - Fast and complex object motion
      - 비디오 내의 객체가 빠르게 움직이거나 복잡한 움직임을 가지는 경우에 대해 수행

DynIBaR:Neural Dynamic Image Based Rendering

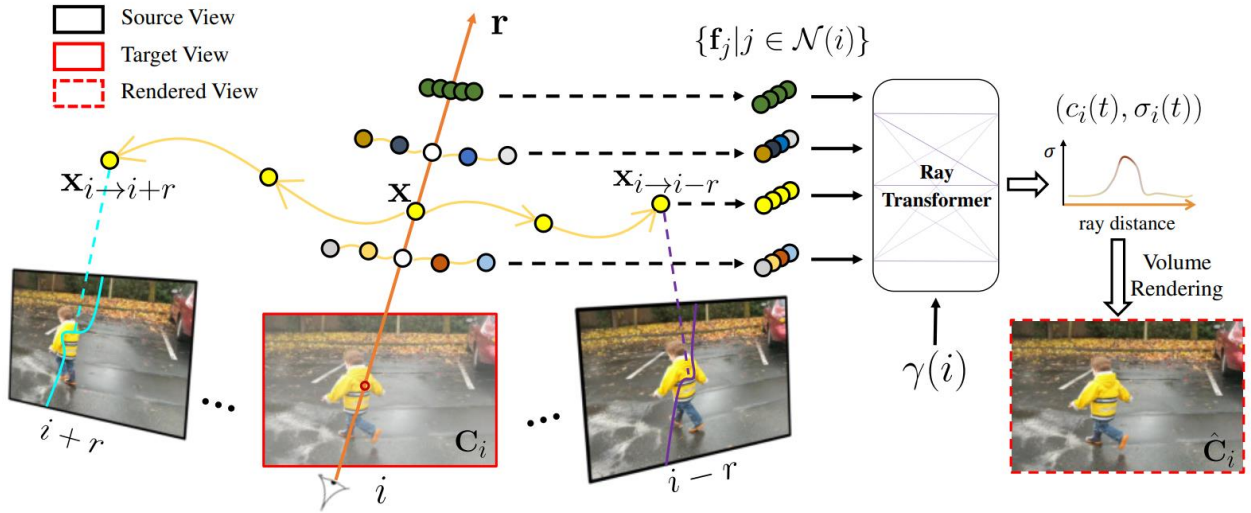
- Introduction
  - Dataset
    - Dynamic Scene Video
      - Motion객체를 움직이는 단안 카메라로 촬영한 비디오
        - ✓Nvidia Dynamic Dataset, UCSD Dynamic Scenes Dataset(30s~1m)
          - 각 데이터셋은 8개의 forward-facing dynamic scene으로 구성
          - 동기화된 Multi-view 카메라로 촬영된 비디오에서 monocular video 생성
    - Calibration
      - 비디오의 각 프레임에 대해 SfM tool을 사용하여 카메라 파라미터 취득(COLMAP)
      - COLMAP은 정적 scene에 대한 카메라 파라미터를 추정
        - ✓Motion이 있는 객체를 제외한 배경 이미지들을 통해 카메라 파라미터 취득
        - ✓Motion이 있는 객체가 화면을 크게 차지하는 경우 카메라 파라미터 취득 실패
          - Matching되는 feature 수가 부족하여 카메라 파라미터 추정에 지장이 생김
          - 카메라 파라미터 취득에 실패한 data는 제외

DynIBaR:Neural Dynamic Image Based Rendering

- Introduction
  - Dataset

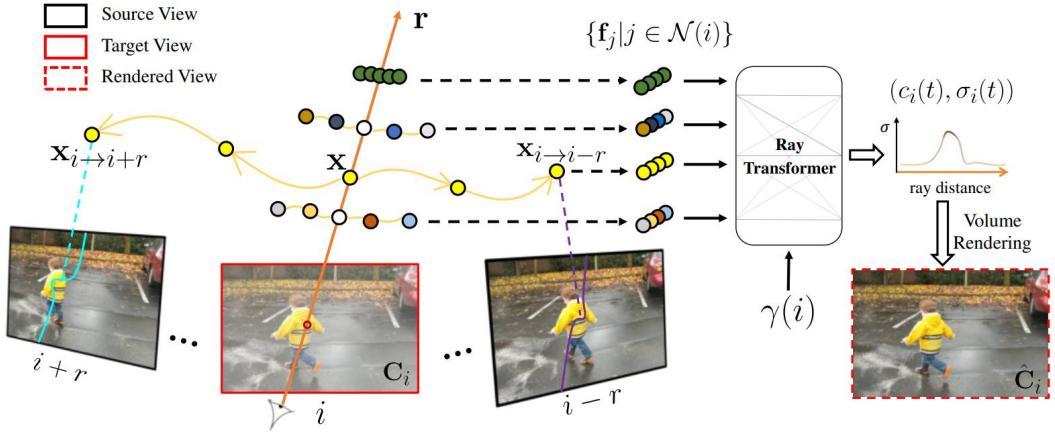
DynIBaR:Neural Dynamic Image Based Rendering

- Methods
  - Dynamic Image-Based Rendering
    - IBRNet과 유사하게 source view의 feature를 렌더링에 이용
      - Dynamic scene에 대해 epipolar geometry 적용 불가
      - Target view에 대한 motion trajectory를 추정하여 source view의 featur를 aggregation
        - ✓Unet 기반의 CNN네트워크를 이용하여 source view에 대한 feature 획득



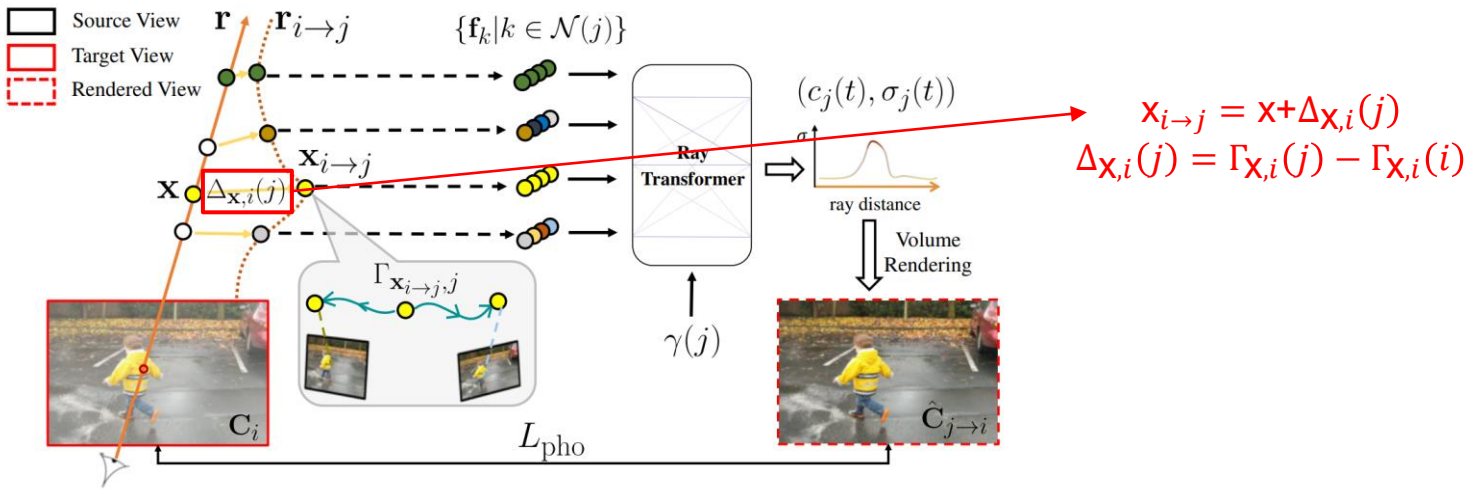
DynIBaR:Neural Dynamic Image Based Rendering

- Methods
  - Motion trajectory fields
    - 학습되어지는 basis로 motion trajectory 표현하여 장면의 움직임을 표현
      - Motion trajectory $\Gamma_{\mathrm{x},i}(j) = \sum_{l=1}^{L} h^l{}_j \phi^l{}_i(\mathrm{x})$
        - ✓Trajectory coefficients $\{\phi_i^l\}_{l=1}^L$
        - ✓Motion basis $\{h_i^l\}_{l=1}^L \mathrm{x}$
      - Trajectory coefficient와 motion basis는 모두 MLP $G_{MT}(\Upsilon(\mathrm{x}), \Upsilon(i))$에 의해 공동 학습
        - ✓Motion basis는 모든 입력 비디오의 time step에 대해 표현 가능



DynIBaR:Neural Dynamic Image Based Rendering

- Methods
  - Cross-time rendering for temporal consistency
    - Dynamic scene에 대한 rendering 수행 시 시간에 대한 인스턴스를 고려하는 것이 중요
      - 시간을 고려하지 않고 렌더링 된 이미지만을 비교할 경우 train image에 오버피팅
    - 시간에 따른 움직임이 고려되는 Motion trajectory를 이용한 loss를 설계하여 최적화
      - $L_{pho} = \sum_r \sum_{j \epsilon \mathcal{N}(i)} \hat{W}_{j \to i}(r) \rho(C_i(r), \hat{C}_{j \to i}(r))$
        - ✓ρ: RGB Loss
    - NSFF에선 Scene Flow를 고려하였으나 DynIBaR에선 motion trajectory 적용
      - Image-Based Rendering에서 scene flow를 고려할 경우 연산량이 크게 증가



DynIBaR:Neural Dynamic Image Based Rendering

- Methods
  - Combining static and dynamic models
    - 동적인 장면과 정적인 장면의 서로 다른 특성으로 인해 각각 나누어 렌더링 진행
      - 동적 장면은 시간에 따라 변화하는 내용을 포함
        - ✓이를 정적인 모델로 표현하는 것은 한계 존재
      - 정적 장면은 시간 관계없이 일정한 내용을 지님
        - ✓이를 동적인 모델로 표현하는 것은 불필요한 계산 초래
      - 동적인 장면과 정적인 장면을 나누어 처리하면 효율적인 계산 가능
        - ✓동적인 장면은 시간에 따라 변화하는 내용이 많아, 이를 모두 고려시 계산↑
        - ✓정적 장면은 시간에 관계없이 일정한 내용이기 때문에, 이를 따로 처리하면 계산 비용을 줄일 수 있음

DynIBaR:Neural Dynamic Image Based Rendering

- Methods
  - Combining static and dynamic models
    - 동적 장면을 고려하기 위해 target view의 주변 소수 source view를 이용할 경우 정적인 고품질 장면을 렌더링 하기에 충분하지 않음
      - 동적인 장면과 정적인 장면의 서로 다른 특성으로 인해 각각 따로 렌더링 진행
      - 동적인 장면과 정적인 장면을 따로 렌더링 후 NeRF in the Wild 에서 사용한 combining 수행
        - ✓Photometric loss를 term을 다시 정의 가능
        - ✓$L_{pho} = \sum_r \sum_{j \epsilon \mathcal{N}(i)} \widehat{W}_{j \to i}(r) \rho(C_i(r), \hat{C}^{full}{}_{j \to i}(r))$
          - $\hat{C}^{full}{}_{j \to i}$: static conten의 color $\hat{C}^{st}$와 dynamic content의 color $\hat{C}^{st}$ 가 결합된 형태

DynIBaR:Neural Dynamic Image Based Rendering

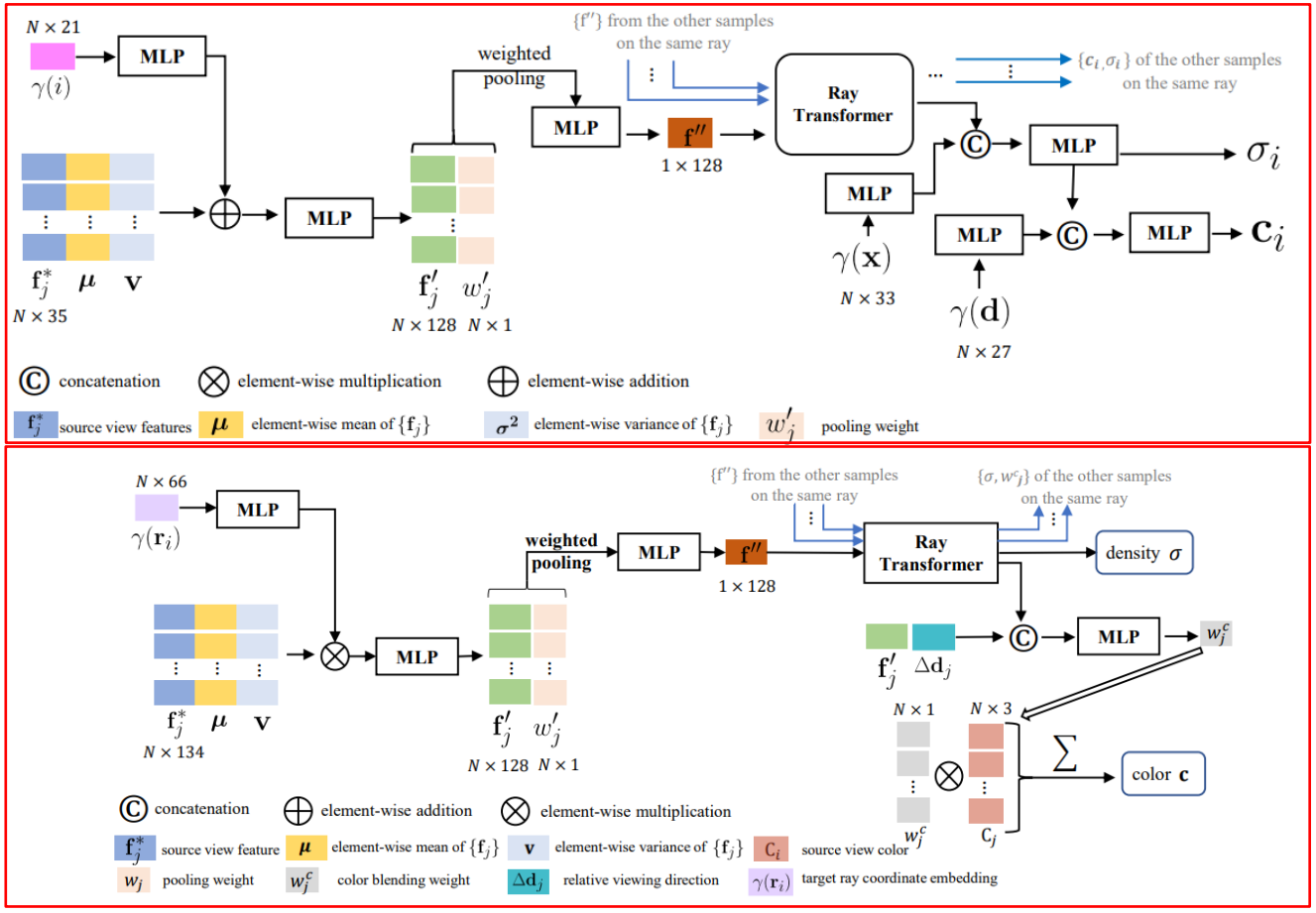
- Methods
  - Image-based motion segmentation
    - 정적인 장면과 동적인 장면을 구분하기 위해 segmentation mask 적용
      - Segmentation mask는 입력 프레임에서 정적인 장면과 동적인 장면을 구분하고, 이를 이용하여 더 나은 재구성 결과를 얻을 수 있도록 함
      - Omnimatte[CVPR 21]를 활용하여 motion mask 생성
        - ✓Video에서 동적 객체와 정적인 장면을 분할하는 모델
      - $L_{mask} = \sum_r (1 - M_i)(r)\rho\left(\hat{C}^{st}(r), C_i(r)\right) + \sum_r M_i(r)\rho\left(\hat{C}^{dy}(r), C_i(r)\right)$

DynIBaR:Neural Dynamic Image Based Rendering

- Methods
  - Total Loss
    - $L = L_{pho} + L_{mask} + L_{reg}$
      - L_{reg}: regularization loss
        - ✓Neural Scene Flow Fields에서 사용한 depth 기반 loss, optical flow 기반 loss, cycle loss가 더해진 형태로 이루어짐

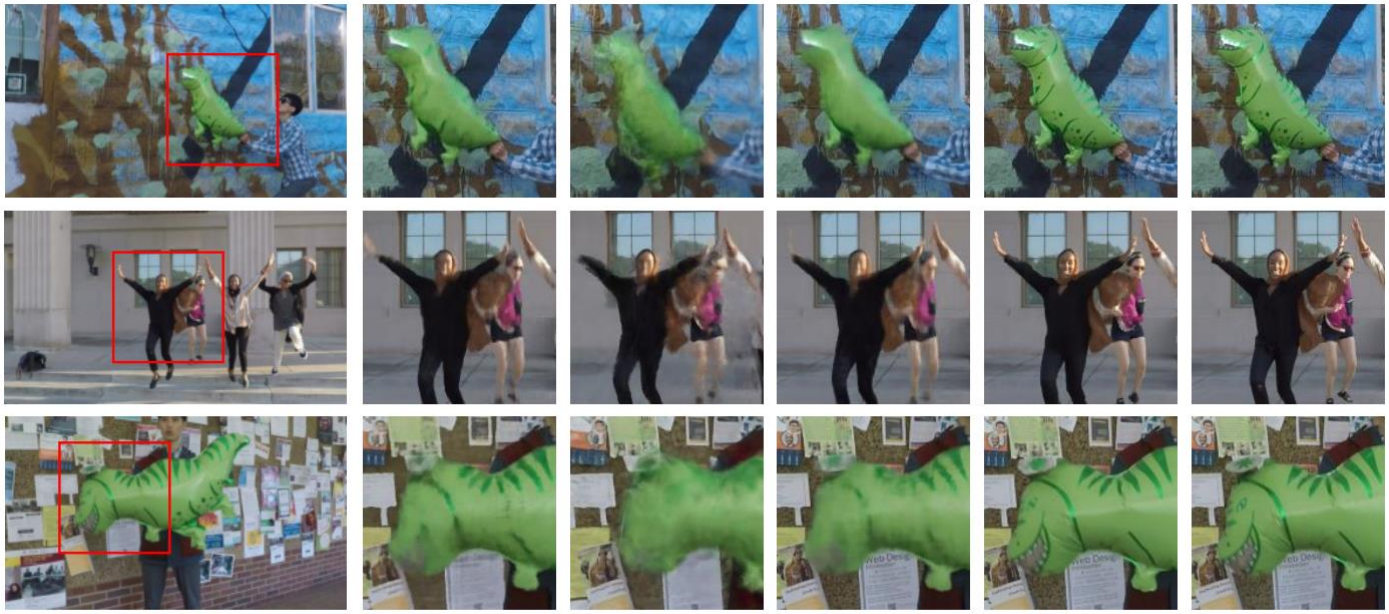
DynIBaR:Neural Dynamic Image Based Rendering

- Methods
  - Combining static and dynamic models



DynIBaR:Neural Dynamic Image Based Rendering

- Results
  - Nvidia dataset에 대한 결과 비교



Our rendered views | DVS | HyperNeRF | NSFF | Ours | GT

Methods	Full			Dynamic Only		
	SSIM↑	PSNR↑	LPIPS↓	SSIM↑	PSNR↑	LPIPS↓
Nerfies [49]	0.609	20.64	0.204	0.455	17.35	0.258
HyperNeRF [50]	0.654	20.90	0.182	0.446	17.56	0.242
DVS [19]	0.921	27.44	0.070	0.778	22.63	0.144
NSFF [35]	0.927	28.90	0.062	0.783	23.08	0.159
Ours	**0.957**	**30.91**	**0.027**	**0.826**	**24.31**	**0.062**

DynIBaR:Neural Dynamic Image Based Rendering

- Results
  - UCSD dataset에 대한 결과 비교

Our rendered views | DVS | HyperNeRF | NSFF | Ours | GT

Methods	Full			Dynamic Only		
	SSIM↑	PSNR↑	LPIPS↓	SSIM↑	PSNR↑	LPIPS↓
Nerfies [49]	0.823	24.32	0.096	0.595	18.45	0.234
HyperNeRF [50]	0.859	25.10	0.095	0.618	19.26	0.212
DVS [19]	0.943	30.64	0.075	0.866	26.57	0.096
NSFF [35]	0.952	31.75	0.034	0.851	25.83	0.115
Ours	**0.983**	**36.47**	**0.014**	**0.909**	**28.01**	**0.042**

DynIBaR:Neural Dynamic Image Based Rendering

- Results

DynIBaR:Neural Dynamic Image Based Rendering

- Results
  - Limitation and feature work
    - 매우 긴 rendering과 optimization 시간
      - A100 GPU 8대를 이용하여 2일간 학습
      - A100 GPU 4대를 이용하여 8시간 inference
      - Voxel 기반의 표현과 결합하여 training/inference 시간 단축 필요
    - 대각선 방향의 시점의 경우 렌더링 품질이 저하됨
      - 고급화된 데이터 기반 prior를 사용하는 canonical space-based method와 결합 필요
    - Source view에서 충분한 대응이 보이지 않으면 렌더링된 콘텐츠가 비현실적임
      - 일반적으로 5개 미만의 source view의 경우 충분하지 않음
      - 고 품질의 렌더링 결과를 주는 Source view 선택 알고리즘에 대한 연구 필요

감사합니다