Neural Scene Chronology

Reconstructing a photo-realistic time-varying 3D model in **CVPR2023**

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

Presented By
Hosung Son

Contents

- Introduction
  - Scene chronology
  - Problem setting
  - Dataset
- Methods
  - 4D reconstruction from Internet Photos
  - Step function encoding
- Results
- Conclusion

Introduction

- Scene chronology
  - 특정 장면의 시간적인 연대(chronology) 특징을 학습하고, 학습되지 않은 임의의 시간에 대하여 photo-realistic한 장면 이미지를 생성
  - 4D(3D space & 1D time) 모델링 기반 chronology 방법론은 ECCV 2014 논문이 시초
    - 해당 논문은 연대에 따른 서로 다른 영상들의 이미지 특징을 기반으로 visibility-graph를 생성하고, Plane-Time Clustering을 위해 RANSAC을 적용하여 patch 단위로 임의 연대에 대한 영상을 렌더링함

      → Neural network을 사용하지는 않았음
    - 목표 영상을 3차원 point cloud로만 구현하며, 임의 시점에 대한 영상 합성은 불가
  - 반면, Neural scene chronology [CVPR2023] 은 Neural radiance field 에 기반하여 chronology, view, illumination synthesis를 모두 렌더링할 수 있는 고성능 모델임

5pointz 데이터 셋 상의 서로 다른 연대에 촬영된 영상 비교 예시

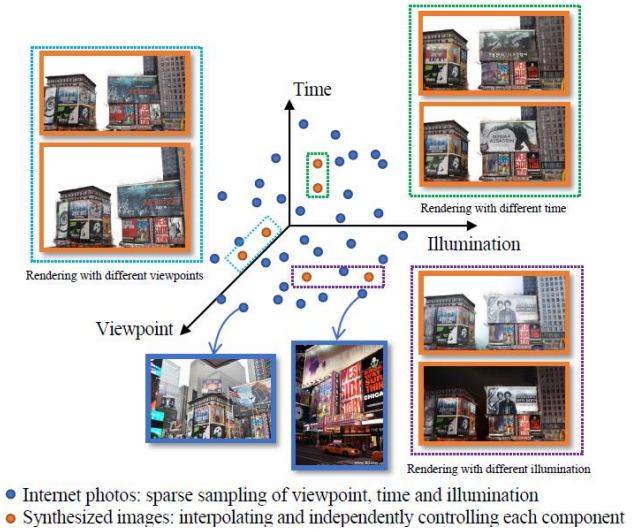
렌더링 된 임의 연대에 대한 *5pointz* 데이터 셋 예시

Introduction

- Problem setting
  - 목표
    - 인터넷 공간 상의 여러 연대 영상들을 학습하여 임의 연대에 대한 시점 합성 모델 구현

      →Temporal signal modelling

      - 특정 연대, 조명 값에 대하여 합성 가능
      - **연대, 시점, 조명의 독립적 제어가 가능한 렌더링 구현이 목표**
      - 각 특성들은 서로 복합적으로 엮여 있기에 개별적 제어가 어려움
  - 데이터 특성
    - 데이터 공간 좌표축
      - 시점(viewpoint), 연대(Time), 조명(illumination)
    - Blue point
      - 서로 다른 연대에 취득된 인터넷 영상
    - Orange point
      - 합성된 시점 영상



Problem illustration

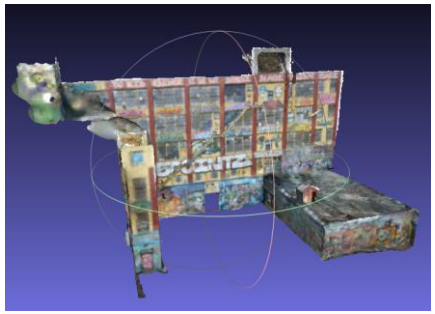
Introduction

- Dataset
  - Retrieve
    - 현실 세계에 존재하는 특정 장면(e. g. *Times square*)을 촬영한 영상을 인터넷에서 취득
  - Calibration
    - 취득된 영상에 대해 SfM tool을 사용하여 카메라 포즈 취득 (COLMAP)
    - Feature point matching 기반이므로, 모든 이미지들이 COLMAP으로부터 카메라 포즈가 취득되는 것은 아님
  - Region of selection
    - 취득된 영상의 개수가 매우 크기 때문에 implicit representation 에 제약이 있을 수 있음
    - 관심 영역 부근(건물의 외벽이 잘 나타난)의 영상들을 추려서 렌더링 학습에 사용

	Times Square	*Akihabara*	*5Pointz*	*The Met*
# Retrieved images	289,794	105,445	23,628	186,663
# Calibrated images	29,629	13,671	6,503	2,184
# Selected images	5,965	1,078	3,521	2,127

Dataset statistics

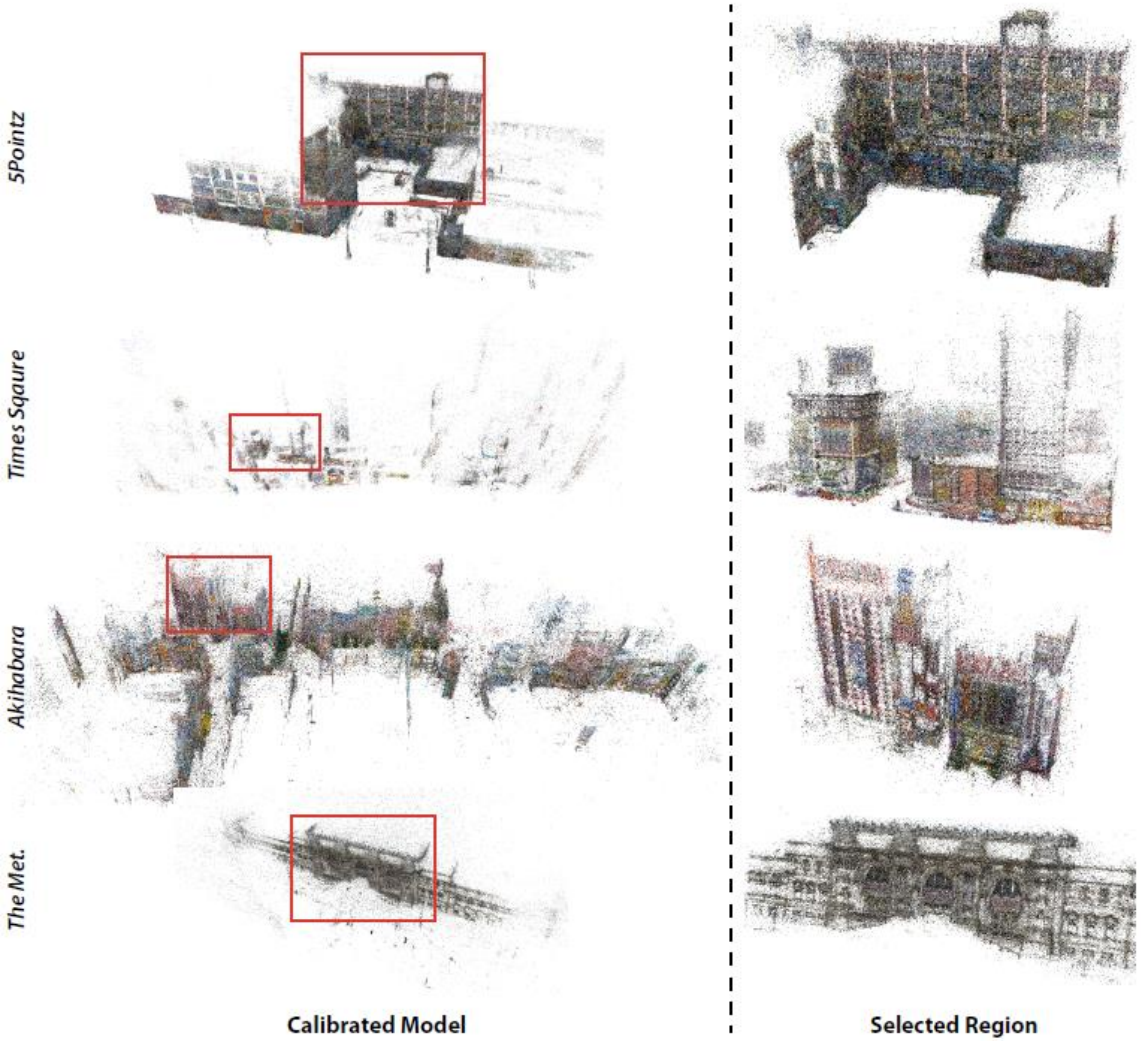
***5pointz* 데이터 셋 예시**



***5pointz* 데이터 셋 point cloud**

Introduction

- Dataset



Methods

- 4D reconstruction from Internet Photos
  - MLP 기반 NeRF 렌더링 방식에 기반
    - 총 4 종류의 입력 데이터를 통해 color 와 density 추정

$$\mathbf{c}, \sigma = \mathbf{F}(\mathbf{x}, t_i, \boldsymbol{l}_i, \mathbf{d})$$

$\boldsymbol{l}_i$: Illumination encoding
$\mathbf{x}$: 3D position
t_i: Time point
$\mathbf{d}$: ray direction

  - 문제 단순화를 위해, 렌더링 목표의 geometry 는 일정하며, 외부의 appearance 만 변화한다고 가정
    - 즉, 시점 방향, 연대, 조명에 따라 기하적 구조가 변화하지는 않는 상태
    - 위의 가정을 통해, 기존 NeRF 모델을 두 가지 형태로 분리

$$\mathbf{v}, \sigma = \mathbf{F}_{geo}(\mathbf{x})$$

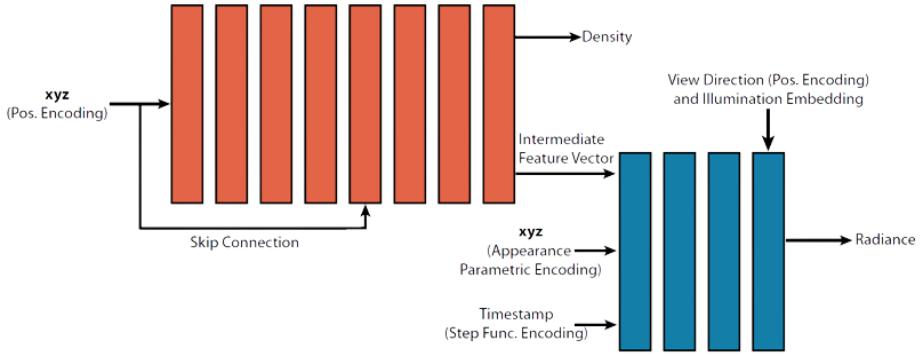
$\mathbf{v}$: Intermediate geometry feature vector

$$\mathbf{c} = \mathbf{F}_{app}(\mathbf{x}, \mathbf{v}, t_i, \boldsymbol{l}_i, \mathbf{d})$$

  - Rendering $\hat{\mathbf{C}}(\mathbf{r}, t_i, \boldsymbol{\ell}_i) = \sum_{k=1}^{K} T(s_k)\alpha(\sigma(s_k)\delta_k)\mathbf{c}(s_k, t_i, \boldsymbol{\ell}_i)$

where $T(s_k) = \exp\left(-\sum_{k'=1}^{k-1}\sigma(s_{k'})\delta_{k'}\right)$

  - Loss function $\mathcal{L} = \sum_{(\mathbf{r},i)\in\Omega} \|\mathbf{C}_i(\mathbf{r}) - \hat{\mathbf{C}}(\mathbf{r}, t_i, \boldsymbol{\ell}_i)\|_2^2$



제안 모델 구조

Each layer of the MLP has 256 neurons (channels).
ReLU is used as the activation function

Methods

- 4D reconstruction from Internet Photos
  - Appearance parametric encoding (APE) from NSVF
    - 도입 배경
      - Appearance MLP를 학습할 때 입력으로 사용되는 3d position (**x**) 에 encoding을 수행
      - Neural scene chronology 의 경우 학습해야 할 scene의 scale과 image 수가 많기 때문에, 그에 따른 model size 및 model capacity가 증가할 필요가 존재
      - 이때, overhead 발생으로 인한 학습 소요 시간이 증가될 수 있으므로 해결 방안 필요
    - 가정
      - 데이터 셋의 특성이 건물 위주이므로, 장면 표현이 Manhattan world 가정을 만족
    - 인코딩 과정

$\mathbf{E} \in D \times D \times C$

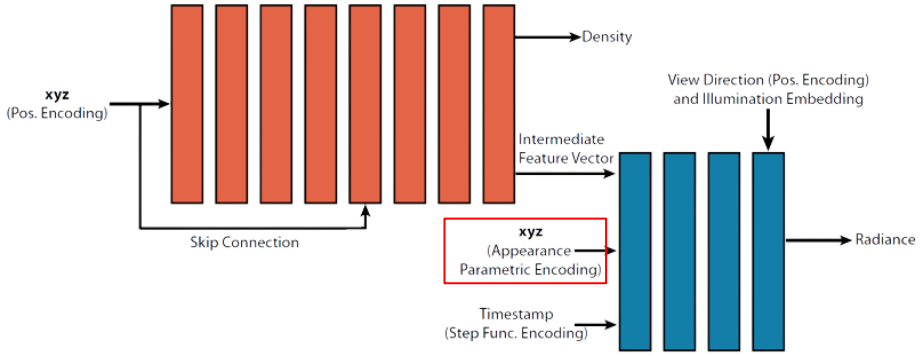
      - Feature planes 설정 ($\mathbf{E}_{xy}, \mathbf{E}_{yz}, \mathbf{E}_{xz}$)
      - Projected point from position **p**

$$p_{xy} = proj(\mathbf{p}, \mathbf{E}_{xy})$$

      - Fetched feature **f** & APE for **p**

$$\mathbf{f}_{xy} = \mathbf{Linearinterp}(\mathbf{E}_{xy}, p_{xy})$$

$$\mathbf{APE(p)} = \mathbf{concat}(\mathbf{f}_{xy}, \mathbf{f}_{yz}, \mathbf{f}_{xz})$$



제안 모델 구조

Each layer of the MLP has 256 neurons (channels).
ReLU is used as the activation function

Methods

- 4D reconstruction from Internet Photos
  - Appearance parametric encoding (APE) from NSVF
    - 도입 배경
      - Appearance MLP를 학습할 때 입력으로 사용되는 3d position (**x**) 에 encoding을 수행
      - Neural scene chronology 의 경우 학습해야 할 scene의 scale과 image 수가 많기 때문에, 그에 따른 model size 및 model capacity가 증가할 필요가 존재
      - 이때, overhead 발생으로 인한 학습 소요 시간이 증가될 수 있으므로 해결 방안 필요
    - 가정
      - 데이터 셋의 특성이 건물 위주이므로, 장면 표현이 Manhattan world 가정을 만족
    - 인코딩 과정

$\mathbf{E} \in D \times D \times C$

      - Feature planes 설정 ($\mathbf{E}_{xy}, \mathbf{E}_{yz}, \mathbf{E}_{xz}$)
      - Projected point from position **p**

$$p_{xy} = proj(\mathbf{p}, \mathbf{E}_{xy})$$

      - Fetched feature **f** & APE for **p**

$$\mathbf{f}_{xy} = \mathbf{Linearinterp}(\mathbf{E}_{xy}, p_{xy})$$

$$\mathbf{APE(p)} = \mathbf{concat}(\mathbf{f}_{xy}, \mathbf{f}_{yz}, \mathbf{f}_{xz})$$

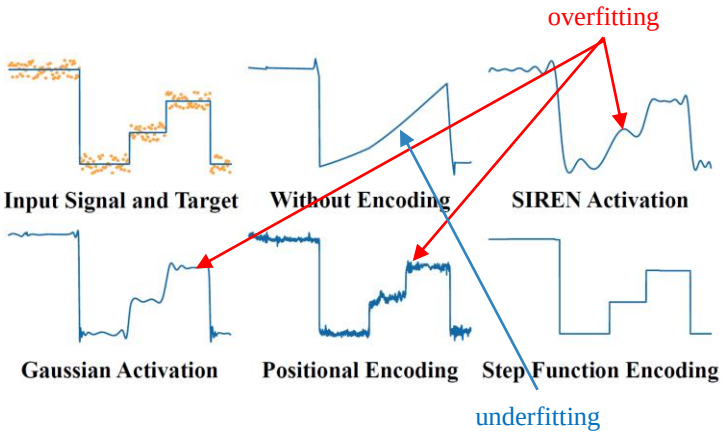
Figure 7. **Ablation on the appearance parametric encoding.** As illustrated above, using a parametric encoding significantly affects the rendering quality.

Appearance Parametric Encoding의 효과

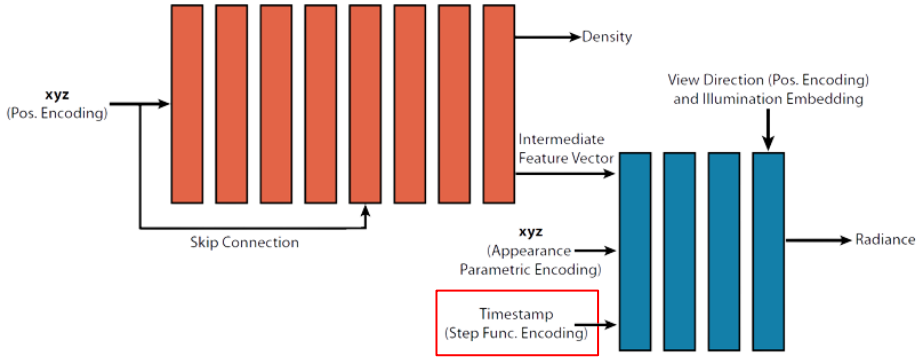
Methods

- Step function encoding for Time Input
  - 제안 배경
    - Chronology 데이터 셋의 시간에 따른 신호 분포는 오른쪽 하단의 orange point와 같이 일정한 부분이 존재함
    - MLP 는 왼쪽 상단의 clean blue curve를 복원할 수 있어야 함
    - 기존 MLP 에 적용되는 여러가지 activation function과 encoding 방식은 이러한 특성을 모델이 반영하도록 학습시키는데 한계가 존재

→ 제안된 Step Function Encoding 은 overfitting 없이 우수한 결과를 낼 수 있음



Constant 1D 신호에 대한 제안 방법론의 MLP 피팅 결과 비교



제안 모델 구조

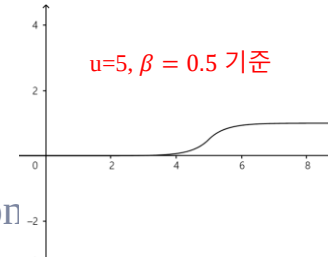
Each layer of the MLP has 256 neurons (channels).
ReLU is used as the activation function

Methods

- Step function encoding for Time Input
  - Step function encoding
    - MLP에 입력 전 time 값에 encoding 수행

standard version

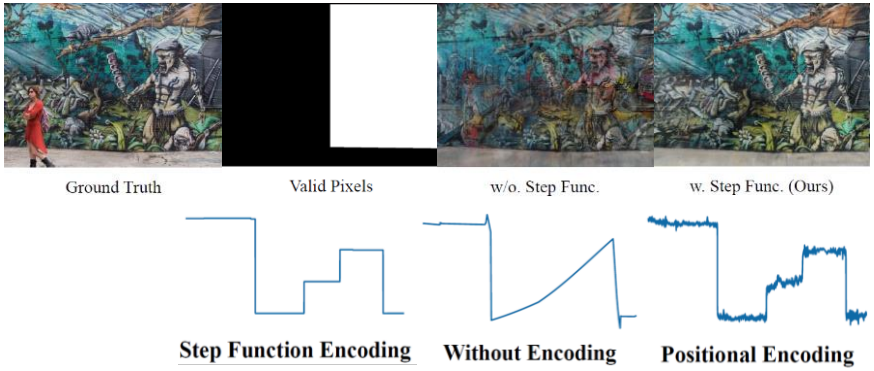
$$h(t) = \begin{cases} 0 & if \quad t \le u \\ 1 & if \quad t > u \end{cases}$$



differentiable version

$$\bar{h}(t) = \begin{cases} \frac{1}{2}\exp\left(\frac{t-u}{\beta}\right) & if \quad t \le u \\ 1 - \frac{1}{2}\exp\left(\frac{-(t-u)}{\beta}\right) & if \quad t > u \end{cases}$$

u, β : learnable parameter



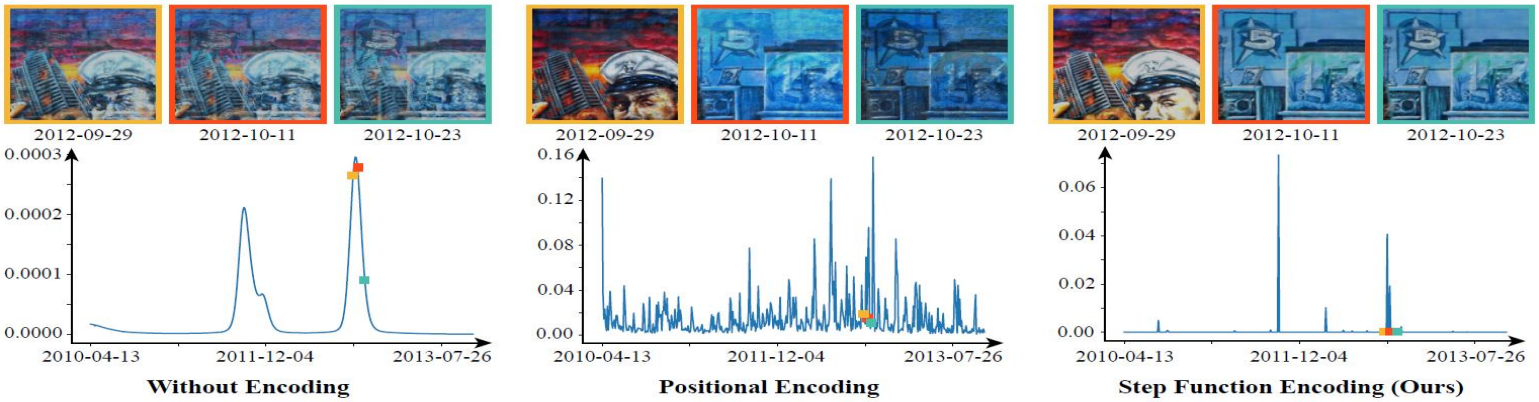
Time encoding 방식에 따른 Chronology 렌더링 비교

```python
class Encoder(nn.Module):
    def __init__(self, hard_forward=True, **kwargs):
        super().__init__()
        input_ch = kwargs['input_dim']
        output_ch = kwargs['output_dim']
        init_val = kwargs['init_val']
        self.hard_forward = hard_forward
        self.mean = nn.Parameter(torch.rand(output_ch))
        self.beta = nn.Parameter(torch.ones(output_ch) * init_val)
        self.input_ch = input_ch
```

```python
def forward(self, x):
    mean = self.mean[None]
    beta = self.beta[None]
    output = x - mean
    msk = output <= 0.
    output[msk] = 0.5 * torch.exp(output[msk] / torch.clamp_min(torch.abs(beta.repeat(len(msk), 1)[msk]), 1e-3))
    output[~msk] = 1 - 0.5 * torch.exp(- output[~msk] / torch.clamp_min(torch.abs(beta.repeat(len(msk), 1)[~msk]), 1e-3))
    if self.hard_forward:
        msk = output <= 0.5
        output[msk] = 0. + output[msk] - output[msk].detach()
        output[~msk] = 1. + output[~msk] - output[~msk].detach()
    return output
```

Methods

- Step function encoding for Time Input



Encoding 방식에 따른 인접 frame 간 MSE 비교
(가로축: 시간 변화, 세로축: 인접 프레임 간 MSE 값)

- 제안 방법론의 chronology 장면 표현 학습 효과
  - PE를 사용하지 않을 경우
    - 인접 프레임과의 MSE가 특정 시간대에서 점진적으로 변화하는 모습을 보임
  - PE를 사용할 경우
    - PE 사용하지 않을 경우와 비교해 더욱 인접 프레임과의 차이가 noisy하게 표현됨
  - SFE를 사용할 경우
    - 시간 변화에 따른 특징 변화 지점에 대해 집중적으로 높은 품질 영상을 표현 가능

Methods

- Step function encoding for Time Input
  - 시점 합성 품질 및 시간적 안정성 정량 평가
    - 평가 지표

	Act. func. / Freq. / Dim.	Temporal stability Mean ↓	Temporal stability Entropy ↓	View synthesis quality PSNR ↑	View synthesis quality SSIM ↑	View synthesis quality LPIPS ↓
w/o. Time	-	N/A	N/A	18.18	0.572	0.461
w/o. Encoding	-	0.116	5.314	20.54	0.719	0.296
Learned Latent	-	11.04	5.590	20.95	0.731	0.291
Activation	SIREN [49]	0.101	5.552	20.54	0.696	0.330
	Gaussian [41]	**0.088**	5.512	20.28	0.704	0.315
Positional Encoding	5	0.247	4.936	20.71	0.731	0.288
	10	5.657	5.795	20.57	0.724	0.294
	15	9.995	5.777	20.64	0.721	0.301
Step Func. Encoding	8	0.102	**1.602**	20.52	0.728	0.289
	16	0.147	2.213	**21.32**	**0.745**	**0.274**
	24	0.190	2.625	21.09	0.734	0.282
	32	0.217	2.806	21.10	0.738	0.281

      - Temporal stability
        - ✓Mean: 인접 프레임과의 MSE 평균
          - illumination 고정하여 video 렌더링
          - 추출된 video 프레임을 time 축에 대하여 인접 frame 간의 MSE 분포 계산
        - ✓Entropy: Normalized MSE-Time 분포에서의 엔트로피

$$E(P) = \sum_{x_i \in \mathcal{P}} -p(x_i) \log p(x_i)$$

특정 시점에서만 MSE가 크게 나타날 경우, 불확실성이 작으므로 엔트로피가 낮음
전반적으로 MSE가 noisy하게 분포할 경우 불확실성이 크므로 엔트로피가 큼

      - View synthesis quality
        - ✓PSNR: GT와의 MSE를 negative log scale로 표현 $PSNR = 10 \log_{10}\left(\frac{R^2}{MSE}\right)$ R: 영상의 픽셀 최대 값
        - ✓SSIM: GT와의 구조적 이미지 유사성까지 고려하여 표현

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}$$

uin8 영상에서의 default setting일 때

        - ✓LPIPS: Human vision 에서의 quality 평가 지표
          - 두 영상을 pretrained VGG network에 입력 후, 중간 layer feature의 유사성 표현

Methods

- Step function encoding for Time Input
  - 시점 합성 품질 및 시간적 안정성 정량 평가
    - 비교 방법론

	Act. func. / Freq. / Dim.	Temporal stability Mean ↓	Temporal stability Entropy ↓	View synthesis quality PSNR ↑	View synthesis quality SSIM ↑	View synthesis quality LPIPS ↓
w/o. Time	-	N/A	N/A	18.18	0.572	0.461
w/o. Encoding	-	0.116	5.314	20.54	0.719	0.296
Learned Latent	-	11.04	5.590	20.95	0.731	0.291
Activation	SIREN [49]	0.101	5.552	20.54	0.696	0.330
	Gaussian [41]	**0.088**	5.512	20.28	0.704	0.315
Positional Encoding	5	0.247	4.936	20.71	0.731	0.288
	10	5.657	5.795	20.57	0.724	0.294
	15	9.995	5.777	20.64	0.721	0.301
Step Func. Encoding	8	0.102	**1.602**	20.52	0.728	0.289
	16	0.147	2.213	**21.32**	**0.745**	**0.274**
	24	0.190	2.625	21.09	0.734	0.282
	32	0.217	2.806	21.10	0.738	0.281

Dim: step function 개수
→ scene transition 횟수보다 커야 함

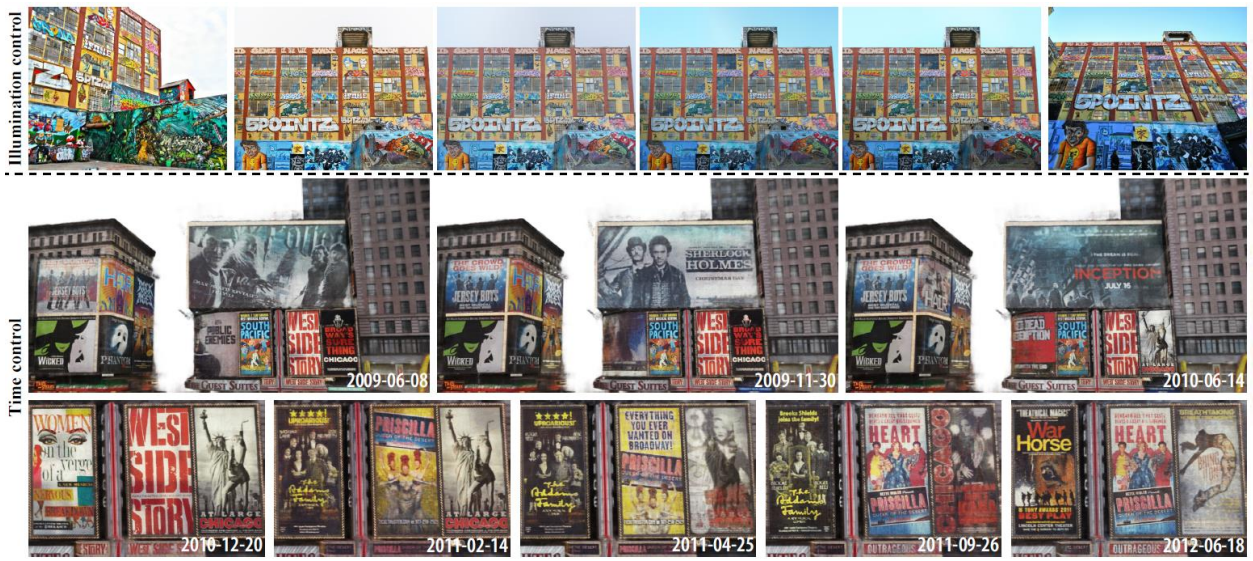
      - w/o. Time
        - ✓Time data 고려하지 않음
      - w/o. Encoding
      - Learned Latent
        - ✓Time 값을 학습된 latent code로 mapping 시킴
      - Activation
        - ✓SIREN: Time data에 대해 반복적으로 linear transformation과 sine activation(periodic function)을 수행 (from Sitzmann *et al.*[NeurIPS 2020])

$$\Phi(\mathbf{x}) = \mathbf{W}_n(\phi_{n-1} \circ \phi_{n-2} \circ \ldots \circ \phi_0)(\mathbf{x}) + \mathbf{b}_n, \quad \phi_i(\mathbf{x}_i) = \sin(\mathbf{W}_i\mathbf{x}_i + \mathbf{b}_i)$$

        - ✓Gaussian: Time 값을 Gaussian function에 통과시킴
      - Positional Encoding
        - ✓기존의 NeRF에서 사용된 fixed band length 의 sinusoidal function 집합
      - Step Function Encoding

Results

- 조명 및 시간 효과의 독립적 제어 정성 결과
  - 참조 이미지 입력 시, 해당 이미지의 조명 특성을 반영하여 렌더링 가능
  - 시간에 따른 변화만을 독립적으로 반영하여 렌더링하는 것도 가능



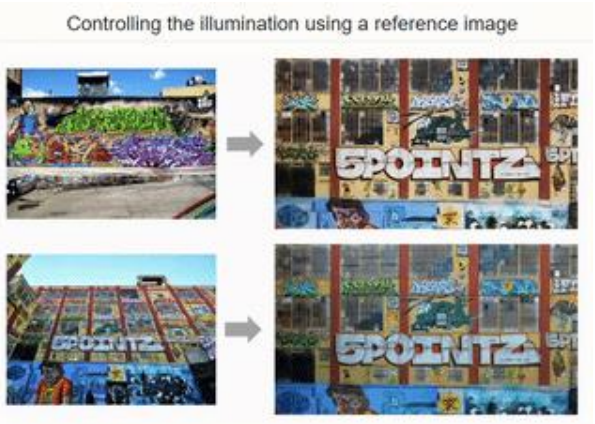


Figure 5. **Independent control of illumination effects and time.** The top row shows the results of interpolating the illumination embeddings of two real images of *5Pointz*, where the leftmost and rightmost images are real images, and all other images are rendered using our method. The bottom rows show the results of rendering scenes across time with fixed illumination embeddings (for *Times Square*).

reference image의 illumination 특성을 반영한 렌더링

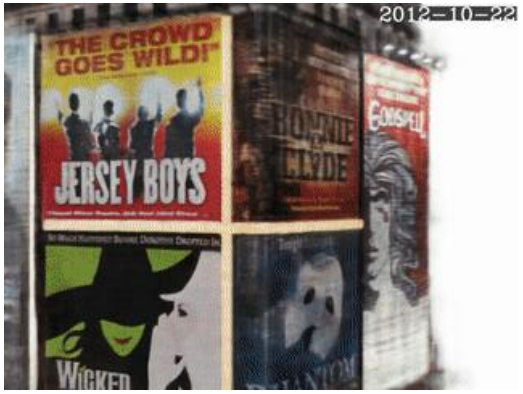
Results

- 정성 & 정량 결과
  - 과거 모델과 비교하여 시간적 안정성과 렌더링 결과 모두 가장 우수
    - NeRFW [CVPR2021], HaNeRF [CVPR2022]

Figure 6. **Qualitative comparison with the state of the art.** The three column images under *Input time: t_x* are rendered using timestamps t_x and the viewpoints of the left image. Our method renders high-quality images and produces plausible images when changing the input time.

	4D view synthesis	*Times Square* Entropy ↓	PSNR ↑	SSIM ↑	LPIPS ↓	*Akihabara* Entropy ↓	PSNR ↑	SSIM ↑	LPIPS ↓	*5Pointz* Entropy ↓	PSNR ↑	SSIM ↑	LPIPS ↓	*The Met* Entropy ↓	PSNR ↑	SSIM ↑	LPIPS ↓
NeRFW [29]	✗	N/A	16.59	0.820	0.211	N/A	17.41	0.853	0.164	N/A	17.52	0.545	0.500	N/A	23.21	0.881	0.159
HaNeRF [6]	✗	N/A	15.39	0.807	0.218	N/A	17.50	0.860	0.160	N/A	16.82	0.539	0.508	N/A	22.32	0.882	0.158
NeRFW-T	✓	4.990	18.71	0.847	0.190	5.483	19.16	0.874	0.140	5.768	19.41	0.611	0.418	4.923	23.83	0.875	0.164
HaNeRF-T	✓	4.929	17.36	0.844	0.189	5.565	18.39	0.873	0.140	5.881	17.90	0.585	0.445	4.943	22.56	0.881	0.156
Ours	✓	**3.122**	**20.87**	**0.894**	**0.132**	**2.482**	**20.31**	**0.902**	**0.101**	**2.213**	**21.32**	**0.745**	**0.274**	**2.399**	**24.07**	**0.895**	**0.129**

Table 3. **Quantitative comparison with the state of the art.** We augment NeRF-W and HaNeRF to take time as input (*-T). Our method outperforms prior methods across all metrics, demonstrating that our method can better handle such time-varying Internet collections.



실제 시간에 따른 변화는 의미론적 이므로, rough 한 특징만 추정하여 렌더링하는 것이 아니라 온전한 특징에 대하여 일정함을 유지하는 것이 필요하다.

Conclusion

- Conclusion
  - Geometry와 appearance 를 분리하여 렌더링 학습하는 구조 제안
    - 데이터 셋 특성에 따른 overhead를 줄이기 위해 appearance parametric encoding 적용
  - Step Function Encoding(SFE) 를 적용하여 Target task 에 적합한 Encoding 방식 제안
    - 연대에 따라 feature 특성이 구간 별로 일정한 데이터 특성 고려
    - 기존의 Fixed sinusoidal function 으로 이루어진 PE를 대체하여, 미분 가능한 Step function 형태를 제안하였으며, 함수의 파라미터는 학습 가능한 요소로써 구현
  - 기존의 in the wild 한 영상에 맞춘 NeRF 모델과 비교하여 가장 우수한 성능 확보
- Limitations and future work
  - 학습 과정에서 부정확한 Time stamp은 방해요소로 남음
    - 특히, 인터넷 영상은 정확한 촬영 시각을 알기 어려움
  - 건물 외벽이 비디오일 경우(*Times square*) 빠른 빈도로 변화가 있으므로, 이미지 컬렉션에 어려움이 존재하여 렌더링에 제약이 존재

Thanks!