NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis

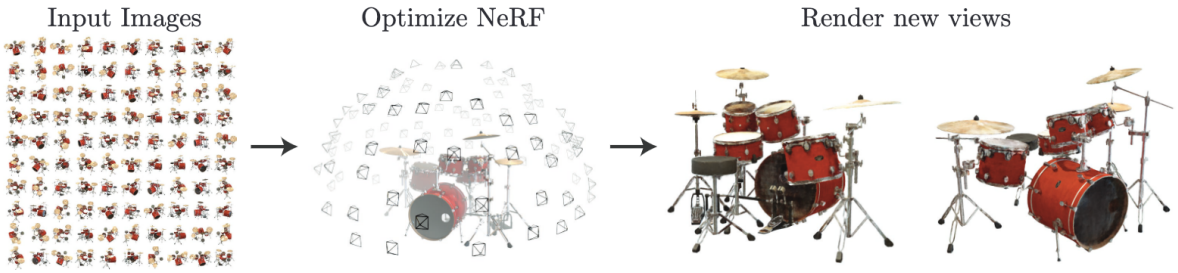
Introduction

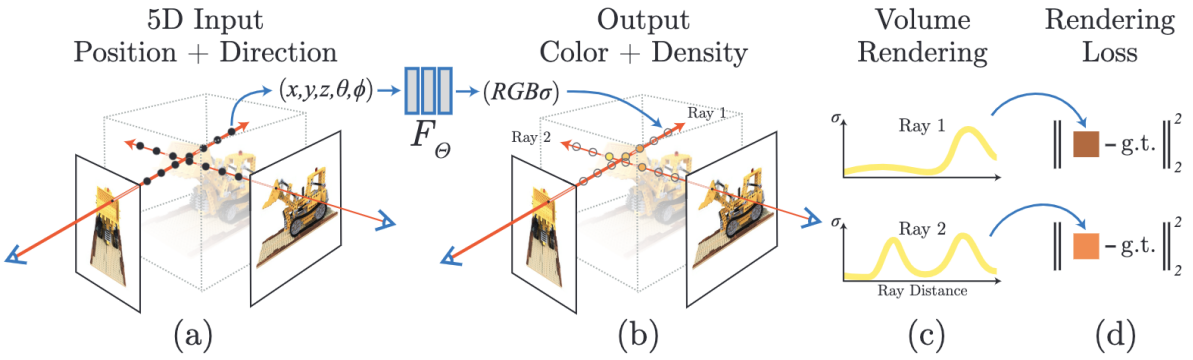
본 논문은 5차원 데이터 (x, y, z, θ, φ)를 입력받아 해당 시점에서 바라봤을 때 객체에서 추출되는 radiance와 density를 추출하는 함수를 고안했다.

- 이 때 radiance는 '(x, y, z)에서 (θ, φ) 방향으로 방출되는 광원'이고 density는 '(x, y, z)를 지나는 레이저(가상의 레이저)에 축적되는 광원을 제어하는 **differential opacity** 같이 작동하는' 것이다.

본 논문에서 제안한 NeRF를 수행하기 위한 과정은 다음과 같다.

1. 3D point의 sampled set 생성하기
2. 생성한 sampled set과 sampled set에 해당하는 2D viewing direction을 MLP에 넣어 해당 위치에서 바라본 객체의 color와 density 얻기
3. classical volume rendering techniques으로 획득한 color와 density를 2D 이미지에 축적하기





본 눈문은 NeRF를 수행하는 과정이 미분이 가능하다고 말했다. 즉, 해당 지점에서 실제로 관측된 이미지(scene)와 MLP로 만들어진 '추측된' 이미지 사이의 에러를 계산해 경사하강법으로 MLP를 최적화 시킬 수 있다는 것이다. 이렇게 여러 관점에서 관측된 이미지의 에러를 줄일 수 있으면 새로운 관점에서 관측될 이미지를 더 정확히 관측할 수 있다.

본 논문은 복잡한 scene을 표현하기 위한 NeRF를 최적화하는 basic implementation이 좋지 못한 성능을 보이며 더 많은 sampled set이 필요하다는 사실을 알아냈다. 그래서 **positional encoding**을 이용해 (x, y, z, θ, φ) 데이터를 변환하고 NeRF에 쓰이는 MLP가 더 높은 주파수도 표현할 수 있게 만들었다. 그리고 **hierarchical sampling procedure**을 도입해 요구되는 쿼리의 숫자를 감소, 고주파의 scene을 표현할 때 필요한 sampled set를 적절한 개수만큼 선별하게 만들었다.

본 논문의 접근법은 'NeRF는 volumetric representation의 이점을 상속 받았다. NeRF는 현실세계에 존재하는 복잡한 모양의 객체들을 표현할 수 있고 우리 눈에 사영된 이미지들을 사용하여 **gradient-based optimization**을 수행할 수 있다. 그리고 복잡한 객체의 scene을 고해상도로 모델링할 때 필요한 cost가 discretized voxel grids로 모델링 할 때 필요한 cost보다 훨씬 적다는 이점도 가지고 있다. 즉, 적은 용량만 있어도 3D모델의 scene을 생성할 수 있다. 3D 파일들이 많은 용량을 가지고 있는 것을 생각했을 때 상당히 큰 장점이다.

본 논문은 NeRF의 contribution을 다음과 같이 정리했다.

1. MLP 네트워크로 continuous scenes with complex geometry and materials을 표현하는 방식을 제안했다.

2. classical volume rendering techniques 기반의 미분 가능한 렌더링 방식을 제안하고 MLP는 렌더링 과정에서 얻은 값과 실제 데이터 사이의 오차를 미분하여 parameter 학습을 수행한다.
3. 5차원 입력 데이터를 더 고차원 데이터로 mapping하는 positional encoding을 제안하고 positional encoding 덕분에 NeRF가 고주파의 scene을 성공적으로 표현할 수 있게 만들었다.

그리고 본 논문은 NeRF가 논문을 쓸 당시에 가장 성능이 좋았던 모든 view synthesis 방식들보다 더 좋은 성능을 보여줬다고 한다.

Neural Radiance Field Scene Representation

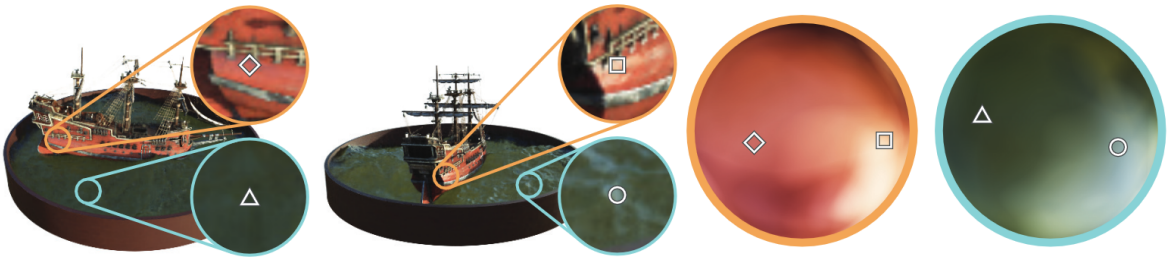
본 논문은 입력데이터로 (x, y, z, θ, φ), 다시 말해 3D 좌표 **X** = (x, y, z)와 **X**에서 바라보는 방향 (θ, φ)을 받아 color (r, g, b)와 volume density σ를 출력하는 함수를 만들었다. 이 때 특정 위치 (x, y, z)에서 객체를 바라보는 방향 (θ, φ)을 직교 좌표계 (x, y, z)로 구성된 unit vector **d**로 표현한다. 본 논문은 5차원 데이터 (x, y, z, θ, φ)로 scene을 나타내는 방식을 (**X**, **d**) → (**c**, σ)로 매핑하는 MLP로 구현하였으며 이 MLP의 parameter Θ를 학습시켜 더욱 현실에 가까운 scene을 만든다.

보통 네트워크를 설계하고 학습시키다 보면 특정 케이스에서는 성능이 잘나오지만 다른 특정 케이스에서는 성능이 안나올 때가 있다. 특정 상황에 많이 편향되게 학습이 진행되기 때문이다. NeRF가 수행하는 view synthesis에서는 '특정 시각에서 관측되는 scene만 잘 예측하는 것'으로 볼 수 있다.

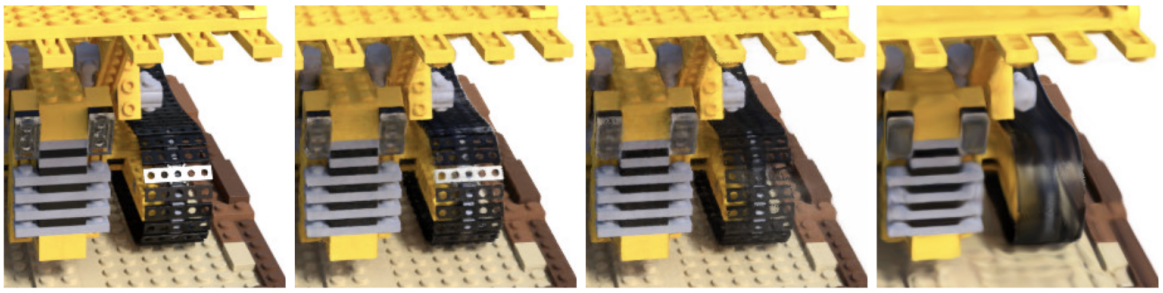
그래서 저자는 이러한 문제를 다음과 같은 방식으로 해결했다.

1. **X**만 가지고 volume density σ을 예측하게 만들기 : 8개의 fully-connected layer(256개의 뉴런을 가지고 있음)로 구성된 MLP가 X를 입력받아 σ와 256차원의 feature vector를 출력한다.
2. **X**와 **d**를 모두 사용해 **c**를 예측하게 만들기 : MLP로 예측한 256차원의 feature vector를 (θ, φ)와 합치고(concatenate) 128개의 뉴런을 가진 fully-connected layer에 넣어 view-dependent RGB를 획득한다.

여기서 나온 MLP, fully-connected layer는 모두 ReLU를 활성화 함수로 사용한다.



(a) View 1 (b) View 2 (c) Radiance Distributions



Ground Truth Complete Model No View Dependence No Positional Encoding

Volume Rendering with Radiance Fields

5D neural radiance field는 공간의 점에서 volume density와 directional emitted radiance로 scene을 표현한다. 본 논문은 classical volume rendering 기법을 사용하여 scene을 통과하는 ray의 색을 랜더링한다. volume density σ(x)는 X에 존재하는 아주 작은 입자에서 사라지는 레이저가 미분이 가능할 확률로 해석할 수 있다. 이러한 성질을 가지고 있는 가상의 레이저, 다시 말해 camera ray $r(t) = o + td$ 와 ray의 구간을 결정하는 t_n, t_f를 이용하여 camera ray와 충돌하는 객체의 일부분을 렌더링 할 때 사용할 색상을 다음의 식으로 결정할 수 있다.

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \mathbf{d})dt\,, \text{ where } T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds\right)$$

여기서 T(t)는 ray가 t_n부터 t까지 객체를 통과하며 축적된 투과율을 말한다. 발사하는 가상의 레이저가 객체의 여러 점들을 통과할 것이다. 그 때 만나는 여러 점에서 구한 투과율을 축적하는 것이죠. 이를 달리 말하면 'camera ray가 t_n부터 t구간에 있을 때 객체와 접촉하지 않을 확률'로 해석할 수 있다. NeRF를 이용해 scene을 생성하려면 desired virtual camera가 발사하는 가상의 광선이 접촉하는 픽셀들을 가지고 C(**r**)을 측정한 값이 필요하다.

본 논문은 quadrature을 사용하여 연속적인 적분을 추정한다. Deterministic quadrature, 다시 말해 적분할 구간을 사각형으로 근사화 후 다 합치는 적분 기법이 있다. 이는 voxel grid를 렌더링 할 때 많이 쓰는 기법이다. 그런데 NeRF에서 이 방식을 사용하면 MLP가 fixed discrete set of location의 값을 가지고 discrete한 적분을 수행하기 때문에 성능이 제한된다. 적분할 때 쓰는 값의 개수가 항상 같기 때문에 가상의 레이저가 접촉하는 좌표가 객체가 아닌 빈 공간일 수도 있다. 즉, 가상의 camera ray가 접촉하는 모든 점에서 sampling한 t_i들을 가지고 적분을 수행하기 때문에 불필요한 pixel도 적분 범위에 들어가 성능이 제한된다. 그래서 본 논문은 다른 방법을 썼으며 그 방식의 이름이 바로 stratified sampling approach이다. stratified sampling approach는 C(**r**)의 적분 구간으로 사용했던 $[t_n, t_f]$를 일정한 N개의 구간으로 나누고 각 구간에서 임의로 하나씩 뽑아 discrete한 적분의 구간으로 사용한다.

$$t_i \sim \mathcal{U}\left[t_n + \frac{i-1}{N}(t_f - t_n),\ t_n + \frac{i}{N}(t_f - t_n)\right]$$

이렇게 적분에 쓸 값을 선별하면 continuous position을 처리하는 MLP에서 얻은 값에서 선별하기 때문에 유효한 값들에서 sampling을 수행할 수 있다. 그렇게 sampling한 t_i들을 가지고 적분을 수행하면 더 높은 성능이 나온다. 이 때 수행하는 적분 역시 quadrature rule을 따르는 discrete한 적분이며 식은 다음과 같다.

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} T_i(1 - \exp(-\sigma_i\delta_i))\mathbf{c}_i\ ,\ \text{where } T_i = \exp\left(-\sum_{j=1}^{i-1}\sigma_j\delta_j\right)$$

이 식으로 구한 C(**r**)은 trivially differentiable하며 alpha를 $\alpha_i = 1 - exp(-\sigma_i$ x $\delta_i)$와 합치기 때문에 값이 감소한다.

Optimizing a Neural Radiance Field

이번 장에는 앞서 설명한 Positional Encoding과 hierarchical sampling procedure에 대해 자세히 설명되어 있다. 이 둘은 NeRF가 고주파의 복잡한 scene을 표현할 수 있게 도와주는 요소들이다. 이 둘의 역할을 다시 한 번 간단히 정리하자면 다음과 같다.

1. Positional encoding
- MLP가 고주파 데이터를 처리할 수 있게 도와준다.
- 입력 데이터에 적용한다.
2. Hierarchical sampling procedure
- NeRF가 고주파 scene을 효율적으로 sample할 수 있게 만들어준다.

Positional encoding

저자는 네트워크 F_θ가 (x, y, z, θ, ϕ) 좌표를 바로 처리하였을 때 고주파 영역의 렌더링 성능이 크게 떨어진다는 것을 발견했다. 네트워크가 저주파를 처리하는 함수로 학습되기 때문에 이러한 현상이 일어나기 때문이다. 그리고 고주파 함수로 입력 데이터를 더 고차원 데이터로 매핑하고 네트워크에 입력값으로 넣으면 고주파 영역의 처리 성능이 향상된 방향으로 학습되는 것도 발견했다.

본 논문은 이 점을 NeRF에 적용했다. 네트워크 $F_\theta = F_\theta^{'} \circ \gamma$ 로 재구성 하였을 때 성능이 매우 크게 향상된다는 점을 발견했다. 여기서 $F_\theta^{'}$는 simply a regular MLP이며 γ는 R 차원에서 R^{2L}차원으로 매핑하는 함수이며 식을 자세히 적으면 다음과 같다.

$$\gamma(p) = \left(\sin\left(2^0\pi p\right), \cos\left(2^0\pi p\right), \cdots, \sin\left(2^{L-1}\pi p\right), \cos\left(2^{L-1}\pi p\right)\right)$$

$\gamma(\cdot)$는 입력 데이터 $X = (x, y, z)$의 각 원소 x, y, z와 X를 통과하는 camera ray의 unit vector d의 원소 x, y, z 모두 적용하여 해당 값들을 [-1, 1] 범위에서 normalize합니

다. Positional encoding은 transformer를 읽어보신 분들에게 매우 익숙한 단어일 것이다. 그러나 여기서 쓰는 것과는 조금 다르며 둘을 비교해보면 다음과 같다.

1. transformer의 positional encoding : 입력값의 순서에 대한 개념이 없는 네트워크에 입력값으로 들어갈 토큰에 순서라는 개념을 부여하기 위해 사용된다.
2. NeRF의 positional encoding : 연속적인 입력 데이터의 좌표들을 좀더 고차원 데이터로 매핑해 MLP가 좀더 쉽게 고차원 성분, 즉 scene에 있는 테두리들을 좀더 쉽게 근사할 수 있게 만들기 위해 사용된다.

Hierarchical volume sampling

본 논문은 원래 camera ray에서 N개의 포인트를 임의로 뽑은 후 렌더링에 사용하려고 했는데 이렇게 하니 비효율적이다. 왜냐하면 camera ray가 통과하는 공간은 객체뿐만 아니라 아무것도 없는 공간도 포함되어 있기 때문이다. 도움이 안되는 것들도 렌더링에 사용하니 좋은 결과가 나오기 힘들다. 그래서 생각한게 마지막 렌더링에서 예측되는 효과에 비례해 포인트를 sampling하는 방식인 Hierarchical volume sampling이다.

본 논문은 NeRF에 사용할 네트워크를 하나가 아닌 두개로 설정했으며 각각 “coarse” 네트워크, “fine” 네트워크라고 이름 지었다.

1. camera ray에서 stratified sampling을 사용해 N_c개의 구간에서 값을 하나씩 뽑는다. 그리고 해당 값을 가지고 $C(r)$을 계산한다. (이 때 “coarse” 네트워크를 사용한다.)
2. 구한 값에서 얻은 정보를 통해 volume density가 상대적으로 높은 좌표를 위주로 값을 뽑는다.

그리고 이를 수행하기 위해서 alpha composited color를 구하는 식 $C_c(r)$을 다음과 같이 c_i의 weighted sum으로 재정의한다.

식 5에서 사용되는 weight w_i를 normalizing할 경우, w_i는 부분적으로 일정한 PDF가 된다.

다음으로 “fine” 네트워크를 사용하는 경우이다.

1. camera ray에서 inverse transform sampling을 사용해 N_f개의 위치에서 값을 하나씩 뽑는다.
2. “fine” 네트워크를 이용해 C_f(**r**)을 계산하는데 이 때 “coarse”네트워크에 넣었던 N_c개의 값도 같이 사용한다. 즉, “fine” 네트워크에 N_c + N_f개의 데이터를 입력값으로

넣는다.

이렇게 하면 visible content가 포함될 것으로 예측된다. 즉, 실제 객체가 존재할 영역에 있을 좌표를 더 많이 사용하게 된다. 또한, 허공이 아닌 실제 객체를 렌더링에 사용하는 비중이 더 높아진다. 이는 곧 성능 상승으로 이어진다. 전체 적분 구간의 nonuniform discretization 으로 뽑힌 값들을 사용한다. 이런식으로 값을 뽑아서 렌더링에 사용하면 성능이 더 좋아진다.

Implementation details

NeRF를 구현하기 위해 사용한 세팅을 설명하는 장이다. 정확히는 학습 과정에 대해 자세히 설명한다.

1. **scene이 저장된 이미지, 해당 이미지를 촬영한 카메라, 카메라의 intrinsic parameters(카메라에 있는 이미지 센서의 크기 등 카메라의 고유 성질), scene bounds**로 구성된 데이터셋을 획득한다.
2. 학습 과정에서 매 epoch마다 camera ray를 만나는 데이터셋의 pixel들을 일부 선별, hierarchical sampling을 수행해 N_c개의 값을 넣어 “coarse” 네트워크에 넣어 값을 얻고 N_c + N_f개의 값을 “fine” 네트워크에 넣어준다.
3. volume rendering procedure을 수행해 camera ray로 바라본 지점의 색상을 렌더링한다.
4. “coarse” 네트워크를 사용해 렌더링한 값과 “fine” 네트워크를 사용해 렌더링한 값과 실제로 렌더링된 값을 가지고 loss를 구한다. 이 때 사용하는 loss는 total squared loss 이다.

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

여기서 R은 각 batch에 있는 camera ray의 집합을 말하며, C(**r)**, C_c(**r**), C_f(**r**)은 각각 ground truth, “coarse”로 얻은 값, “fine” 네트워크로 얻은 값을 나타낸다.

본 논문은 “fine” 네트워크와 “coarse” 네트워크를 사용해 loss를 구다. 그러나 “coarse” 네트워크만 학습을 수행한다. “coarse” 네트워크가 “fine” 네트워크에 있는 샘플들을 할당하는데 이용하기 위해서이다. 두개의 네트워크를 같이 쓴다고 했을 때 하나의 네트워크만 학습시켜도 원하는 값을 출력하게끔 학습시킬 수 있으니까 하나만 학습시키는게 학습 시간을 아낄 수 있다.