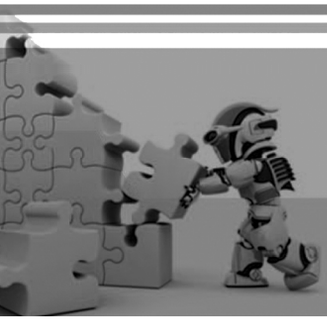

NeRF의 원리와 기술 동향

최준혁, 최성록(서울과학기술대학교 컴퓨터공학과)

1. 서론

우리는 다양한 방법으로 3차원 현실 공간에 존재하는 물체를 컴퓨터 환경에 옮기려 한다. 이미지는 3차원 공간을 카메라의 2차원 이미지 평면으로 투영한 데이터이고, 여러 장의 이미지로 부터 다양한 방법을 통해 물체의 3차원 모델을 생성하거나 필요한 정보를 추출할 수 있다.

3차원 복원(3D reconstruction)은 카메라에서 획득한 2차원 이미지들을 이용해 물체의 3차원 형상 모델을 역으로 생성하는 과정이다. 3차원 복원은 2차원 이미지로부터 3차원 정보를 얻어내는 과정이기 때문에 카메라 투영 모델은 물론 이미지를 촬영한 카메라 사이의 시점 관계도 알아야 한다. 이미지 사이의 시점 관계를 파악하면 3차원 공간의 한 점이 각 이미지의 어디에 투영되었는지를 알 수 있다. 3차원 복원 기술은 다양한 곳에 사용된다. 그래픽 디자이너들이 현실세계의 물체를 컴퓨터로 옮겨 리터치를 하고, 건설환경 분야에서는 주변 지형과 지물 또는 건물의 외형을 복원하여 필요한 분석과 설계를 수행한다. 또 로봇 분야에서는 자율주행을 위한 SLAM 및 지도작성, 장애물 및 환경인지, 그리고 로봇 팔을 이용한 자동화를 위한 물체인지 등에 사용한다.

최근 딥러닝을 이용한 3차원 복원 기술이 많은 관심을 받고 있고, 가장 대표적인 모델로 Neural Radiance Field(이하 NeRF)가 있다. 본 기고문에서는 NeRF의 원리와 이를 이용한 최근 기술 동향에 대해 소개한다. 우선 NeRF의 원리 쉽게 이해하기 위한 컴퓨터그래픽스에서 사용하는 배경 용어 먼저 간략히 살펴본다. 그리고 NeRF 여러 가지 요소들과 특징, 장단점을 알아본다. 마지막으로 NeRF를 응용하는 세 가지 주요 분야(3D reconstruction, visual SLAM, auto labeling)의 최근 대표 논문을 간략히 소개한다. 본 기고를 통해 독자들이 NeRF를 이용한 3차원 복원의 원리와 기술 흐름을 이해하고, 이를 바탕으로 자신의 연구와 개발에 도움이 되길 희망한다.

2. NeRF의 기본 원리

2.1. 컴퓨터그래픽스의 주요 용어

NeRF를 소개하기 앞서 먼저 컴퓨터그래픽스의 몇 가지 기본 용어를 먼저 리뷰하고자 한다.이러한 용어에 대한 이해를 통해 NeRF에 사용되는 여러 과정들과 파라미터들을 직관적으로 이해할 수 있다.

2.1.1. 카메라 투영 모델(Camera Projection Model)

카메라 투영모델은 3차원 공간상의 한 점(point)을 2차원 카메라 이미지에 투영되는 과정을 표현한 수학적 모델이다. 다양한 카메라 투영 모델이 존재하고, 핀홀 카메라 모델(pinhole camera model)이 가장 널리 사용된다. 핀홀 카메라 모델은 그림 1과 같이 매우 작은 구멍을 통해서 입사되는 빛이 이미지 평면에 투영되는 원리를 설명하는 모델이다. 핀홀 카메라에서 작은 구멍은 투영된 모든 빛이 지나는 점, 즉 초점(focal point) 이자 카메라의 중심(camera center)이다. 핀홀 카메라 모델은 렌즈의 왜곡이 없는 경우 아래와 같은 수식으로 표현된다.

$$\mathbf{x} = \mathrm{K}(\mathrm{R}\mathbf{X} + \mathbf{t}) = \mathrm{K}\mathbf{X}'$$

$\mathbf{X}$는 월드 좌표계(world coordinate)에서 표현된 3차원 점 [X

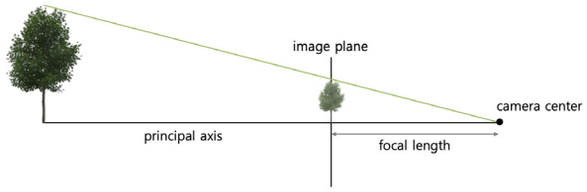

그림 1. 핀홀 카메라 모델에서의 이미지 평명(image plane)으로의 투영 예.

$,Y,Z]^{\mathrm{T}}$이고, **x**는 동차좌표(homogeneous coordinate)로 표현된 2차원 이미지 상의 점 $[u, v, w]^{\mathrm{T}}$이다. 동차좌표계의 점은 $(u/w, v/w)$과 같은 정규화를 통해 이미지 상의 점으로 표현된다. 카메라 행렬(camera matrix)는 초점거리(focal length) f나 주점(principal point) (c_x, c_y)과 같은 카메라 내부 파라미터(intrinsic parameters)를 포함하는 아래와 같은 3x3 행렬이다.

$$\mathrm{K} = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

카메라의 자세(camera pose), 즉 카메라의 방향(orientation)과 위치(position)는 각각 3x3 회전행렬 R과 3x1 벡터 **t**로 표현된다. 실제 R과 **t**는 월드좌표계에서 표현된 3차원 점 **X**를 카메라 좌표계에 대한 점 $\mathbf{X}'$으로 바꿔주는 점 변환(point transformation)이다. 카메라의 투영에 사용되는 변환 R과 **t**는 카메라의 위치와 시점에 따라 달라지기 때문에 외부 파라미터(extrinsic parameter)라고 한다.

2.1.2. 광선(Ray)

컴퓨터 그래픽스에서 광선(ray)은 우리의 시점(eye or camera)으로 입사 또는 방사되는 빛의 줄기를 의미하고, 특정 색의 속성을 갖고 이미지 평면(또는 뷰-포트)에 투영된다. 광선은 시점의 위치 **o**와 광선의 방향을 나타내는 벡터 **d**을 이용해 아래와 같이 표현할 수 있다.

$$\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$$

그림 1에서 볼 수 있듯이 광선 $\mathbf{r}(t)$는 시점 **o**에서 시작하여 **d** 방향으로 임의의 길이 만큼 쭉 뻗어나가며 이미지 평면에 부딪히는 벡터로 볼 수 있다. 렌즈 왜곡이 없는 핀홀 카메라 모델에서 카메라 시점의 위치와 이미지 평면 위의 점 **x**을 향하는 광선의 방향은 각각 아래와 같이 도출된다.

$$\mathbf{o} = -\mathrm{R}^{\mathrm{T}}\mathbf{t} \quad \text{and} \quad \mathbf{d} = \mathrm{R}^{\mathrm{T}}\mathrm{K}^{-1}\mathbf{x}$$

광선의 방향 벡터 **d**는 일반적으로 크기가 1인 정규화된 벡터로 변환하여 사용한다. 카메라 행렬과 카메라의 방향과 위치가 주어지면, 각 픽셀 **x**에 대한 광선의 벡터 $\mathbf{r}(t)$를 도출할 수 있다. 광선의 벡터는 컴퓨터그래픽스에서 2차원 영상을 생성하는 렌더링의 기본 정보로 사용된다.

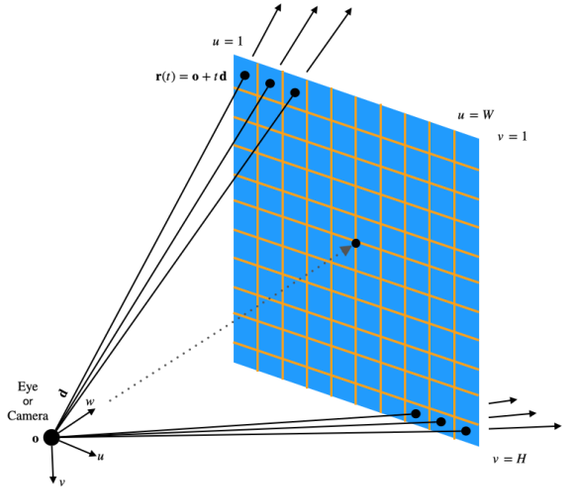

그림 2. 뷰-포트에 투영되는 광선(ray).

참고) 컴퓨터 비전 분야에서 사용되는 이미지 평면(image plane) 용어는 컴퓨터 그래픽스 분야에서는 뷰-포트(viewport)라는 용어로 사용한다.

2.1.3. 렌더링(Rendering)

렌더링은 컴퓨터 그래픽스에서 3차원 객체를 주어진 시점의 2차원 이미지로 투영하는 과정이다. 실제 카메라의 투영 과정과 유사하고, 실제 3차원 물체와 실제 이미지 평면이 아닌 컴퓨터 프로그램을 통해 가상의 3차원 물체를 뷰-포트로 투영한다는 차이점이 있다. 컴퓨터 그래픽스에서는 이미 알고 있는 3차원 물체를 (현실과 같이) 실감나고 (제약된 컴퓨팅 자원과 시간 내에) 효율적으로 뷰-포트에 투영하는 것이 중요한 문제이다.

일반적으로 렌더링은 렌더링을 적용할 범위에 따라 체적 렌더링(volume rendering)과 표면 렌더링(surface rendering)으로 나뉜다. 체적 렌더링은 객체의 내부까지 렌더링하게 되고, 표면 렌더링은 객체의 표면만 렌더링한다. 일반적으로 표면 렌더링으로 충분하지만, 복잡한 물리적 현상(예: 안개, 화산폭발로 인한 재, 구름 등)을 실감나게 표현하기 위해서는 표면 렌더링(surface rendering)을 사용하는 경우 오히려 계산이 너무 복잡해지거나 그 결과가 비현실적인 경우가 많다. 따라서 이러한 경우 체적 렌더링(volume rendering)을 통해 빠르고 효과적인 결과를 얻을 수 있다.

그림 3. 체적 렌더링이 필요한 현상의 예: 화산 폭발(Pixabay License).

체적 렌더링은 광선이 주어진 물체의 체적(volume)을 지나며 모은 색상 광체(color radiance)를 합쳐서 뷰-포트에 렌더링한 것이다. 컴퓨터 그래픽스에서 [그림 3]의 화산 폭팔에 의한 구름과 같이 체적이 크면서 내부가 비어있거나 밀도가 낮은 오브젝트들을 렌더링 할 때 사용된다. NeRF는 체적 렌더링, 특히 광선행진(ray marching)을 이용한 방식을 사용한다.

2.1.4. 체적 광선 행진(Volume Ray Marching)

체적 렌더링에서 광선행진(ray marching)은 빛이 발광체로부터 나와 공간 내 체적을 통과하며 내부 입자와 다양한 형태의 상호작용을 하며 카메라에 입사하는 과정을 모사한다. 광선투사는 빛, 즉 광선을 [그림 8]과 같이 작은 조각(sample)으로 나누어 일어난 상호작용을 묘사한다. 광선투사는 [그림 4]와 같이 진행 방향에 따라 정방향 광선행진(forward ray marching)과 역방향 광선행진(backward ray marching)으로 구분된다. 정방향 광선행진은 빛이 카메라에서 부터 출발하여 광원까지 이동하는 방법이고, 역방향 광선행진은 반대로 광원에서 출발한 빛이 카메라로 입사하는 방법이다. 카메라의 투영된 이미지를 렌더링하는데는 카메라에 입사하지 않는 광선은 고려할 필요가 없다. 따라서 빠르고 효율적인 계산을 위해 정방향 광선행진을 사용하는 것이 일반적이다.

2.1.5. 투광도(Transmittance)와 체적밀도(Volume Density), 체적렌더링(Volume Rendering)

빛은 체적을 지나며 흡수, 산란 등과 같은 다양한 현상이 겪는다. 예를 들어 빛이 체적을 통과하는 길이(path length)가 길어질수록 많은 양이 흡수되어 그 세기가 약해진다. 마찬가지로 체적의 농도(concentration)가 높거나, 체적을 이루는 입자가 빛을 흡수하는 정도, 즉 흡수율(absorptivity)이 크면 빛이 체적을 통과하며 그 세기가 약해진다. Beer-Lambert law [12]는 앞서 언급한 체적을 통과하는 길이, 체적의 농도, 체적의 흡수율과 같은 세 가지 요소와 빛의 투광도(transmittance) T 사이의 관계를 설명한다.

NeRF는 아래와 같이 체적밀도(volume density) 개념을 이용한 간략화된 투광도 모델을 사용한다. NeRF에서 체적밀도 $\sigma(\mathbf{r}(s))$는 광선 $\mathbf{r}$이 광선 위의 한 지점 s에서 사라질 (미분가능한) 확률로 정의한다. 이를 쉽게 이해하기 위해서는 Beer-Lambert law에서 체적의 농도나 체적의 흡수율과 관련된 변수로 생각할 수 있다. 즉 체적밀도가 크다는 말은 체적의 농도가 높거나 흡수율이 높다는 직관을 가질 수 있다.

NeRF는 아래와 같이 조금 간략화된 Beer-Lambert Law를 사용한다. 주어진 t 지점의 투광도 $T(t)$는 카메라에서 가까운(near) 지점 t_n부터 광선 $\mathbf{r}$이 출발할 때, 체적의 농도와 흡수율을 고려한 각 지점의 체적밀도(volume density) σ를 미소(infinitesimal) 길이 대해 적분하여 얻을 수 있다.

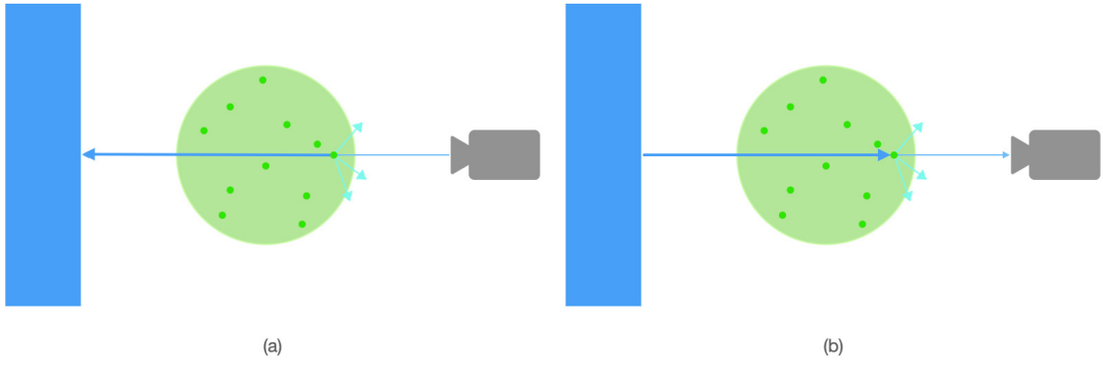

그림 4. 체적 광선 행진(volume ray marching) 의 종류: (a) forward ray marching, (b) backward ray marching.

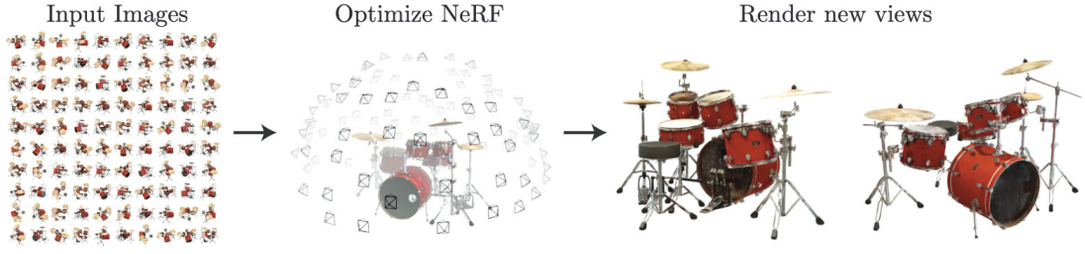

그림 5. NeRF 흐름도 [1].

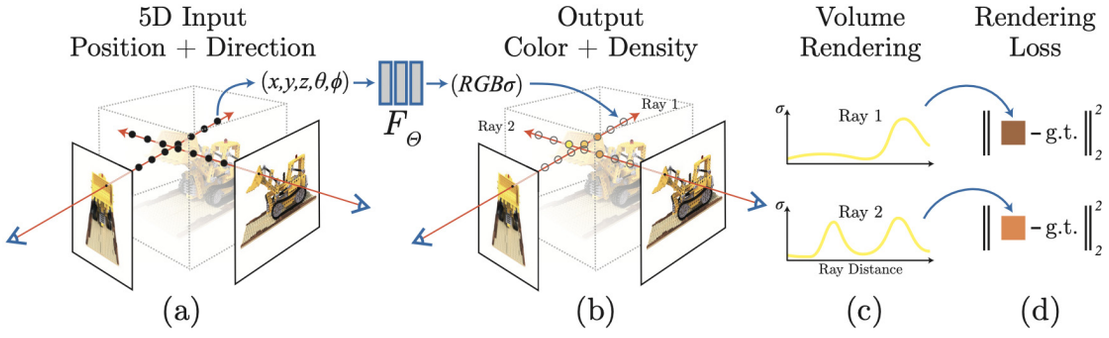

그림 6. NeRF의 입력(a)과 출력(b), 그리고 체적렌더링을 통해 각 광선의 샘플 지점마다 획득한 색상(c), 그리고 이를 이용한 제곱에러 손실함수(d) [1].

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

따라서 위의 투광도을 기반으로 광선 $\mathbf{r}$이 카메라에서 가까운(near) 지점 t_n부터 카메라로 t_f부터 먼(far) 지점 까지 이동하였을 때, 체적렌더링을 수행하면 광선에 의해 채적렌더링하여 얻은 색상 $C(\mathbf{r})$은 아래와 같이 투광도, 체적밀도, 색상의 곱을 광선이 통과하는 길이에 대해 적분한 것과 같다.

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

위의 체적 렌더링 수식의 적분은 구적법(quadrature rule)을 이용[2]하여 컴퓨터에서 이산적으로 연산이 가능하도록 아래와 같이 도출할 수 있다.

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

T_i를 해당 지점 직전인 i-1까지의 투광도, 1-exp(- $\sigma_i \delta_i$) 항을 i지점에서의 투광도의 역, 즉 불투명도(opacity)로 주어진 색 $\mathbf{c}_i$가 발현될 정도로 생각해도 된다. 또 위의 수식에서 $T_i(1-\exp(-\sigma_i\delta_i))$항을 α_i로 정의하면, 위의 수식은 $\hat{C}(\mathbf{r}) = \sum_{i=1}^{N}\alpha_i\,\mathbf{c}_i$와 같은 가중치 합, 즉 알파 합성(alpha compositing) [13]과 같이 생각할 수도 있다.

2.2. Neural Radiance Field (NeRF)

NeRF(Neural Radiance Field)의 3차원 물체나 공간의 체적렌더링에 필요한 체적밀도 σ와 색 $\mathbf{c}$를 출력하는 심층신경망이다. NeRF의 심층신경망은 광선 위의 3차원 점(x,y,z)과 해당 위치에서의 광선의 방향(θ, ϕ)을 입력으로 하는데, 학습된 사용된 3차원 점과 방향 뿐만 아니라 그 사이의 임의의 위치와 방향이 주어져도 학습된 결과를 고려한 연속적(continuous)이고 의미있는 값을 도출한다. 따라서 [그림 9]와 같이 학습에 사용되지 않은 새로운 시점에 대한 고품질의 영상을 생성(novel view synthesis)할 수 있다. 예를 들어 주어진 두 시점 사이의 고품질

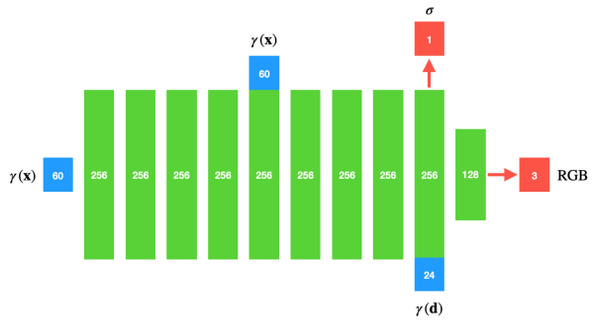

그림 7. NeRF의 심층신경망 모델: 3차원 위치 와 광원의 방향 입력과 9개의 FC 레이어로 구성.

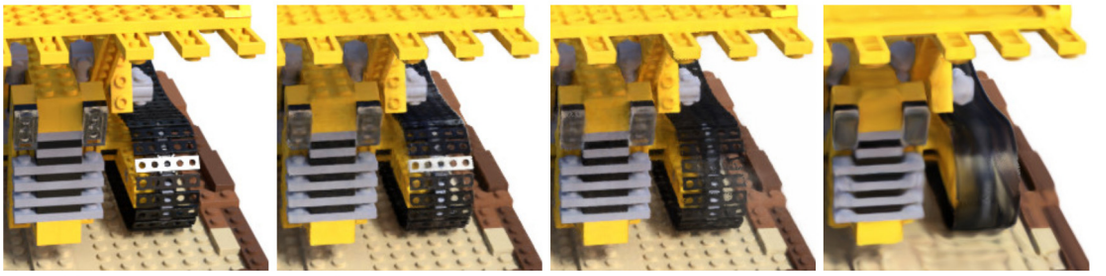

그림 8. 실제 이미지와 NeRF의 최종 결과, NeRF에서 시점을 무시한 결과, NeRF에서 위치인코딩을 적용하지 않은 결과 [1].

그림 9. NeRF와 FastNeRF의 성능 비교 [5]: (좌) NeRF vs. (우) FastNeRF.

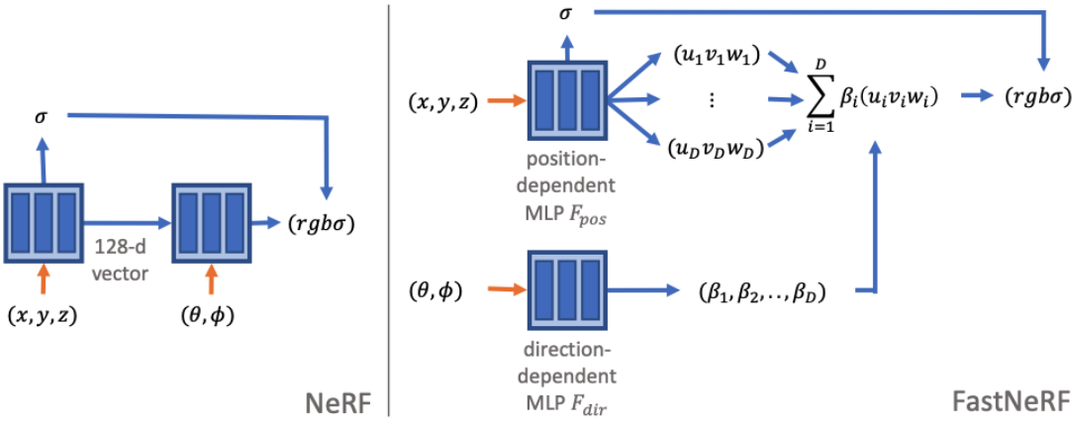

그림 10. NeRF와 FastNeRF의 모델 비교 [5]: (좌) NeRF vs. (우) FastNeRF.

영상들을 생성하여 부드럽게 움직이는 비디오를 만들 수도 있다. 주어진 영상의 시점(viewpoint)는 학습시 주어져야 하기 때문에 COLMAP [3]과 같은 structure-from-motion 기술을 통해 사전에 학습 영상 사이의 시점 관계를 획득하여야 한다. NeRF는 폴리곤메시(polygon mesh)나 점군(point cloud)와 달리 3차원 물체나 공간의 형상을 직접적이고 명시적으로 표현하지 않는다. 대신 NeRF는 블랙박스와 같은 신경망을 이용하여 radiance field를 통해 간접적으로 표현하기 때문에 NeRF를 암시적(implicit) 표현법의 하나로 생각할 수 있다.

NeRF를 이용해 주어진 시점의 영상을 얻는 것은 체적렌더링을 통해 진행된다. [그림 10]의 (a)와 같이 원하신 시점 영상의 각 픽셀의 광선(ray) $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ 를 생성하고, 광선을 주어진 구간 (t_n,t_f)에 대해 다음과 같이 균일하게 샘플링하여 3차원 점의 지점을 선택한다.

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

샘플링된 점 (x_i, y_i, z_i)과 광선의 방향을 NeRF에 입력하여 체적밀도 σ_i와 색상 $\mathbf{c}_i$를 얻는다. 얻어진 값을 체적렌더링 수식을 통해 가중치 합을 구해 해당 픽셀의 색상값 $\hat{C}(\mathbf{r})$을 얻을 수 있다.

NeRF의 학습은 체적렌더링과 제곱에러(squared error) 손실함수 통해 진행된다. 주어진 데이터셋의 각 영상의 각 광선에

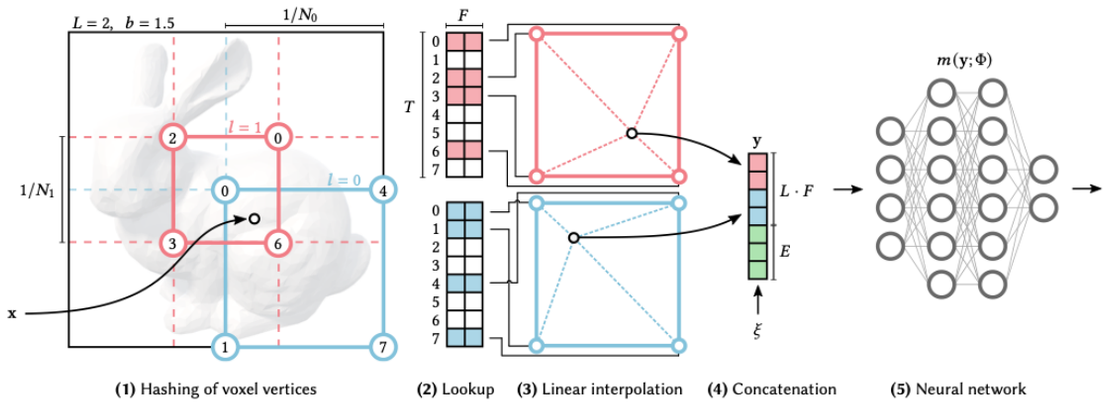

그림 11. Instant Neural Graphics Primitives 모델 [6].

대해 위의 광선 샘플링과 체적렌더링을 통해 해당 픽셀의 색상값 $\hat{C}(\mathbf{r})$를 얻을 수 있다. 하나의 광선 샘플링된 점들을 하나의 배치(batch) 단위로 학습이 이루어지는데, 해당 배치(batch)에서 얻은 픽셀의 색상값 $\hat{C}(\mathbf{r})$와 영상 데이터셋에 있는 해당 위치의 참값 $C(\mathbf{r})$에 대해 $\|\hat{C}(\mathbf{r}) - C(\mathbf{r})\|_2^2$ 같은 제곱에러 손실함수를 최소화하는 최적화를 수행한다.

2.2.1. NeRF의 심층신경망 모델

NeRF의 심층신경망 모델은 [그림 11]과 같이 ReLU 함수를 사용하는 9개의 FC 레이어로 구성되어 있다. 신경망의 입력값으로 3차원 위치 **x**와 해당 위치에서의 광원의 방향 **d**를 사용한다. 이때 이를 1번째 레이어에 한 번에 입력하는 것이 아니라, 1번째 레이어에서는 3차원 위치 **x**만 넣고, 5번째 레이어의 결과에 다시 3차원 위치 **x**를 붙여서 다음 레이어의 입력으로 사용한다. 8번째 레이어의 결과로 체적밀도 σ를 얻을 수 있고, 9번째 레이어에 광원의 방향 **d**를 넣어 RGB 색상값 **c**를 얻는다. 3차원 위치 **x**와 광원의 방향 **d**를 분리하여 넣는 이유는 체적밀도의 경우 광원의 방향에 관계없이 같은 값을 가져야하고, RGB 색상은 광원의 방향에 따라 달라져야 하기 때문이다. 예를 들어 도자기와 같은 물체는 반사 때문에 보는 시점에 따라서 같은 위치의 색상이 달라져 보이는데, NeRF의 신경망 모델은 이러한 램버시안(non-Lambertian) 반사를 효과적으로 표현할 수 있다. NeRF의 저자는 이를 multi-view consistent 라고 하였다.

NeRF는 고품질의 결과를 얻기 위해 두 가지 추가적인 과정을 거치는 것이 중요한 특징이다. 입력 데이터를 그대로 신경망의 입력에 넣지 않고, 위치인코딩(positional encoding) γ로 변환하여 넣는다. 또 제한된 수의 광선 위의 샘플을 통해 효과적인 결과를 얻기 위해 계층적 체적샘플링(hierachical volume sampling)을 수행한다.

2.2.2. 위치인코딩(Positional Encoding)

신경망에 입력되는 작은 값의 차이는 유사한 값을 출력한다. 때문에 NeRF를 이용해 렌더링한 영상은 [그림 12]의 오른쪽과 같이 디테일이 뭉게지기 쉽다.

NeRF는 이러한 문제를 입력값의 위치인코딩(positional encoding)으로 해결한다. NeRF의 위치인코딩은 Transformer을

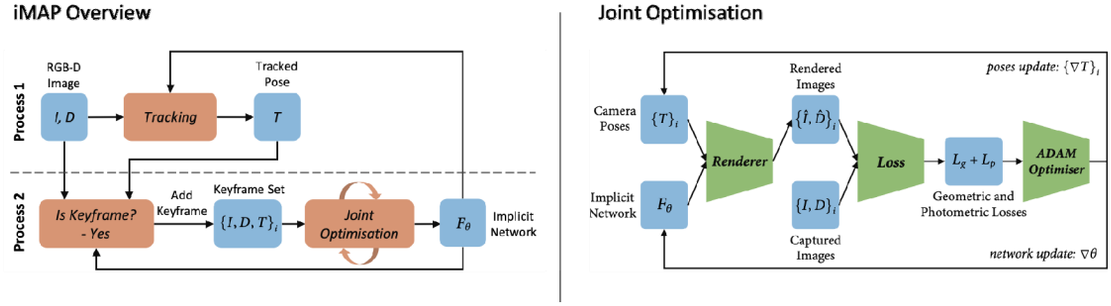

그림 12. iMAP 시스템 파이프라인과 카메라 포즈 및 네트워크 최적화 흐름도 [7].

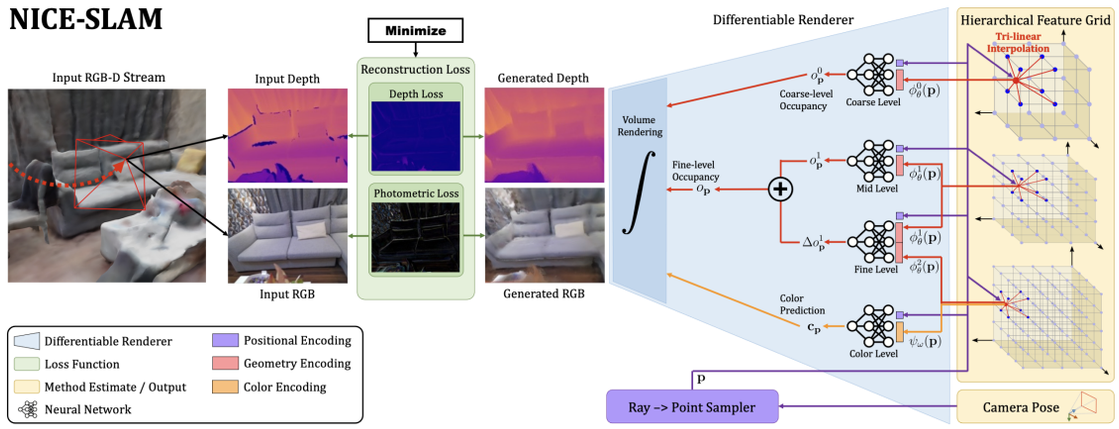

그림 13. NICE-SLAM의 흐름도 [8].

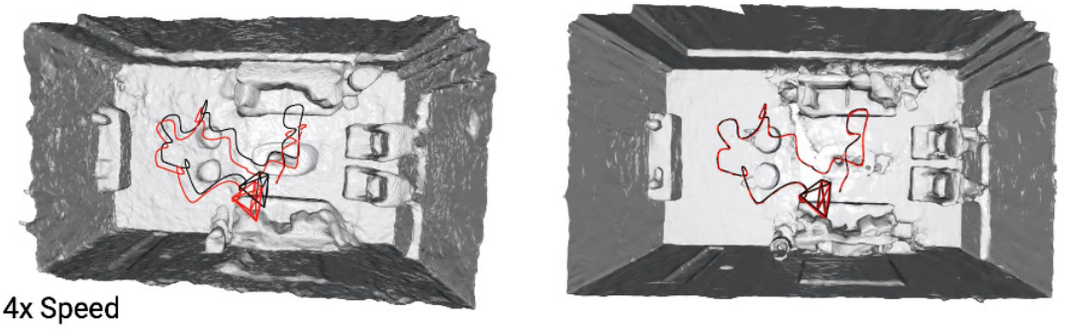

그림 14. iMAP과 NICE-SLAM 결과 비교 [8]: (좌) iMAP vs. (우) NICE-SLAM.

이용한 단어 인코딩과 조금 닮았지만 다른 목적과 다른 형태의 인코딩 방법이다. NeRF의 위치인코딩을 주어진 1차원의 값 p를 아래와 같이 $2L$차원으로 확장한다. (NeRF에서는 L=10 사용)

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

위의 위치인코딩의 주어진 값 p에 $2^d\,\pi$ 와 같은 형태의 수를 곱하는데, 인코딩 결과 벡터 앞 부분은 π 와 같이 작은 수를 곱하고 인코딩 결과 벡터의 뒷 부분은 $2^9\,\pi$ (L=10)와 같이 큰 수를 곱한다. 이렇게 큰 수를 곱하는 것은 주어진 값 p의 작은 값의 부분을 증폭시키는 역할을 한다. 증폭된 결과에 진폭이 제한된 주기 함수인 삼각함수가 적용되기 때문에 증폭된 결과의 $2\,\pi$ 보다 큰 값은 결과에 영향을 주지 않아 주어진 p값 의 작은 값에만 영향을 받는 인코딩이 된다. NeRF는 이와 같이 위치인코딩을 통해 주어진 값의 큰 값 영역부터 작은 값 영역까지 고르게 신경망에 입력하여 신경망이 세밀한 위치 변화나 광원의 방향 변화에도 민감하게 결과를 출력하도록 만든다. 이를 통해 NeRF는 [그림 12]의 왼쪽과 같은 세밀하고 고품질의 렌더링이 가능하다.

2.2.2. 계층적 체적샘플링(Hierarchical Volume Sampling)

주어진 광선을 주어진 넓은 범위(t_n, t_f)에 정해진 N개의 점으로 균일하게 샘플링하는 경우 물체의 세밀하거나 작은 부분을 놓치기 쉽니다.

NeRF는 균일한 샘플링 이후, 물체가 존재하는 부분에 더욱 촘촘하게 샘플링하는 두 단계의 샘플링 기법을 사용한다. 앞서 NeRF에서 사용되는 체적렌더링은 아래와 같은 가중치(weight) 합으로 표현할 수 있다.

$$\hat{C}_c(\mathbf{r}) = \sum_{i=1}^{N_c} w_i\, c_i \quad \text{where} \quad w_i = T_i\left(1 - \exp(-\sigma_i\delta_i)\right)$$

가중치 W_i는 해당 색상이 광선의 투영될 색상에 대한 영향치로서, 큰 가중치 값을 갖는 해당 지점 근처는 보다 촘촘하게 샘플링하는 것이 아이디어이다. 즉 NeRF에서는 균일한 성긴(coarse) 샘플링과 가중치에 따른 촘촘한(fine)한 샘플링

을 모두 이용한다. 가중치에 따른 촘촘한 샘플링을 위해 주어진 가중치를 $\hat{w}_i = w_i / \sum_{j=1}^{N_c} w_j$와 같이 정규화하여 모든 가중치의 합이 1이 되도록 한다. 그리고 0과 1사이의 난수를 생성하여 룰렛-휠 방식의 stratified sampling을 이용하면 큰 가중치를 갖는 곳을 더 빈번하게 샘플링할 수 있다.

NeRF는 광선의 3차원 점 선택에 두 단계의 샘플링 적용하고, NeRF 학습을 위한 손실함수에서도 아래와 같이 두 가지 샘플링에 따른 두 가지 렌더링 결과 $\hat{C}_c(\mathbf{r})$와 $\hat{C}_f(\mathbf{r})$를 각각 손실함수에 넣어 신경망을 학습하는데 이용한다.

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

2.2.3. NeRF의 장단점

NeRF는 체적렌더링의 미분가능한 특성을 고려하여 학습에서 주어지지 않은 3차원 위치와 광선 방향에 대해서도 연속적(continous)으로 의미있는 결과를 준다. 즉 NeRF는 multi-view stereo (MVS)와 같이 점군에 의존한 전통적인 물체 표현 방식이 관찰값에만 의존하는 부분을 극복할 수 있고, 전통적인 방법에 비해 고품질의 새로운 시점을 렌더링하는 것이 가능하다. 또한 위치인코딩이나 계층적 체적샘플링을 통해 기존의 다른 신경망을 이용한 표현보다 세밀한 렌더링이 가능하다. 또 상대적으로 단순하고 작은 신경망 모델을 사용하기 때문에 고품질의 물체 정보를 15MB 정도의 매우 작은 크기로 저장할 수 있는 장점도 있다.

그러나 NeRF는 하나의 물체를 학습하고 렌더링하는데 긴 시간이 소요된다. V100 급의 GPU에서도 물체 학습에 약 1-2일 정도의 시간이 소요된다고 한다. 또한 하나의 영상을 생성하는 렌더링 시간도 꽤 많은 시간이 필요하기 때문에 SLAM이나 VR/AR과 같이 실시간 동작이 필요한 응용에 사용하기는 아직 부족하다.

3. NeRF 관련 최근 기술 동향

최근 NeRF의 단점을 보완하거나 다양한 분야에 이를 활용하는 많은 의미있는 연구들이 진행되고 있다. 본 기고에서는 세 가지 연구 방향에서 대표적인 연구들을 개략적으로 살펴보고자 한다. 우선 1) NeRF 모델을 개선하는 것에 대한 연구를 살펴보고, 2) visual SLAM 분야에 NeRF를 적용하는 연구 그리고 3) 자동 데이터셋 라벨링의 위한 auto labeling 분야에 NeRF를 적용하는 연구를 차례로 살펴본다.

3.1. Advanced NeRF

기존 NeRF을 개선한 모델 중 대표적인 것 중 하나는 FastNeRF [5]이다. FastNeRF는 NeRF의 단점인 큰 공간복잡도(space complexity)에 따른 긴 렌더링 시간 문제를 극복하자 하였다. FastNeRF는 기존 NeRF의 신경망을 분리하고 캐시 메모리의 사용을 극대화하여 NeRF의 단점을 극복하고자 하였다. FastNeRF는 NeRF의 두 가지 입력인 position과 direction을 단일 네트워크가 아닌 두 개의 네트워크로 분리하였다. 두 입력 데이터인 position과 direction이 각각 k와 l 크기의 값으로 인코딩되어 사용된다고 할 때, 기존 NeRF의 공간복잡도는 $O(k^3 l^2)$인데, FastNeRF에서는 $O(k^3 \times (1+3D) + l^2 \times D)$와 같이 줄어든다.

NVIDIA에서 제안된 Instant NeRF [6]은 NVIDIA의 CUDA와 기존 NeRF의 positional encoding 부분을 해시맵(hash map)과 선형보간법으로 바꾸어서 계산의 효율성을 크게 향상했다. NVIDIA의 CUDA ToolKit [10] 은 복잡한 연산을 GPU 가속화을 이용해 수행하는 기능을 제공하고, Tiny CUDA Neural Networks [11] 는 다른 라이브러리 보다 더 작은 효율적인 뉴럴 네트워크를 지원한다. Instant NeRF가 포함된 NVIDIA의 Instant Neural Graphics Primitive는 NeRF 뿐만 아니라 gigapixel image, signed distance function, neural radiance caching에도 적용되어 매우 인상적인 결과를 보여준다. 해당 기술은 SIGGRAPH 2022의 최우수논문으로 수상하였고, GUI를 제공하여 일반 사용자 쉽고 편리하게 사용할 수 있도록 되어있다.

3.2. Visual SLAM

SLAM(Simultaneous Localization and Mapping)은 위치인식 동시에 주변의 지도를 작성하는 기술로 로봇이나 자율차, 드론의 자율주행 뿐만 아니라 AR/VR에도 사용된다. SLAM에서 중요한 요소 중 하나는 공간의 지도를 어떻게 표현하는가이다. 전통적인 SLAM 기법은 지도를 랜드마크나 영상 특징점(feature point)의 집합, 즉 sparse하게 표현하였다. 최근에는 지도를 보다 dense하게 표현하고, dense한 지도를 그대로(direct) 활용하여 위치인식을 수행하는 연구가 많은 주목을 받았다. 이러한 새로운 접근 방법은 전통적인 접근 방법에서 필요한 특징점 추출과 정합 과정이 없기 때문에, 이러한 단계에서 발생하는 정확도와 계산시간에 대한 한계(bottleneck)로 부터 자유롭다. NeRF를 이용한 SLAM 기술은 이러한 최근의 접근 방법의 연장선으로 볼 수 있다. 즉, 기존의 dense한 표현 방식인 복셀(voxel)이나 폴리곤메시(polygon

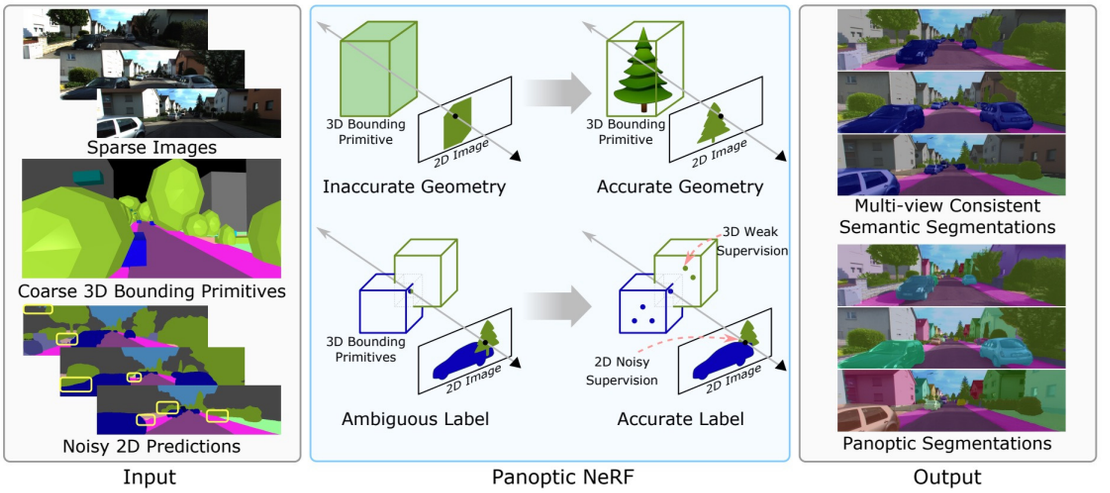

그림 15. Panoptic NeRF의 흐름도 [9].

mesh), 점군(point cloud)을 대신하여 NeRF를 사용하는 것이다. 따라서 NeRF를 빠르고 효율적으로 학습시키고 위치인식을 위해 이를 효과적으로 사용하는지가 중요한 요소이다.

iMAP [7]은 RGB-D 카메라를 이용하여 NeRF를 SLAM 문제 효율적으로 적용한 기술이다. RGB 영상뿐만 아니라 depth 데이터를 이용하여 3차원 점의 위치를 알 수 있고, 이를 통해 학습하고 렌더링하는 과정이 빠르고 쉽게 진행된다. iMAP에서는 기존의 NeRF에서 광선행진을 하는 부분이 생략되고, 시점 정보에 따라 달라지는 빛 반사, 즉 SLAM 지도에 불필요한 광선(ray)의 방향을 고려하지 않는다. 또 모델의 크기를 4개의 hidden layer로 바꾸어 신경망의 크기를 줄였다. iMAP은 RGB 데이터를 고려한 광학적에러(photometric loss)와 depth 데이터를 고려한 기하학적에러(geometric loss)를 학습을 위한 손실함수로 사용하였다. iMAP은 10 Hz의 위치추정과 2 Hz의 지도작성이 가능하다. 기존의 visual SLAM이나 RGB-D SLAM에 비해 크게 느린 속도를 갖지만, 최초로 신경망 지도를 이용하여 온라인으로 동작하는 SLAM 기술이라는 의의가 있다.

NICE SLAM [8]은 iMAP을 좀 더 개선한 기술이다. iMAP의 단점은 지도의 퀄리티가 떨어지고, 작은 크기의 지도만 학습 가능하며 학습 시간이 길다. 이를 해결하기 위해 NICE SLAM은 하나의 커다란 신경망이 아닌 4개의 작은 신경망으로 나누어 구조적 scene과 카메라 포즈를 학습한다. 이렇게 학습된 모델은 [그림 18]에서 볼 수 있듯이 오른쪽에서 왼쪽 방향의 Differentiable Renderer를 통해 depth 이미지와 color 이미지를 렌더링 할 수 있다. 해당 구조를 통해 지도의 퀄리티를 개선하고 실외의 대형 공간에서도 SLAM이 가능하며 카메라 트래킹 속도 또한 iMAP 보다 빠른 결과를 보인다. NICE-SLAM은 약 20 Hz의 위치추정과 8 Hz의 지도 생성이 가능하며 보이지 않는 곳 또한 채워주는 기능을 보인다.

3.3. Auto Labeling

데이터를 수집하고 라벨링을 하는 과정은 컴퓨터비전을 비롯한 딥러닝, 기계학습 업무에서 가장 많은 시간과 노력이 필요한 작업이다. 따라서 이러한 시간과 노력을 줄이기 위해 다양한 연구들이 진행되어 왔다. 데이터를 적게 필요로 하게 하는 weakly-supervised learning 기술, 실제 현실세계에서 수집한 데이터가 아닌 가상세계에서 수집한 데이터를 활용하는 Sim2Real 기술, 그리고 자동으로 라벨링을 수행하는 auto labeling 기술이 그 예이다.

Panoptic segmentation은 물체/공간 종류(class)에 대한 semantic segmentation과 개별 물체에 대한 instance segmentation을 합친 가장 복잡한 레벨의 segmentation 기술이다. 따라서 panoptic segmentation의 학습 영상을 만들기 위해서는 매우 고난이도의 복잡한 라벨링이 필요하다. 따라서 숙련된 사람도 panoptic segmentation을 위해 1장의 영상에 약 1.5시간을 소요[9]한다고 한다.

Panoptic NeRF [9]는 panoptic segmentation을 위한 라벨링을 간단한 3차원 도형으로 라벨링한 초기 결과와 2차원 바운딩박스를 이용하여 수행한다. Panoptic NeRF는 NeRF가 생성해내는 3차원 정보와 2차원 렌더링 사이의 관계를 이용하여 최종적으로 매우 정교하게 라벨링된 결과를 제공한다. Panoptic NeRF를 통해 영상 1장 약 0.75분에 panoptic segmentation을 완료할 수 있다고 한다.

4. 맺음말

본 기고에서는 최근 많은 관심을 받고 있는 NeRF에 대한 소개와 이를 이용한 세 가지 분야의 최근의 인상싶은 연구들을 소개하였다. NeRF에 대한 이해를 돕기 위해 컴퓨터그래픽스에서 사용하는 체적 렌더링(volume rendering)이나 광선행진(ray marching)과 같은 용어를 리뷰하였고, 이를 바탕으로 NeRF의 원리를 소개하였다. NeRF는 신경망을 이용해 3차원 공간과 물체를 표현하는 블랙박스(implicit) 표현 방법으로 기존의 표현 방법보다 훨씬 정교하고 새로운 시점에 대해서 깔끔한 렌더링 결과를 만들어준다. 그러나 상대적으로 긴 모델의 학습 시간과 렌더링 시간은 NeRF의 큰 문제점 중 하나이고, FastNeRF를 비롯해 RGB-D SLAM 분야의 iMAP이나 NICE-SLAM은 이러한 단점을 극복하고자 하였다. 본 기고에서 미처 소개하지 못한 멋진 연구들이 지금 시간에도 많이 발표되고 있고, 본 기고가 독자들에게 NeRF에 대한 관심을 환기하는 계기가 되길 바란다.

감사의 글

본 기고는 과학기술정보통신부 및 한국연구재단의 'BRIDGE융합연구개발'사업의 지원으로 작성되었습니다. (과제명: AI기반 3차원 곡면에서의 위치 인식 및 이동 경로 생성 기술, 과제번호: 2021M3C1C3096810)

REFERENCES

[1] Ben Mildenhall et. al., "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis", ECCV, 2020, https://www.matthewtancik.com/nerf

[2] Nelson Max et. al., "Optical models for direct volume rendering." IEEE Transactions on Visualization and Computer Graphics, vol. 1, no. 2, 1995 https://doi.org/10.1109/2945.468400

[3] Johannes L. Schonberger, et. al., "Structure-from-Motion Revisited", CVPR, 2016, https://colmap.github.io/

[4] Scratchapixel, https://www.scratchapixel.com/

[5] Stephan J. Garbin, et. al., "FastNeRF: High-Fidelity Neural Rendering at 200FPS", CVPR, 2021, https://microsoft.github.io/FastNeRF/

[6] Thomas Muller, "Instant Neural Graphics Primitives with Multiresolution Hash Encoding", SIGGRAPH, 2022, https://nvlabs.github.io/instant-ngp/

[7] Edgar Sucar et. al., "iMAP: Implicit Mapping and Positioning in Real-Time", CVPR, 2021, https://edgarsucar.github.io/iMAP/

[8] Zihan Zhu et. al., "NICE-SLAM: Neural Implicit Scalable Encoding for SLAM", CVPR, 2022, https://pengsongyou.github.io/nice-slam

[9] Xiao Fu, et. al., "Panoptic NeRF: 3D-to-2D Label Transfer for Panoptic Urban Scene Segmentation", CVPR, 2022, https://fuxiao0719.github.io/projects/panopticnerf/

[10] NVIDIA CUDA Toolkit, https://developer.nvidia.com/cuda-toolkit

[11] NVIDIA Tiny CUDA Neural Networks, https://github.com/NVlabs/tiny-cuda-nn

[12] Beer-Lambert Law, Wikipedia, https://en.wikipedia.org/wiki/Beer–Lambert_law

[13] Alpha compositing, Wikipedia, https://en.wikipedia.org/wiki/Alpha_compositing

저자약력

최 준 혁

· 2015년~2022년 단국대학교 기계공학과 (공학사)
· 2022년~현 재 서울과학기술대학교 컴퓨터공학과 (석사과정)

최 성 록

· 2001년~2006년 서울대학교 기계항공공학과 (공학사)
· 2006년~2008년 KAIST 로봇공학학제전공 (공학석사)
· 2014년~2019년 KAIST 로봇공학학제전공 (공학박사)
· 2008년~2020년 ETRI 지능로보틱스연구본부 (선임연구원)
· 2021년~현 재 서울과학기술대학교 컴퓨터공학과 (조교수)