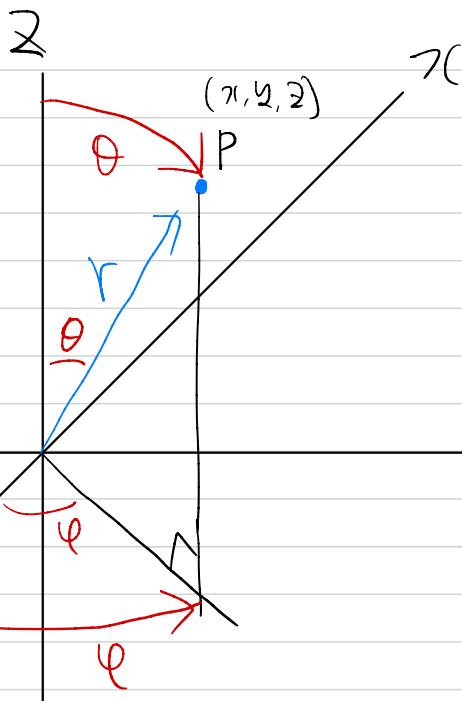


NeRF



• location 정보 (x, y, z)

- 투명도 density에 영향을 줌
객체를 바라보는 방향에 관계없이
객체의 한 점이 가지는 투명도는 고유하거
때문에 투명도는 객체에서의 location
정보에 영향을 받음

y

• viewing direction 정보 (θ, ϕ)

- rgb에 영향을 줌
객체의 한 점을 바라보는 방향에
따라서 보이는 색상이 달라짐
ex). 빛의 반사에 따른 영향 등

density : 투명도의 역수

[density 낮다 = 투명도가 높다 = 뒤에 있는 점들이 잘 보임]

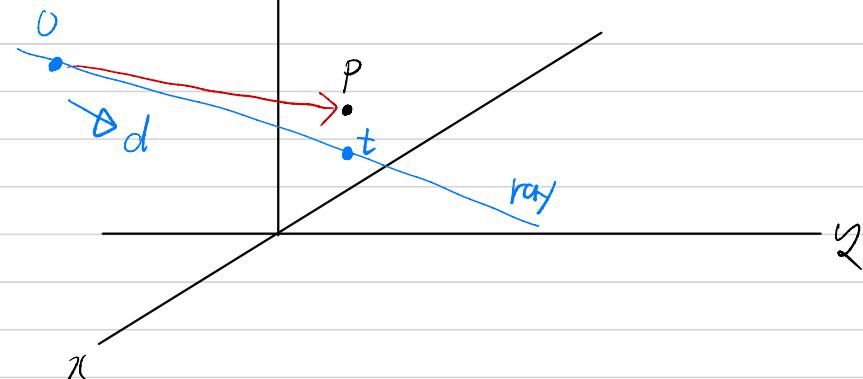
[density 높다 = 투명도가 낮다 = 뒤에 있는 점들이 잘 안보임]

∴ ray의 한 지점 t에서의 density가 클수록 투명도가 낮고, 다른 지점에 비해 잘 보이기 때문에
t 지점에서의 색상 값에 영향되는 가중치가 커진다.

camera 위치 $O (x, y, z)$ viewing direction $d (\theta, \phi)$

$$r(t) = O + \underbrace{td}_{\rightarrow 3D 좌표 (x, y, z)}, t는 해당 ray 위에 있음$$

\hookrightarrow ray 위에 위치한 챡플링된 점 t



〈NeRF에서 Deep Neural Network의 이용〉

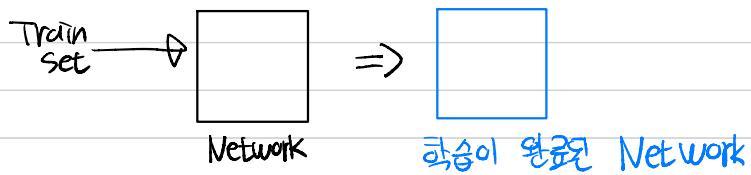
NeRF

$$\xrightarrow{\quad} F : (X, d) \rightarrow (C, \sigma)$$

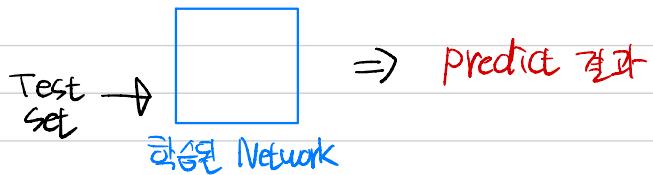
X, d 를 입력으로 받아 C, σ 를 출력으로 내주는 함수

= NeRF에서는 해당 함수를 Deep Neural Network로 구성(정의) 함

〈일반적인 Deep Neural Network〉



1. 초기 네트워크에 Train 데이터가 들어가면 학습이 완료된 네트워크를 얻는다.



2. 학습된 네트워크에 Test 데이터 즉, 실제 데이터가 들어오면 예측된 결과를 얻는다.
= 학습된 네트워크를 계속 재사용

<NeRF의 Deep Neural Network>

다른 입력데이터가 들어왔을 때
이미 학습을 시켰던 네트워크를 재사용하기 않는다는 의미

1. Train 데이터를 이용한 별도의 학습 과정이 없다.

= 별도의 학습된 네트워크를 얻지 않고, 입력 데이터가 들어가서 연산이 끝난 네트워크를 재사용하기 않는다.

주제 결과를 억우

2. 실제 렌더링 작업을 수행하기 위해서 2D 이미지 한 개를 넣어주면 전해진 epoch 만큼 반복하면서 네트워크 학습 한다,
epoch 만큼 학습 후 얻은 결과값 (c, S)으로 입력 받았던 한 Scene에 대해서 렌더링 한다.

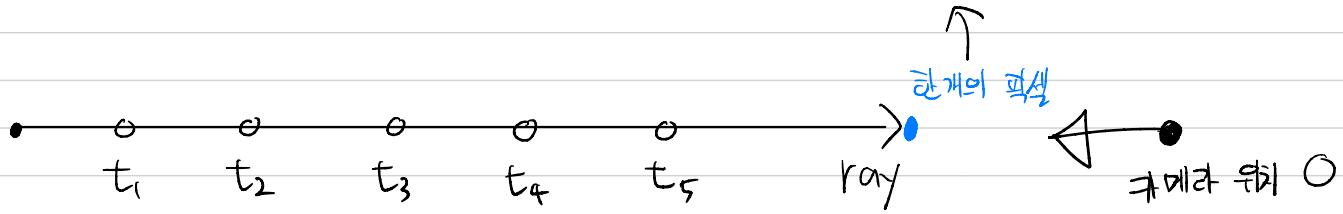
새로운 강연의 2D 이미지를 입력 받으면 이전 이미지로 학습된 네트워크에서 학습을 하기 않고, 새로운 네트워크를 한개 더 생성하여 해당 네트워크에서 학습한 후 입력 받은 Scene에 대해서 렌더링 한다.

∴ 한 장의 2D 이미지를 입력으로 넣어서 한 개의 scene을 렌더링 하는 과정이
곧 네트워크를 학습 시키는 과정이다.

Scene을 렌더링 하는 과정에서 필요한 rgb 값과 $density$ 의 정확도를 높이기 위해서 Deep Neural Network를 이용하는 것으로 판단된다.

<Volume Rendering>

○에서 ray를 바라 봄을 때 렌더링 되는 한개의 픽셀



픽셀은 ray 위에 있는 모든 $r(t)$ 에 대해서 각각의 density 값에 따른 색상의 가중치 합
 = 각각의 $r(t)$ 들은 density가 크거나 작음에 따라 색상을 표현하는 가중치가 다르다.
 따라서, 모든 $r(t)$ 는 표현되는 정도가 달라지기 때문에 이를 색상의 가중치 합은 ray의 한 픽셀로 볼 수 있다.

$$\Rightarrow C(r) = \frac{\int_{t_n}^{t_f} T(t) \times \sigma(r(t)) \times c(r(t), d) dt}{\text{density}} \quad \text{RGB} = \text{color}$$

$C(r)$: 이미지 픽셀 값

t_n : ray가 오브젝트를 향했을 때의 시작점

t_f : ray가 오브젝트를 향했을 때의 끝점

density를 RGB에 대한 확률로 생각
 density가 원인, RGB가 결과?
 $\begin{cases} \text{RGB} = \text{확률 변수} \\ \text{density} = \text{확률} \end{cases} \Rightarrow$ 해당 ray가 표현하는 픽셀을
 RGB와 density의 기댓값으로
 구할 수 있다.

확률 변수 : 확률 현상에 기인하여 어떤 결과 값이 확률적으로 정해지는 변수

확률 density의 확률 현상은 density가 가질 수 있는 값의 범위 중에서 어떤 값이 나올지 모르는 현상

\Rightarrow 따라서, 확률 density가 클 수록 확률 변수 RGB가 더 많이 반영된다.

$T(t)$: Transmittance 두과도
 ray가 구간 $[t_n, t]$ 사이에서
 객체와 만나지 않을 확률이라고 볼 수 있다.

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

\Rightarrow ray의 적분 구간 중에서 ray의 시작점인 t_n 부터 t_f 까지의 density 누적 값을
 음수로 exponential로 측정한 것

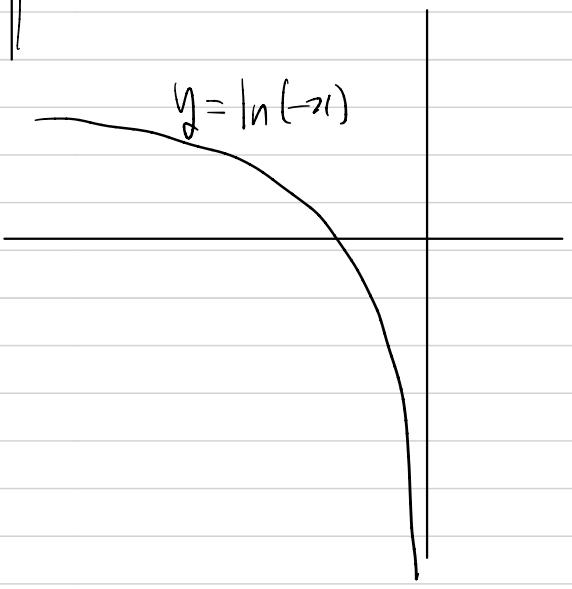
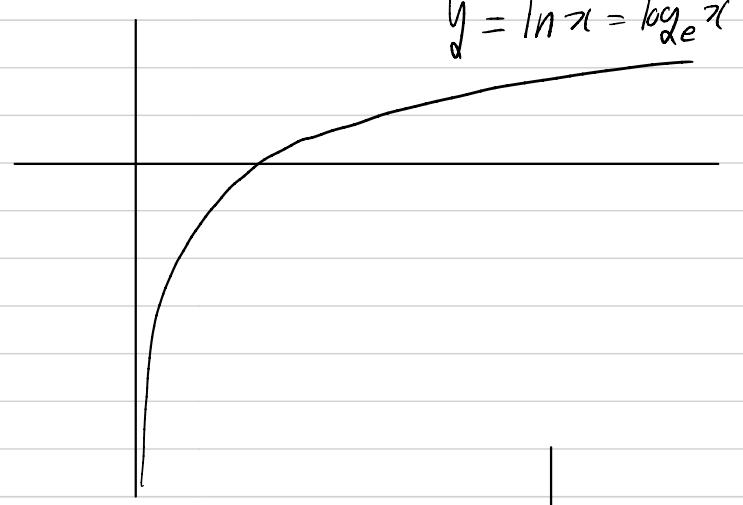
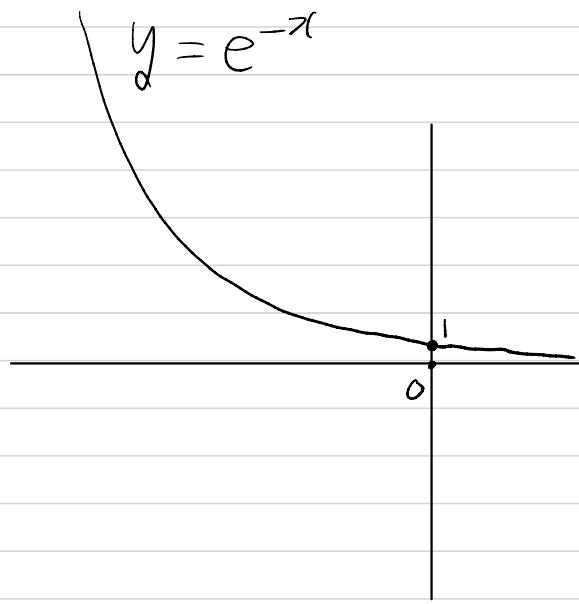
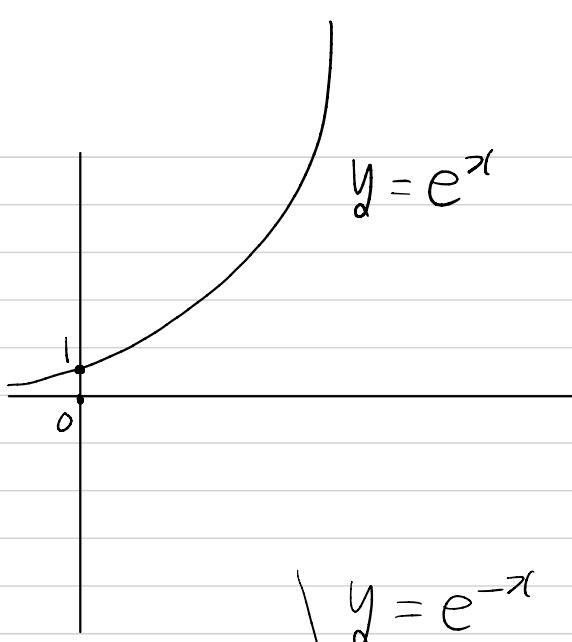
$$y = \exp(-x) = e^{-x}, \text{ density 누적값} = \alpha(t) \text{ 일 때 } T(t) = \exp(-\alpha(t)) = e^{-\alpha(t)}$$

$\therefore T(t)$ 값이 클수록 ray에서 t 뒤에 있는 영역이 잘 보이지 않는다.

지수 함수 (exponential function) : 로그함수의 역함수

지수 함수 $y = \exp(x)$: 자연로그의 역함수

자연로그 : 밑이 e인 로그, $e^x = y$ 일 때 자연로그는 $\ln y = x$ 이다.



< exponential에서 음수를 취하는 이유 >

$$\alpha = \int_{t_n}^t \sigma(r(s)) ds$$

$T(t) = |\alpha|/\alpha$ 가 객관화 불가능한 확률

$$T(t) = \exp(-\alpha) = e^{-\alpha}$$

$T(t)$

density 낮을 수록 투명도 높다.

density 누적값 < 0 이면 투명도 높다.

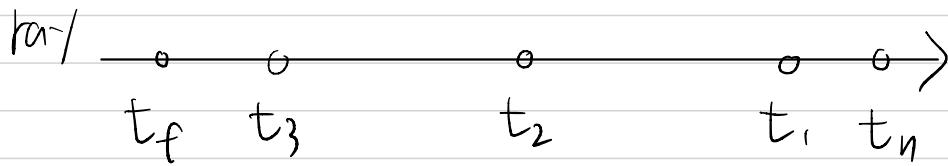
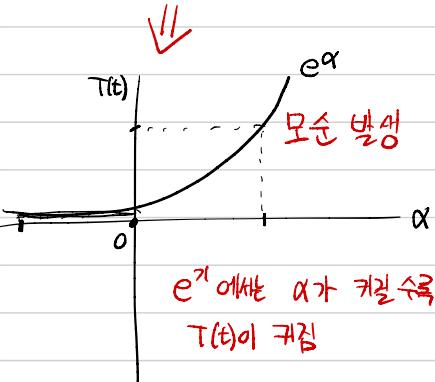
$\alpha > 0$ 는 $[t_n, t]$ 까지의 density들이 크다는 의미

(density 를 수록 투명도 높다. \rightarrow 투과율 낮다.)
 $= T(t)$ 가 크다.

$(\alpha (= density 합) 적을 수록 투명도 높다.)$
 \rightarrow 투과율 높다. $= T(t)$ 가 크다.)



$\alpha > 0$ 이면 $T(t)$ 이 작아지고,
 $\alpha < 0$ 이면 $T(t)$ 이 커짐이 성립한다.



$$\int_{t_n}^{t_2} \sigma(r(s)) ds = t_n \text{ density} + t_1 \text{ density} + t_2 \text{ density}$$

< $C(r)$ 을 실제로 계산하는 방법 >

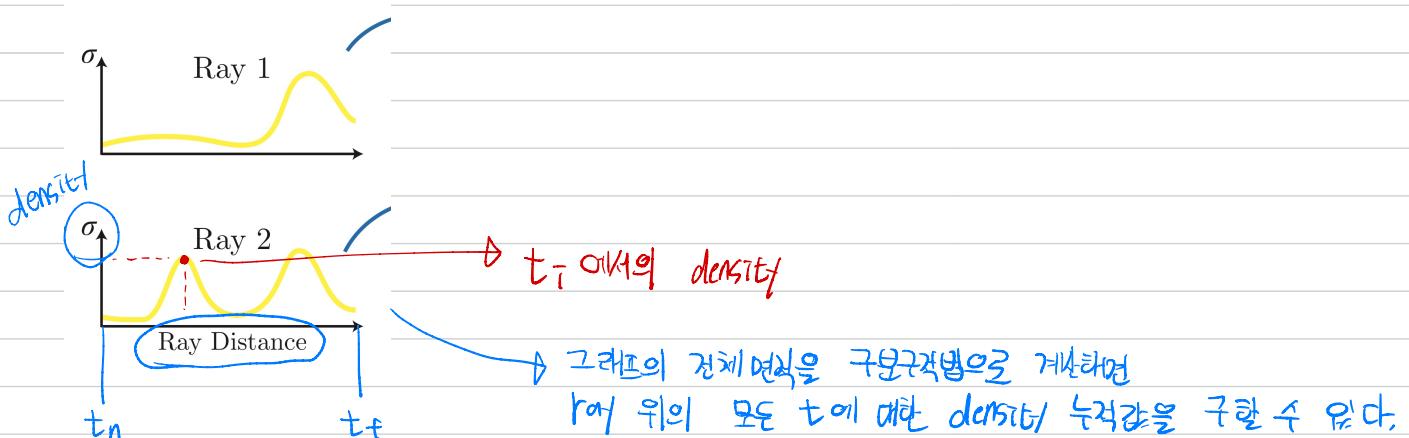
- 구분구적법 Deterministic Quadrature

일반적인 구분구적법으로 계산하면 구간 $[t_n, t]$ 사이에 있는 ray 위의 모든 점들이 적분에 포함된다.

→ ray가 실제 객체와 접촉하는 것이 아닌, 공기와 접촉하는 불필요한 점들이 포함됨을 의미
→ 성능 저하

따라서, 일반적인 구분구적법 사용 못함

Volume Rendering



but, 불필요한 지점의 density까지 포함되기 때문에 성능 저하 발생

⇒ 위의 문제를 해결하기 위해서는 ray에서 임의로 점들을 추출한다.

하지만, 이 방법도 임의로 선택한 점들이 서로 연관 관계가 없어서 점들 간의 의미가 없을 수 있다.
추출한 점들의 분산이 커질 수 있어서 점들 간의 거리가 멀어질 수 있다.

→ 점들이 continuous한 값이 아닌 discrete(분리된) 한 값이다.

< 층화 표집 stratified sampling approach >

층화 표집 기법으로 앞선 문제들을 해결 한다.

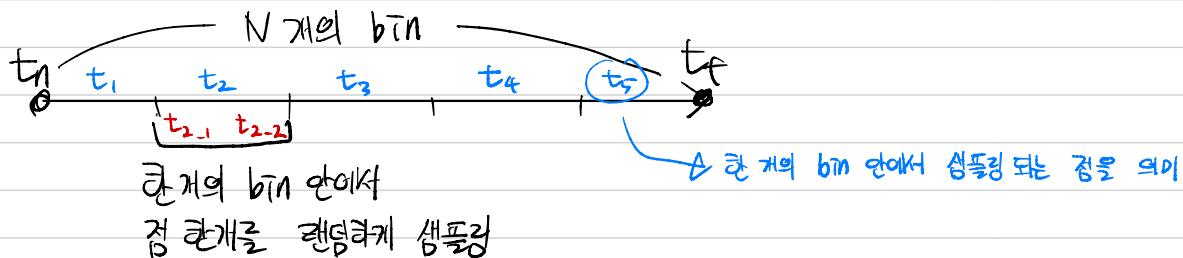
층화 표집 ; 모집단을 속성에 따라 계층으로 구분하고 해당 계층에서 각각 표본을 추출하는 방법

↳ 계층을 명확한 속성에 따라 나누었으면 계층별로 추출된 표본의 추정치가 정확하다.

따라서, 오브젝트를 지나는 ray 위의 유의미한 점들은 학습에 사용할 수 있다.

→ ray 위의 continuous한 함수의 점 t_i 들을 층화 표집으로 추출하는 수식

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



$$\left[t_n + \frac{2-1}{5}(t_f - t_n), t_n + \frac{2}{5}(t_f - t_n) \right]$$

$$= \left[t_n + \frac{1}{5}(t_f - t_n), t_n + \frac{2}{5}(t_f - t_n) \right]$$

이 구간 내에서 t_2 를 샘플링 할 수 있게 된다.

t_{i-1}, t_{i-2} 추출 : 위의 샘플링 방식으로 학습을 반복해서 진행하면
동일한 ray에 대해서 $C(n)$ 를 계산할 때마다 새로운 점을 샘플링 하기
때문에 결과적으로 continuous한 점들에 대해 학습이 가능해진다.

매번 새로운 점을 샘플링 하게 되는 이유 : 매번 학습할 때마다 이전에 샘플링했던 점을 다시 샘플링 할
확률이 매우 낮기 때문이라고 추측됨

↳ 학습하지 않음!

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

↓
수식 변화

$$\hat{C}(r) = \sum_{i=1}^N T_i (1 - \exp(-\sigma_i \delta_i)) c_i \quad \leftarrow \text{Optical Models for Direct}$$

Volume Rendering 참고

$$\begin{cases} T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right), \quad T(t) = \exp\left(-\int_{t_n}^t \sigma(s) ds\right) \\ \delta_i = t_{i+1} - t_i \\ t_i \sim U\left[t_n + \frac{i-1}{N}(t_f - t_n), t_n + \frac{i}{N}(t_f - t_n)\right] \end{cases}$$

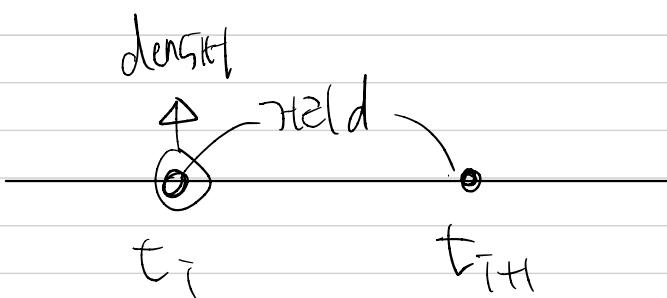
$\hat{C}(r)$ 는 $[t_n, t_f]$ 한 개를 N 개 구간으로 나누어서 각 구간마다 샘플링된
점들의 픽셀 정보를 누적한 값

시그마 $\sigma(r(t))$ 는 t 에서의 density

$r(t)$ 는 샘플링된 점 t

∴ $\sigma(r(t))$ 는 σ_i 로 표현 가능 ($1 \leq i \leq N$, 총 N 개의 정 샘플링)

델타 δ_i 는 t 와 그 다음 샘플링 지점 간의 거리



컴퓨터에서 적분계산은 적분기호 형태가 아닌 시그마 기호 수식에서 구분구적법으로 근사하게 계산해야 되기 때문에 볼륨 렌더링 기법에서는 다음과 같은 변환식(방법)을 많이 사용한다.
* 수식 변환 과정을 증명하려는 못했지만, 다음과 같이 변환한 이유는 현재 페인기에 서술 했다.

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

수식 변환

$$\hat{C}(r) = \sum_{i=1}^N T_i (1 - \exp(-\sigma_i \delta_i)) C_i \quad \leftarrow \text{Optical Models for Direct Volume Rendering 참고}$$

$$T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right) \quad \text{변환} \quad T(t) = \exp\left(-\int_{t_n}^t \sigma(s) ds\right)$$

$$\delta_i = t_{i+1} - t_i$$

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

$\sigma(r(t))$: 챕터링 이전의 모든 경로에서의 density 값

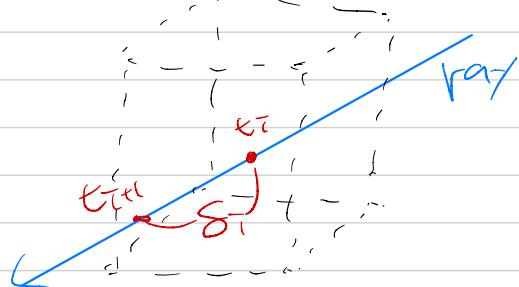
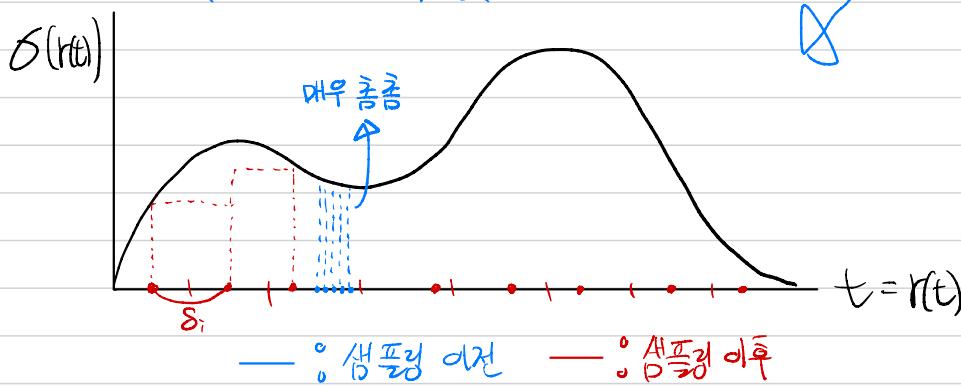
$\sigma_i \delta_i$: 챕터링 이후의 구분구적법(Quadrature)으로 정의된 t_i 에서의 density 값

직분기호에서
시그마 기호로
변환하면서
 $G(r(t))$ 에서
 $G_i \delta_i$ 로
변환된 이유

투명도 \downarrow , 투과율 \uparrow

T_i

$$\sum_{j=1}^i \sigma_j \delta_j (\text{투명도})$$

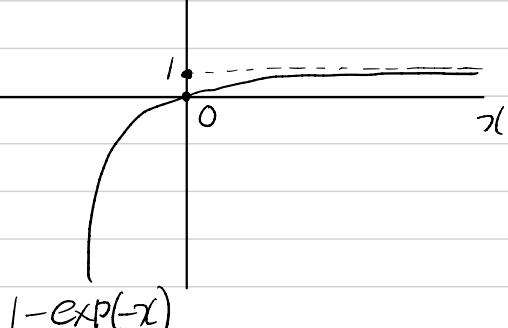


$\langle G(r(t)) \rightarrow 1 - \exp(-\sigma_i \delta_i) \text{ 변환 이유} \rangle$

$$\gamma = \sigma_i \delta_i$$

투명도 $\sigma_i \delta_i$ 의 범위는 ≥ 0 이다.

$\sigma_i \delta_i$ 값에 따라 $\hat{C}(r)$ 픽셀에 반영되는 영향은 투명도가 0 일 때는 빛이 더 이상 통과 할 수 없기 때문에 0을 곱해서 픽셀 정보를 없애준다. 투명도가 0 보다 클 때는 $0 < \gamma < 1$ 의 범위 내에서 만큼만 픽셀 정보에 영향을 준다.



따라서, 투명도 값에 따라 픽셀 정보에 주는 영향력의 크기를 $0 \sim 1$ 사이로 조절하기 위해서 $1 - \exp(-\sigma_i \delta_i)$ 의 수식을 써함