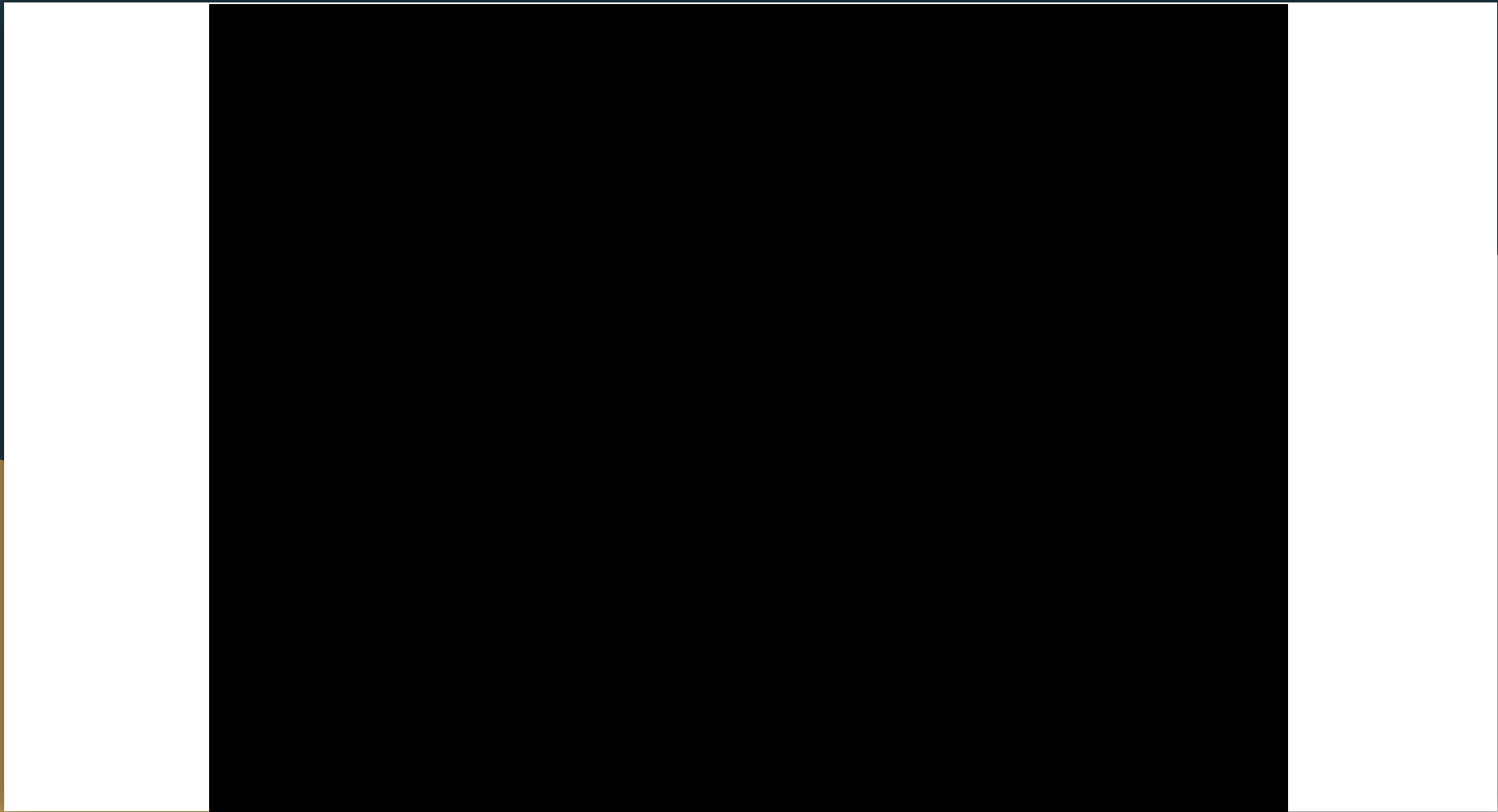


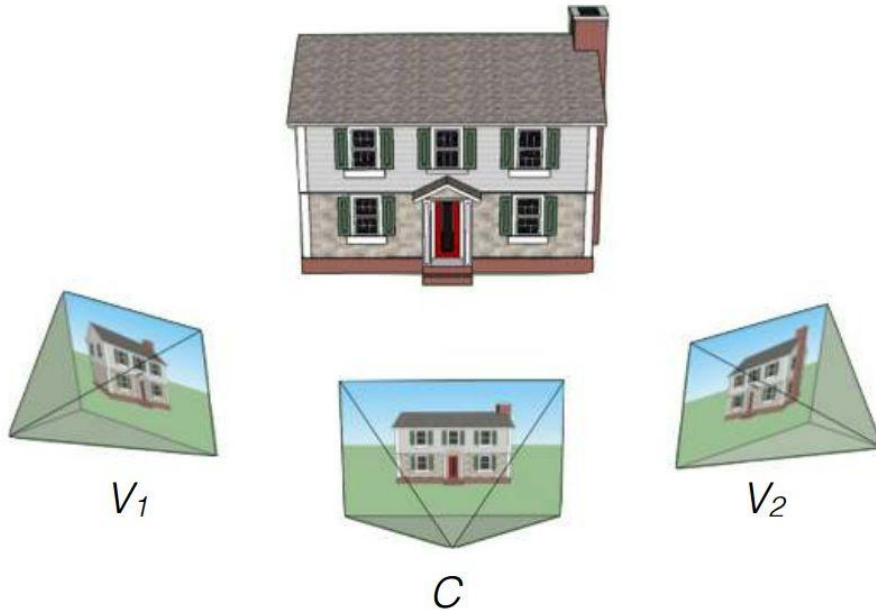
# NERF: Representing Scenes as Neural Radiance Fields for View Synthesis

Докладчик:  
Макоян Артем, БПМИ192





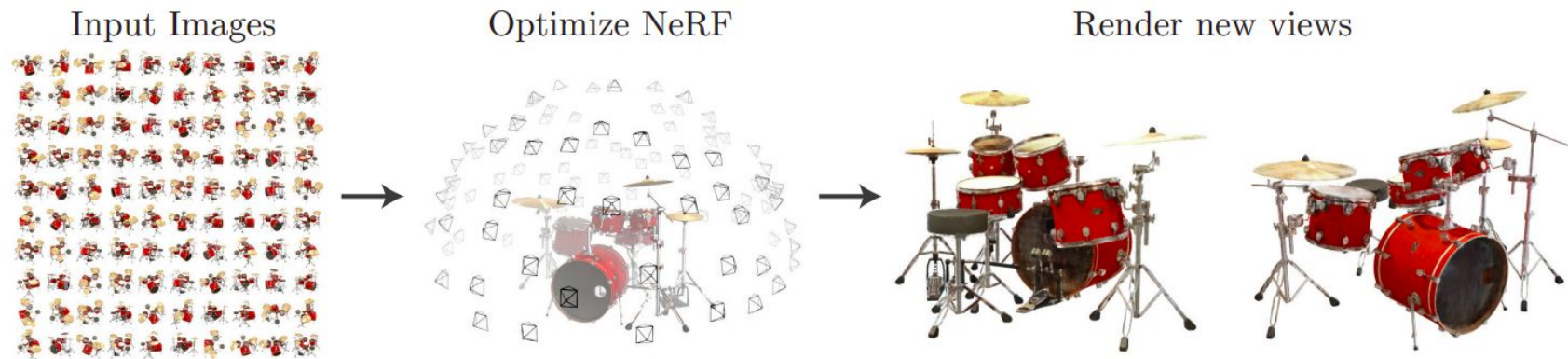
# View Synthesis



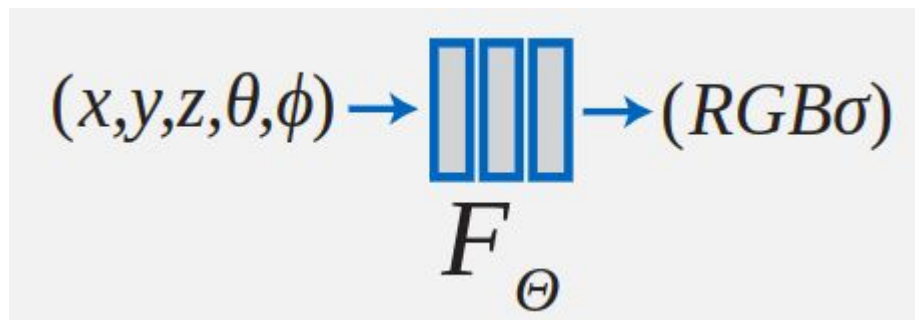
Задача: Есть объект. По позиции и параметрам камеры получить снимок объекта, как будто в этой позиции стояла камера.

# NeRF: идея

- Датасет: есть много объектов и для каждого много фотографий, снятых с разных сторон
- Каждый инстанс нейросети отвечает за свой объект (можно сказать, мы “переобучаем” сеть на фотографиях одного объекта)
- С помощью сети мы сможем генерировать новые картинки
- По сути, веса сети “кодируют” объект



# NeRF: подробности



На вход: точка и взгляд на нее -  $(x, y, z, \theta, \phi)$

$(x, y, z)$  - координаты точки

$(\theta, \phi)$  - координаты взгляда на нее

Пока не очень понятно, что такое “взгляд”!

На выход: цвет точки и ее плотность -  $(RGB, \sigma)$

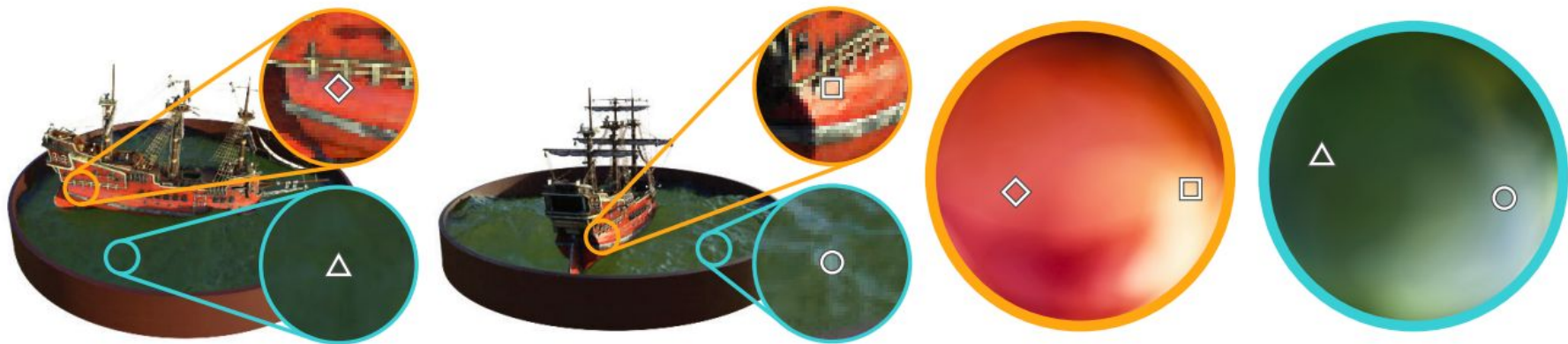
RGB - цвет точки при данном взгляде в формате RGB

$\sigma$  - плотность точки

Пока не очень понятно, что такое “плотность”!

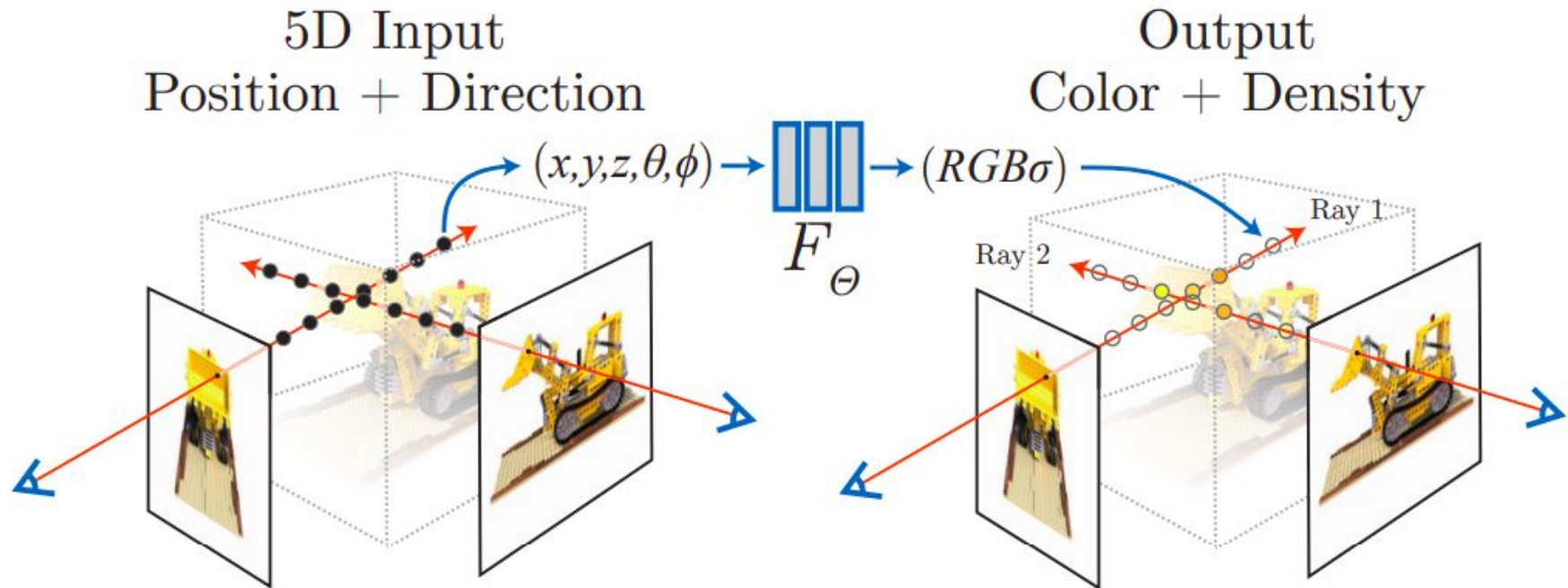
# “Взгляд” на точку

Цвет точки определяется не только ее координатами, но и тем, с какой стороны мы на нее смотрим. Допустим на точку может по-разному падать свет. Также есть полупрозрачные объекты.



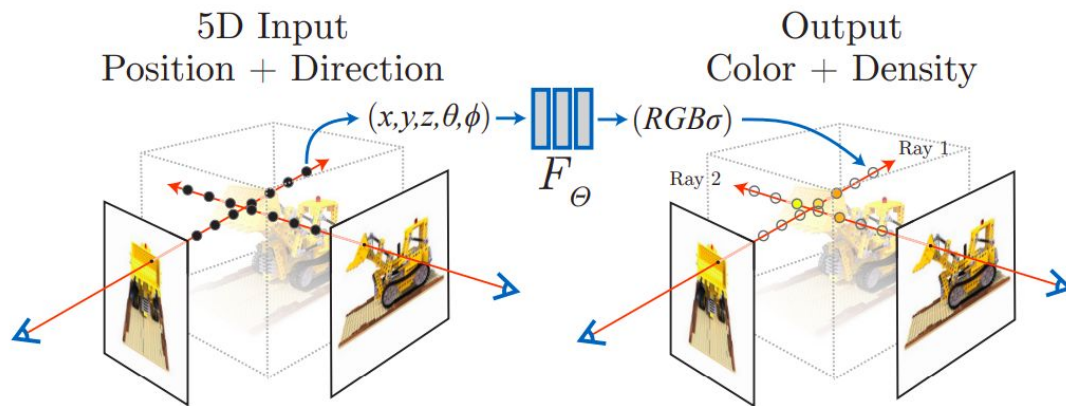


# NeRF: получение новых картинок





# NeRF: получение новых картинок

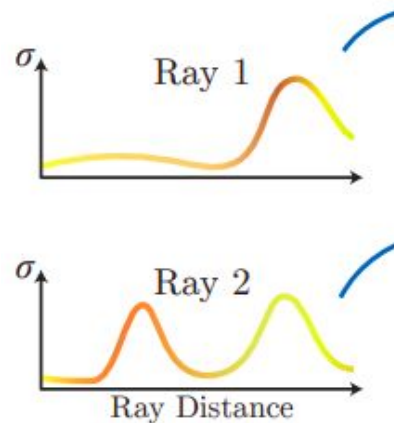


- В каждый пиксель проводим луч из центра камеры
- На луче генерируем много точек, направление взгляда на точку берем по лучу
- Предсказываем цвета точек и их плотность
- Усредняем все цвета точек с коэффициентами (позже подробнее)

# Плотность точки

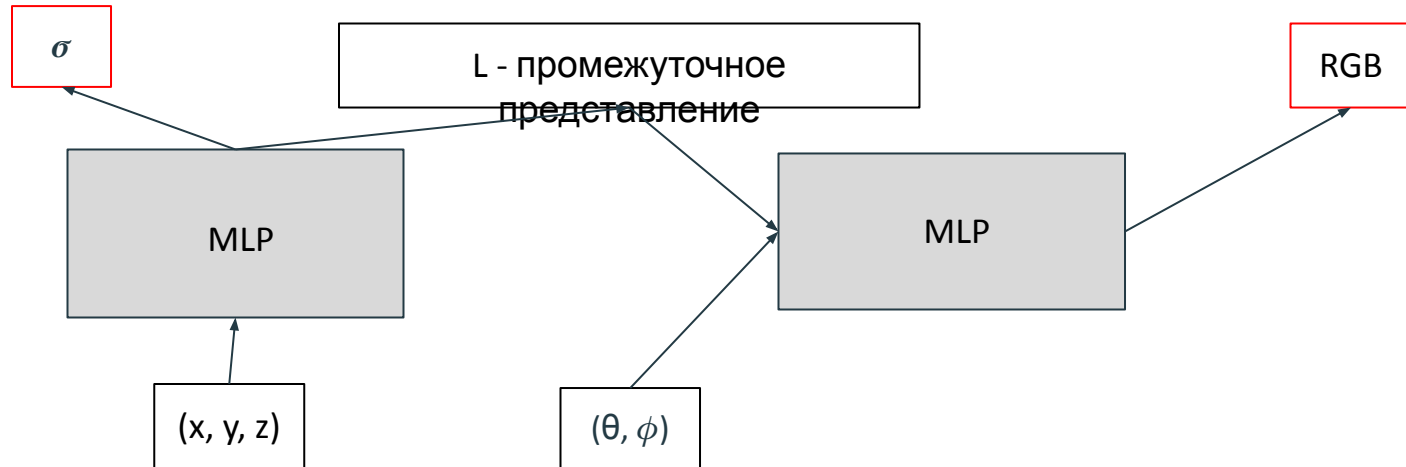
Из-за полупрозрачных точек нужно учитывать не одну точку на луче, а все. В каком-то смысле, плотность точки показывает какой вклад вносит точка в итоговый пиксель луча.

## Volume Rendering



# NeFR: архитектура

- Архитектура сети довольно простая: просто MLP
- Заметим, что плотность точки не зависит от взгляда на нее



# Volume Rendering with Radiance Fields

Как же все-таки получить цвет точки по цветам и плотностям точек на луче?

The volume density  $\sigma(\mathbf{x})$  can be interpreted as the differential probability of a ray terminating at an infinitesimal particle at location  $\mathbf{x}$ . The expected color  $C(\mathbf{r})$  of camera ray  $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$  with near and far bounds  $t_n$  and  $t_f$  is:

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t) \sigma(\mathbf{r}(t)) \mathbf{c}(\mathbf{r}(t), \mathbf{d}) dt, \text{ where } T(t) = \exp\left(-\int_{t_n}^t \sigma(\mathbf{r}(s)) ds\right). \quad (1)$$

Смысл  $\sigma(x)$  - вероятность того, что луч не сможет пройти сквозь точку  $x$

Смысл  $T(t)$  - вероятность того, что луч дойдет до координаты  $t$

# Volume Rendering with Radiance Fields

Считаем интеграл с помощью квадратуры. Детерминированный способ ограничивает представление нашей модели, поэтому разделим луч (а скорее отрезок  $[t_n, t_f]$ ) на  $N$  равных частей и насэмплим в каждом из них по точке из равномерного распределения на части.

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

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^N T_i (1 - \exp(-\sigma_i \delta_i)) \mathbf{c}_i, \text{ where } T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right), \quad (3)$$




where  $\delta_i = t_{i+1} - t_i$  is the distance between adjacent samples. This function for calculating  $\hat{C}(\mathbf{r})$  from the set of  $(\mathbf{c}_i, \sigma_i)$  values is trivially differentiable and reduces to traditional alpha compositing with alpha values  $\alpha_i = 1 - \exp(-\sigma_i \delta_i)$ .

# Positional Encoding

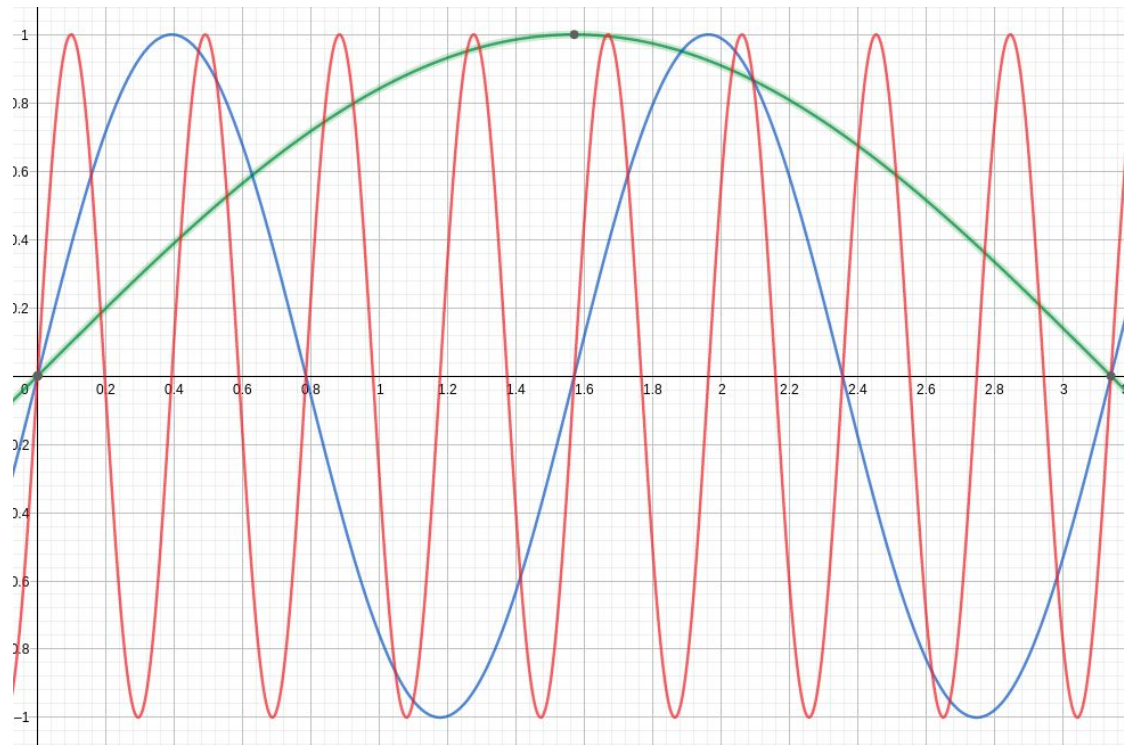
$$\gamma(p) = (\sin(2^0 \pi p), \cos(2^0 \pi p), \dots, \sin(2^{L-1} \pi p), \cos(2^{L-1} \pi p))$$

- Для каждой координаты (x, y, z) функция  $\gamma$  применяется отдельно и получаем вектор размерностью  $(2L) \times 3$
- После нормализации считаем, что  $-1 \leq x, y, z \leq 1$
- Цель такого кодирования отличается от аналогичного в трансформерах: мы хотим, чтобы наша модель умела приближать высокочастотные функции

# Postional Encoding

|   |                    |
|---|--------------------|
|  | $f(x) = \sin(x)$   |
|  | $g(x) = \sin(4x)$  |
|  | $h(x) = \sin(16x)$ |

Учитывая все кривые, мы можем отслеживать как маленькие изменения в координатах, так и большие





# Результаты оптимизаций



Ground Truth



Complete Model



No View Dependence



No Positional Encoding

# Hierarchical volume sampling: идея

Для каждого пикселя мы генерируем очень много точек! Что делать?

Сделаем 2 сети: одна отвечает за “грубое” (coarse) приближение, другая за более “тонкое” (fine)

- Сначала по старой стратегии генерируем точки  $p_i$ , “грубой” нейросетью считаем цвета и плотности точек
- Затем вокруг плотных точек генерируем точки  $q_i$ , и “тонкой” нейросетью считаем цвета и плотности точек  $p_i$  и  $q_i$
- Итоговый ответ - считаем цвет старой стратегией для выхода “тонкой” сети

# Hierarchical volume sampling: подробности

Генерируем точки  $\mathbf{p}_i$ , для них чуть перепишем известное равенство

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

на

$$\hat{C}_c(\mathbf{r}) = \sum_{i=1}^{N_c} w_i \mathbf{c}_i, \quad w_i = T_i (1 - \exp(-\sigma_i \delta_i))$$

Тогда нормализовав  $w_i$  мы получаем кусочно-постоянное распределение на прямой, из которой будем генерировать точки.

Заметим, что чем больше вес, тем плотнее там точки

# NeRF: обучение

- Каждую сеть “переобучаем” на одном объекте. Для каждого объекта нужны картинки с позициями и параметрами камер, а также границы объекта
- Из каждой картинки получаем батч лучей с цветами (для какого-то множества пикселей)
- Далее делаем Hierarchical volume sampling, пересчитываем цвета с помощью Volume Rendering
- Лосс,  $C_c$  - цвет из “грубой” или coarse сети,  $C_f$  - цвет из “тонкой” или fine сети

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



*Ship*



*Lego*



*Microphone*



*Materials*



Ground Truth   NeRF (ours)   LLFF [28]   SRN [42]   NV [24]



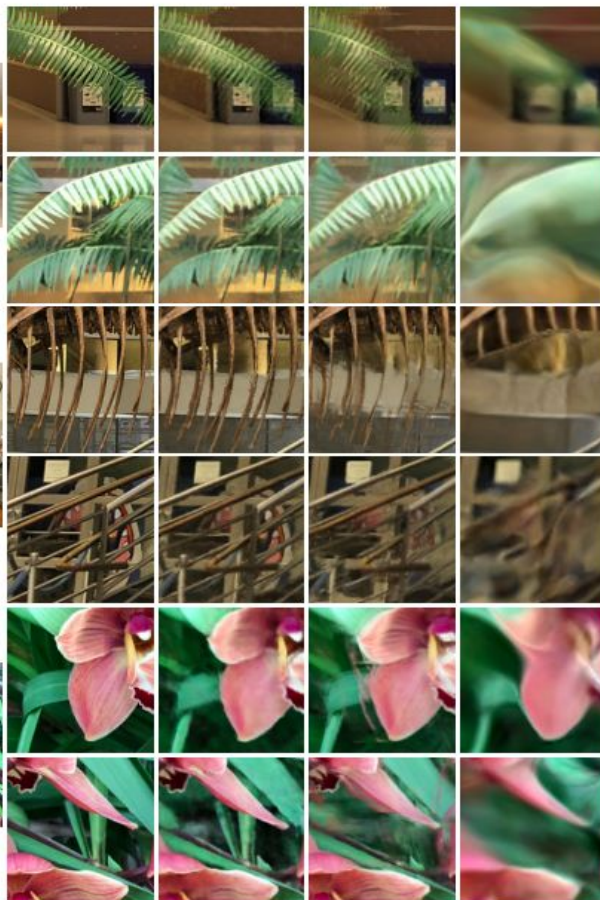
*Fern*



*T-Rex*



*Orchid*



Ground Truth

NeRF (ours)

LLFF [28]

SRN [42]