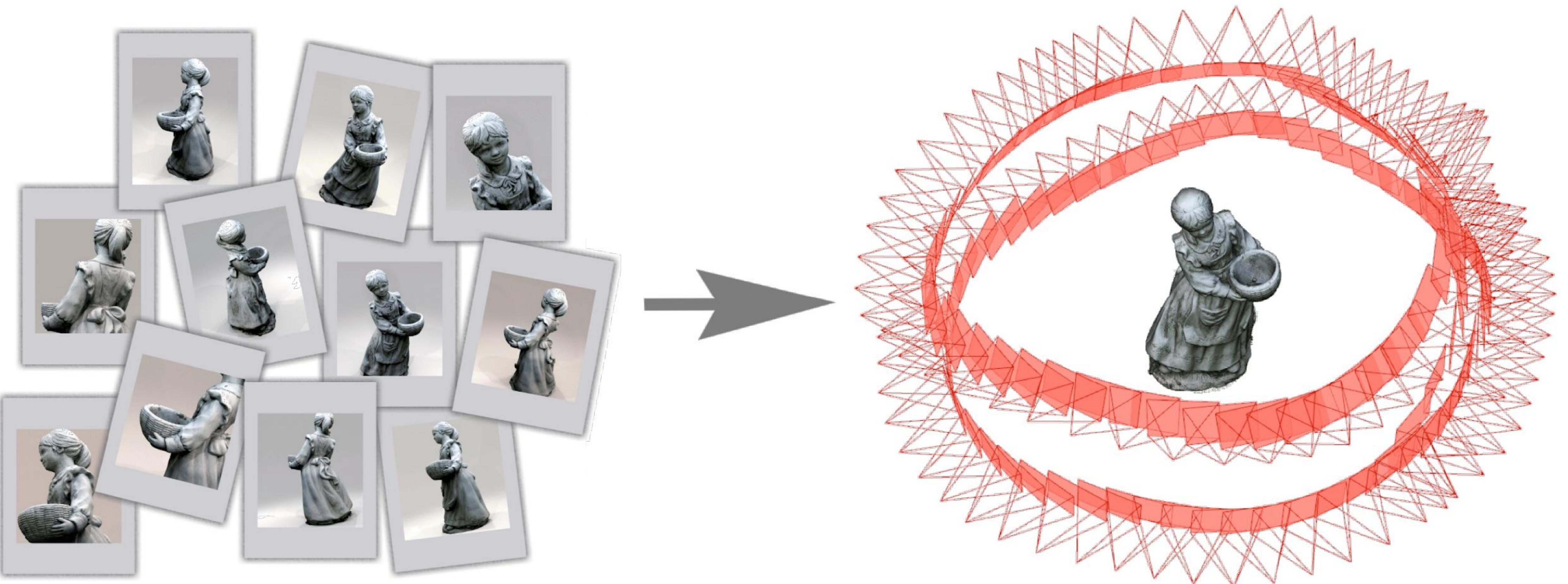


# **COLMAP + BARF**

**3D реконструкция сцен**

# COLMAP

**Structure from Motion (SfM)- процесс построения 3D сцен из набора фотографий**



# Постановка задачи

- Задача COLMAP - понять, где была сделана фотография и сделать реконструкцию
- Позиции камер неизвестны, либо измерены неаккуратно



# SfM Pipeline

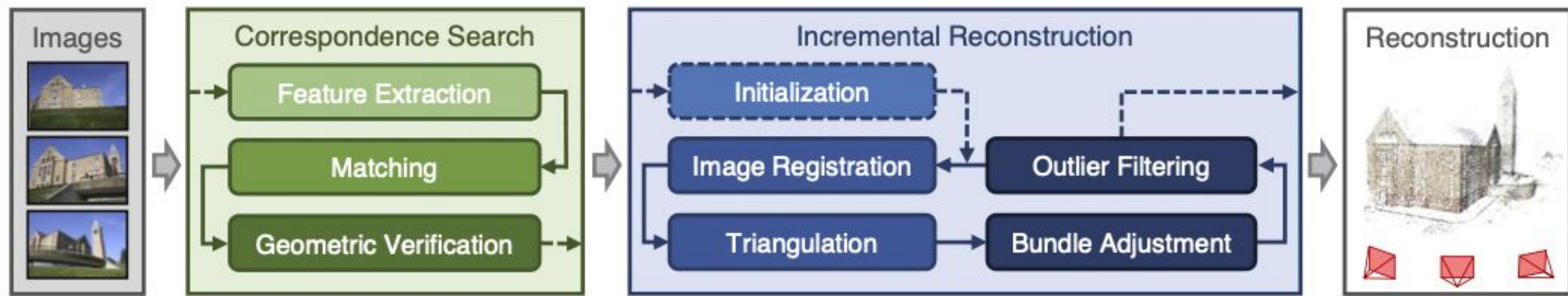
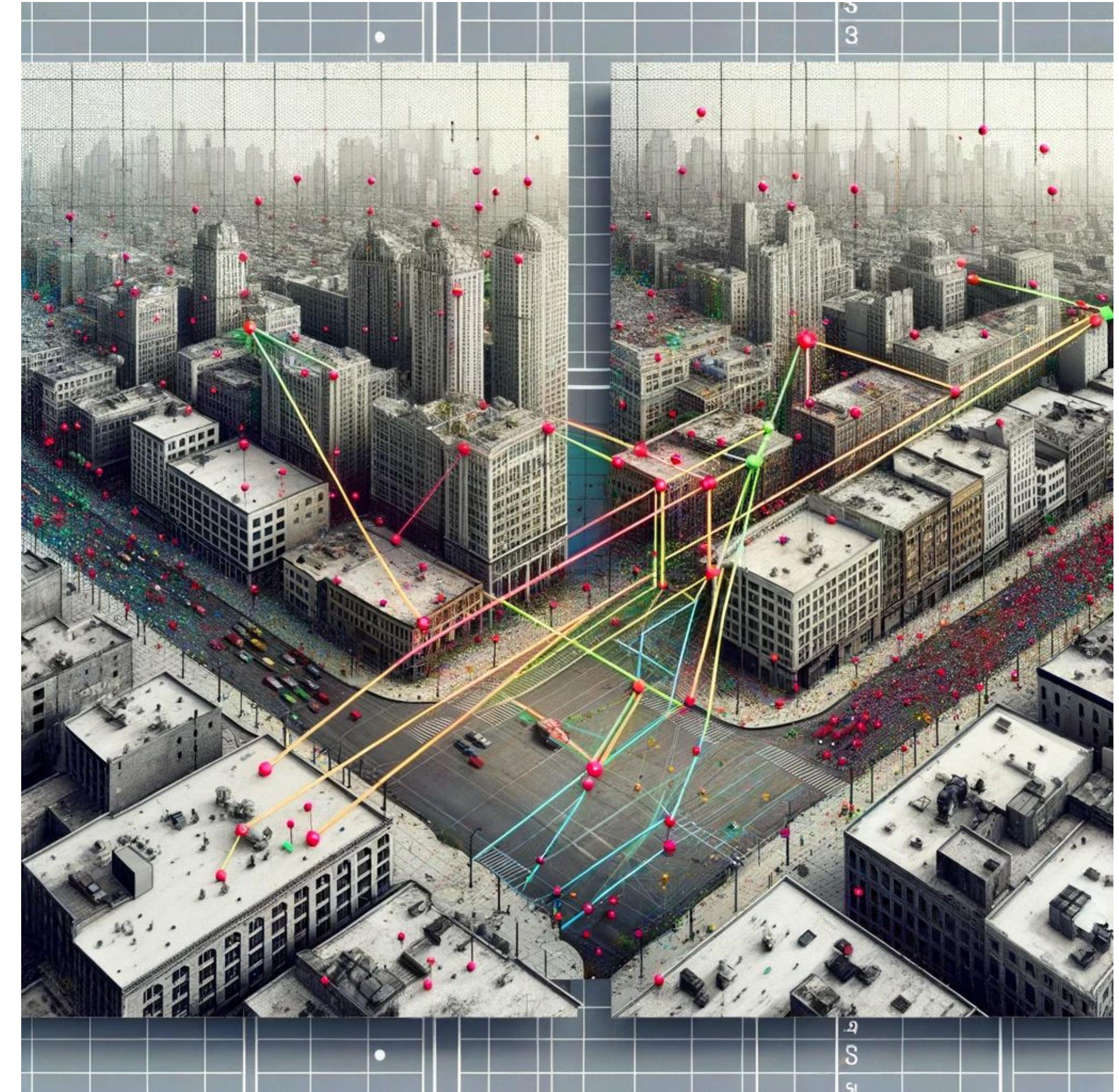
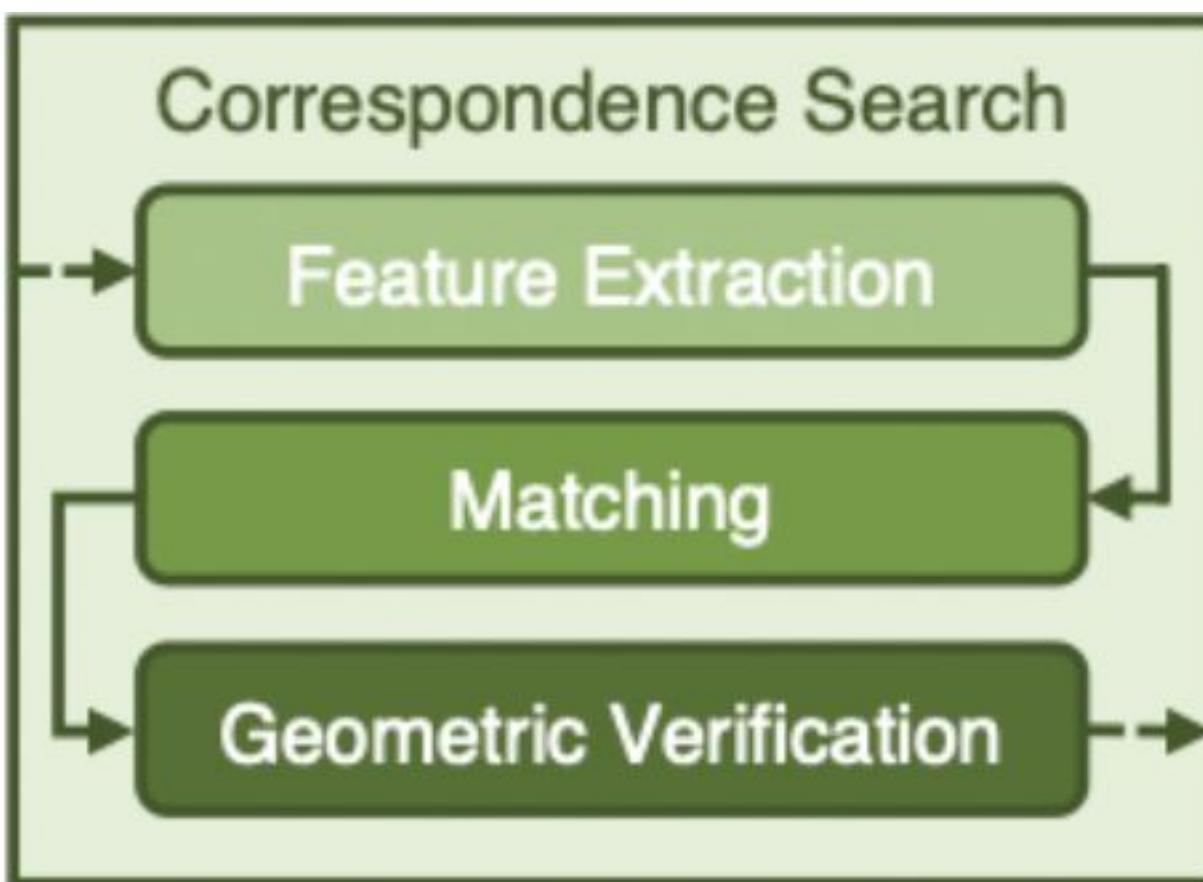


Figure 2. Incremental Structure-from-Motion pipeline.

# Correspondence Search

Поиск соответствия обнаруживает перекрытие сцены во входных изображениях

Хотим сгруппировать фото так, чтобы каждая группа отвечала за конкретную область сцены

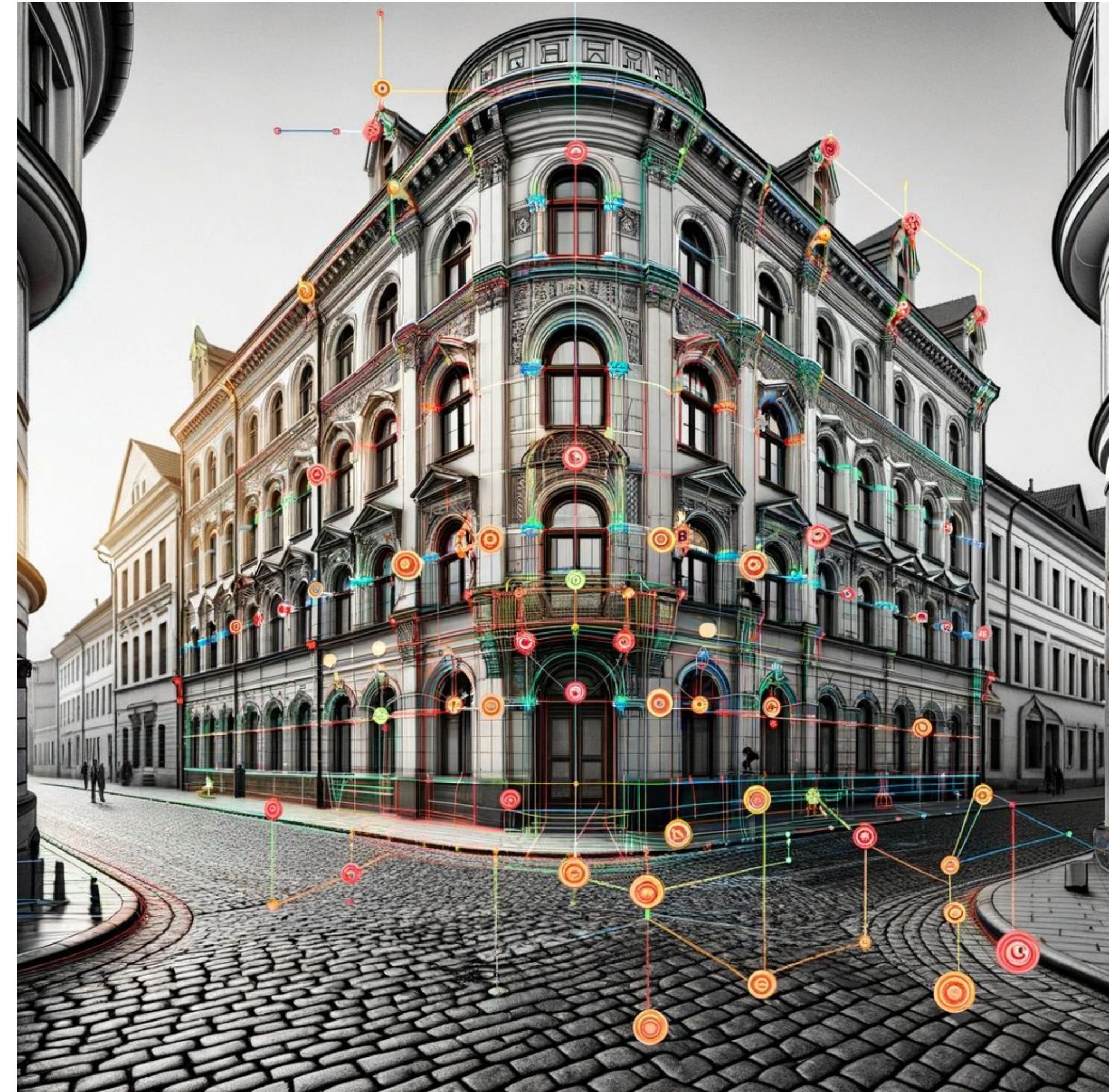


# Correspondence Search (feature extraction)

Каждому изображению сопоставляем множество пар

$$\mathcal{F}_i = \{(\mathbf{x}_j, \mathbf{f}_j) \mid j = 1 \dots N_{F_i}\}$$

- $\mathbf{x}$  - это какой то локальный признак
- $\mathbf{f}$  - это дескриптор внешнего вида

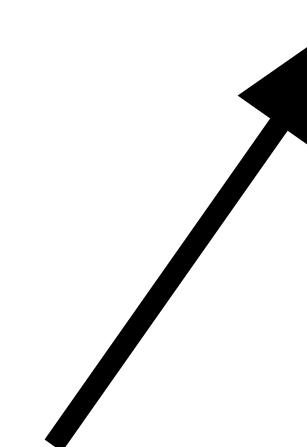


# Correspondence Search (matching)

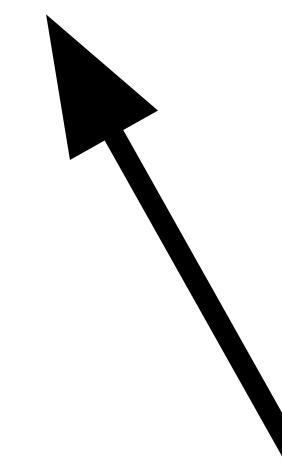
**SfM** ищет изображения, которые смотрят на одну и ту же часть сцены, используя дескрипторы в качестве описания внешнего вида изображений.

**Input:**  $\mathcal{F}_i = \{(\mathbf{x}_j, \mathbf{f}_j) \mid j = 1 \dots N_{F_i}\}$

**Output:**  $C = \{(I_a, I_b) \mid I_a, I_b \in I, a < b\}$        $\mathcal{M}_{ab} \in \mathcal{F}_a \times \mathcal{F}_b$



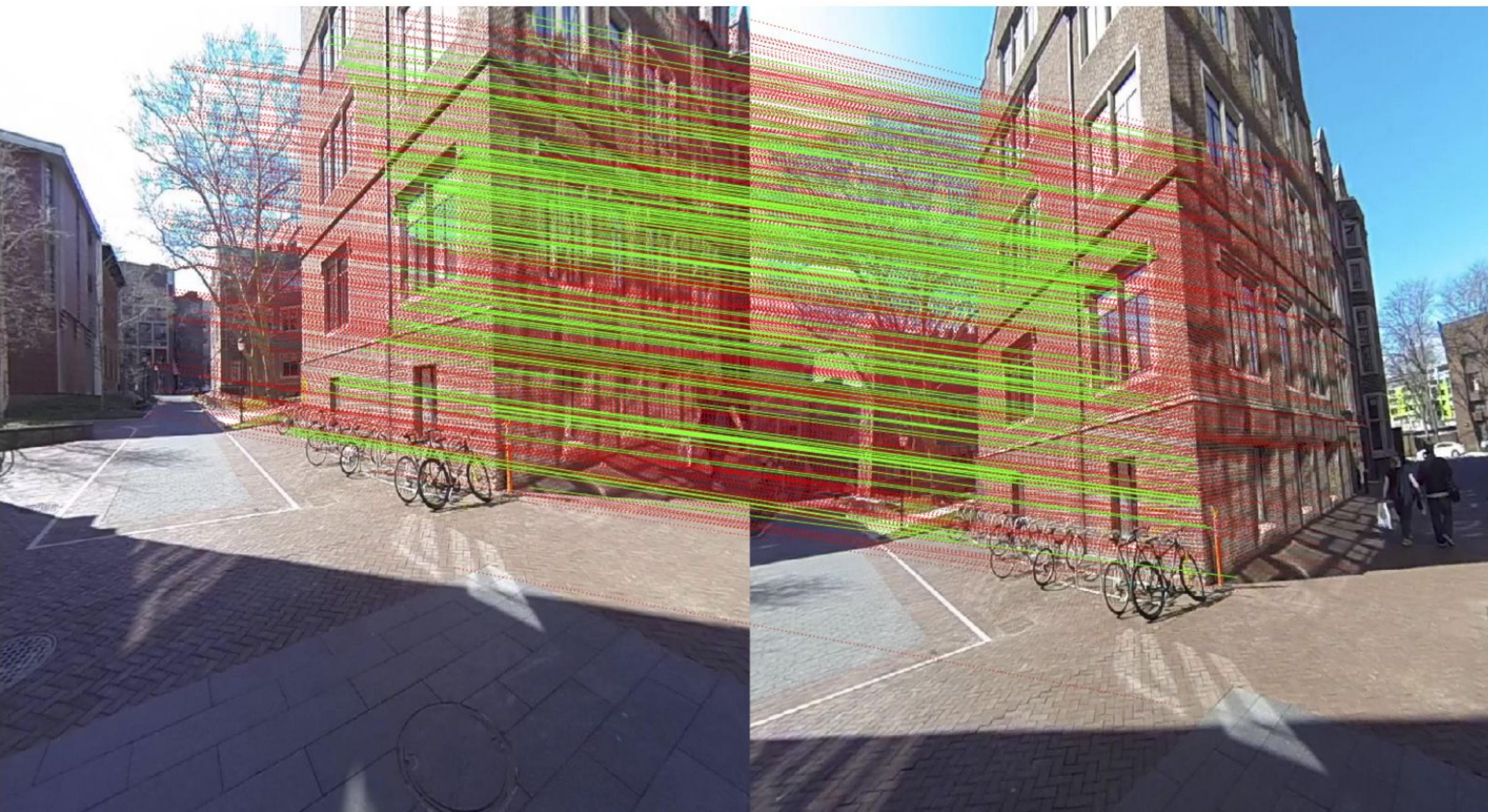
набор пар изображений с большим кол-вом совпадающих фичей



Множество фичей которые смэтчились

# Correspondence Search (geometric verification)

**Output:**  $\mathcal{F}_i = \{(\mathbf{x}_j, \mathbf{f}_j) \mid j = 1 \dots N_{F_i}\}$  + геом. соотношение + граф сцены



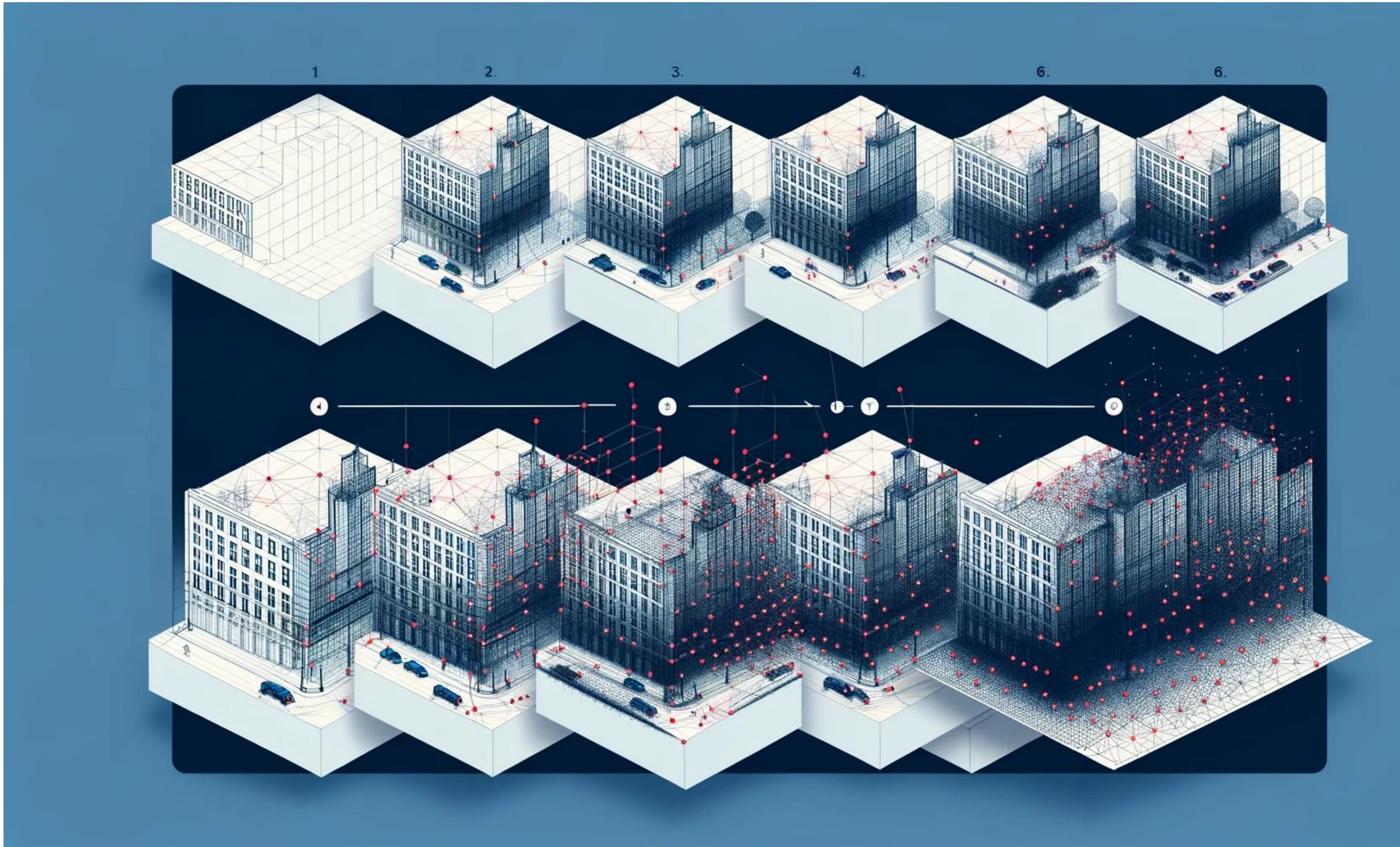
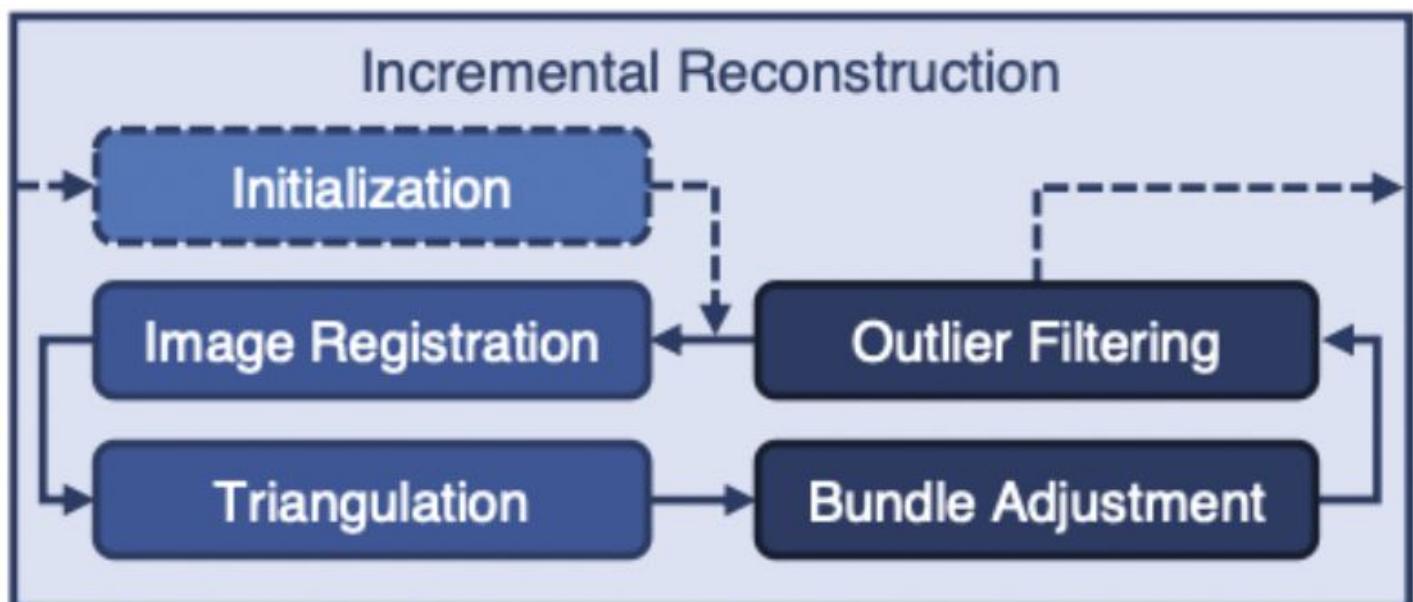
# Incremental Reconstruction

## Input:

- Граф сцены

## Output:

- Оценки на угол обзора
  - Реконструкция как множество пикселей в  $R^3$



# Incremental Reconstruction (initialisation)

- **Должна быть инициализирована двумя изображениями**
- **Важно использовать фото с перекрытиями, где много дескрипторов**
- **Много перекрытий -> точная реконструкция**
- **Мало перекрытий -> реконструкция не получится**

# Incremental Reconstruction (image registration)

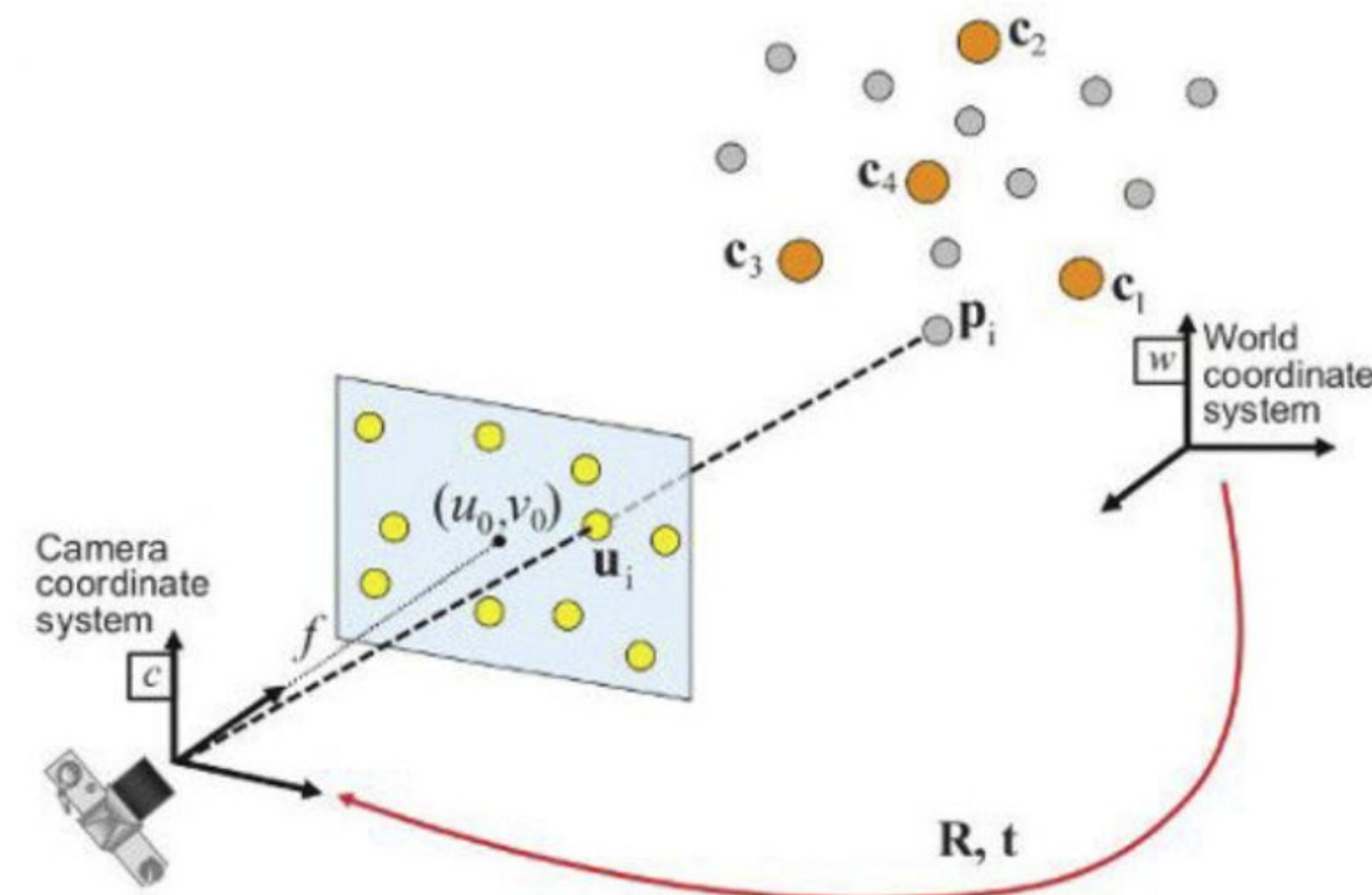
- Добавляем изображения по одному
- Решаем PnP
- Получаем расположение камеры

**ВАЖНО:** next best image selection

Цель: выбрать такое изображение, которое принесёт наибольший вклад в модель,  
например, добавит новые точки

# Incremental Reconstruction (triangulation)

Добавляем новые точки на сцену



# Incremental Reconstruction (Bundle Adjustment)

**Уточняем трехмерные координаты точек и позиции камер в трехмерной реконструкции**

**Хотим минимизировать функционал:**

$$E = \sum_j \rho_j \left( \|\pi(\mathbf{P}_c, \mathbf{X}_k) - \mathbf{x}_j\|_2^2 \right)$$

$\pi$  - функция, которая проецирует точки сцены и позиции камеры в пространство на изображении

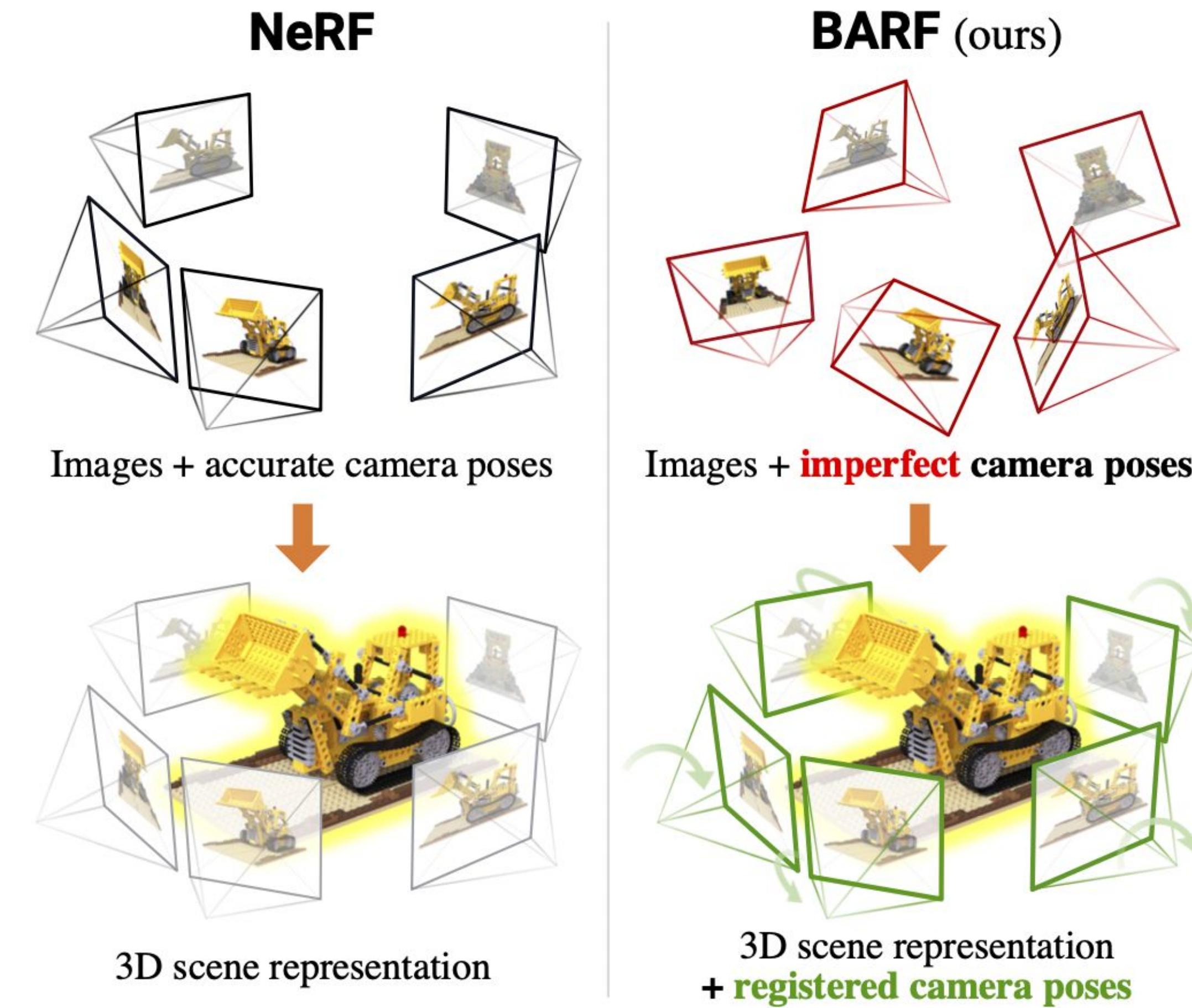
$\mathbf{X}_k$  - координаты трехмерной точки в пространстве

$\mathbf{P}_c$  - позиция камеры

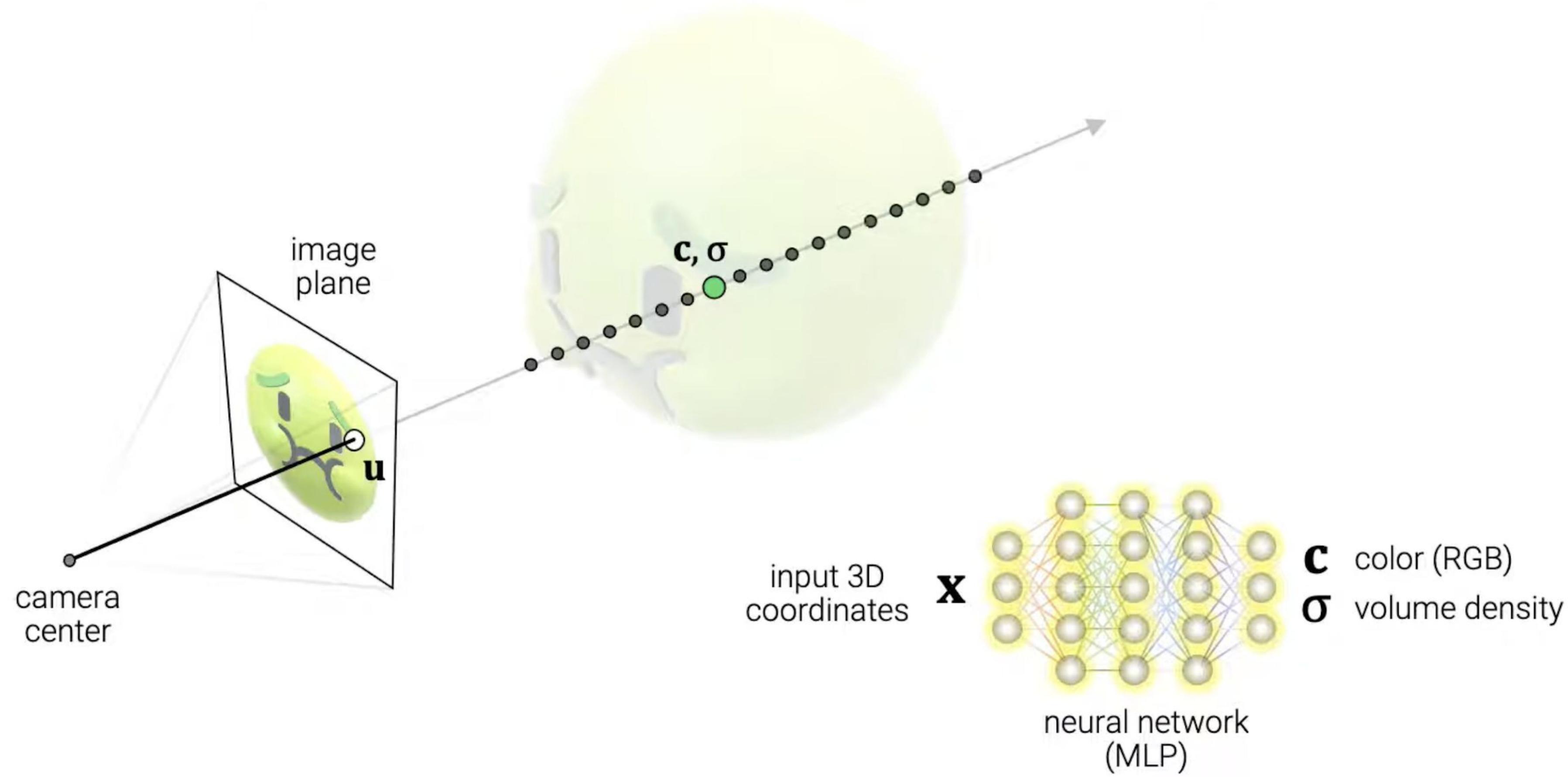
$\mathbf{x}_j$  - наблюдаемые двумерные координаты точки на изображении

$\rho_j$  - функция взвешивания, которая уменьшает влияние выбросов на итоговый результат оптимизации

# BARF: 😤 Bundle-Adjusting Neural Radiance Fields



# NeRF



# NeRF Loss Function

**volume rendering (RGB цвет в пикселе u)**

$$\hat{\mathcal{I}}(\mathbf{u}) = \int_{z_{\text{near}}}^{z_{\text{far}}} T(\mathbf{u}, z) \sigma(z\bar{\mathbf{u}}) \mathbf{c}(z\bar{\mathbf{u}}) dz$$

**такой интеграл на практике приближается с помощью сэмплов точек на разной глубине  $\{z_1, \dots, z_N\}$**

**для каждой глубины получаем у - выход MLP = f(x)**

**таким образом мы можем переписать интеграл:**

$$\hat{\mathcal{I}}(\mathbf{u}) = g(\mathbf{y}_1, \dots, \mathbf{y}_N) \quad g : \mathbb{R}^{4N} \rightarrow \mathbb{R}^3$$

# NeRF Loss Function

**У каждой 6-DoF (с 6 степенями свободы) камеры есть параметры  $\mathbf{p} \in \mathbb{R}^6$**

**Возьмем точку  $\mathbf{x}$  в координатах камеры и преобразуем ее в 3D координаты сцены с помощью  $\mathcal{W} : \mathbb{R}^3 \rightarrow \mathbb{R}^3$**

**Получим выражение для volume rendering следующего вида:**

$$\hat{\mathcal{I}}(\mathbf{u}; \mathbf{p}) = g\left(f(\mathcal{W}(z_1 \bar{\mathbf{u}}; \mathbf{p}); \Theta), \dots, f(\mathcal{W}(z_N \bar{\mathbf{u}}; \mathbf{p}); \Theta)\right)$$

Камера может перемещаться вперед, назад, влево и вправо, вверх и вниз (это три степени свободы для перемещения), а также вращаться вокруг трех перпендикулярных осей: вокруг оси X (крен), вокруг оси Y (тангаж) и вокруг оси Z (рыскание)

# NeRF Loss Function

$$\min_{\Theta} \sum_{i=1}^M \sum_{\mathbf{u}} \left\| \hat{\mathcal{I}}(\mathbf{u}; \mathbf{p}_i, \Theta) - \mathcal{I}_i(\mathbf{u}) \right\|_2^2$$

frames  
 $\Theta$   
 $i = 1$   
 $\mathbf{u}$   
RGB  
(rendered)  
camera  
pose  
network  
params.  
RGB

- $\hat{I}(u; p_i, \Theta)$  — это предсказанное NeRF значение цвета (RGB) для пикселя с координатами  $u$  на изображении, которое получается в результате рендеринга с использованием параметров камеры  $p_i$  и параметров нейросети  $\Theta$ .
- $I_i(u)$  — это фактическое значение цвета (RGB) того же пикселя  $u$  на реальном изображении  $i$ .
- $\sum_u$  — это суммирование по всем пикселям изображения.
- $\sum_{i=1}^M$  — это суммирование по всем изображениям в датасете, где  $M$  — это количество кадров или изображений.

# BARF Loss Function

$$\min_{\Theta} \sum_{i=1}^M \sum_{\mathbf{u}} \left\| \hat{\mathcal{I}}(\mathbf{u}; \mathbf{p}_i, \Theta) - \mathcal{I}_i(\mathbf{u}) \right\|_2^2 \rightarrow \min_{\mathbf{p}_1, \dots, \mathbf{p}_M, \Theta} \sum_{i=1}^M \sum_{\mathbf{u}} \left\| \hat{\mathcal{I}}(\mathbf{u}; \mathbf{p}_i, \Theta) - \mathcal{I}_i(\mathbf{u}) \right\|_2^2$$

frames

$\Theta$

$\mathbf{p}_i$

$\mathbf{u}$

$\hat{\mathcal{I}}(\mathbf{u}; \mathbf{p}_i, \Theta)$

$\mathcal{I}_i(\mathbf{u})$

RGB  
(rendered)

camera  
pose

network  
params.

RGB

$\Theta$

$\mathbf{p}_i$

$\mathbf{u}$

$\hat{\mathcal{I}}(\mathbf{u}; \mathbf{p}_i, \Theta)$

$\mathcal{I}_i(\mathbf{u})$

RGB  
(rendered)

camera  
pose

network  
params.

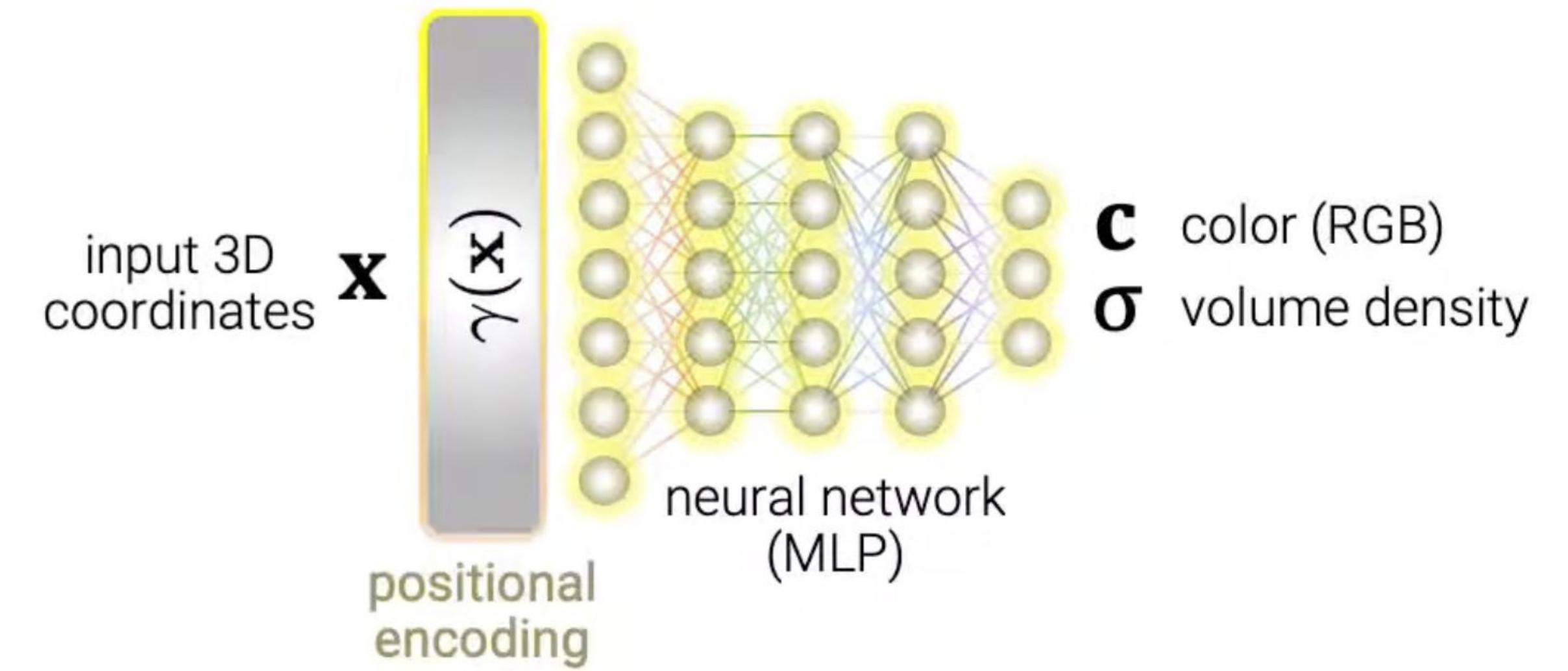
RGB

**Просто добавим к обучаемым параметрам нейросети еще и параметры расположения камер**

# Positional Encoding

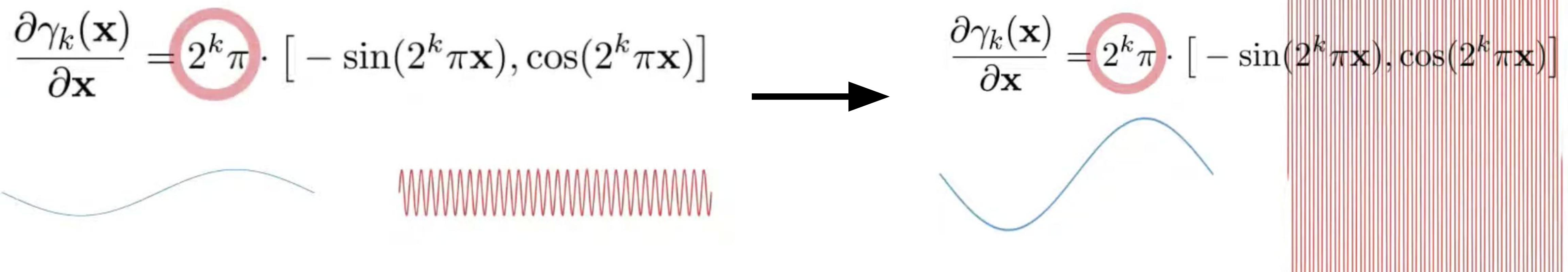
$$\gamma(\mathbf{x}) = \begin{bmatrix} \gamma_0(\mathbf{x}) \\ \gamma_1(\mathbf{x}) \\ \gamma_2(\mathbf{x}) \\ \vdots \\ \gamma_{L-1}(\mathbf{x}) \end{bmatrix}$$

where  $\gamma_k(\mathbf{x}) = [\cos(2^k\pi\mathbf{x}), \sin(2^k\pi\mathbf{x})]$



# Positional Encoding Problems

Мы очень хотим использовать Positional Encoding, но есть проблема...



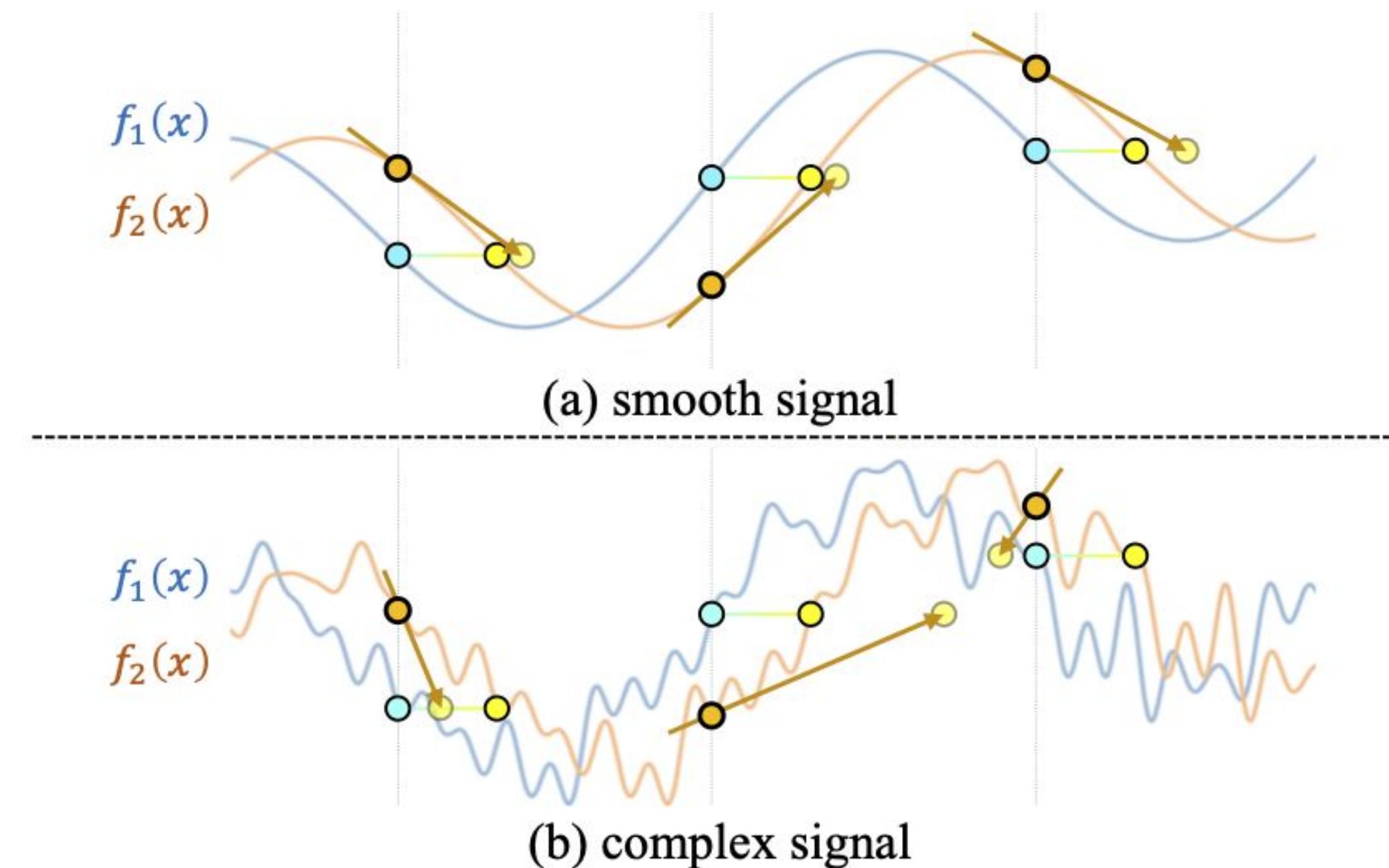
Градиенты становятся очень большие по норме, что мешает обучению

# Positional Encoding Problems

Рассмотрим игрушечный пример.

Пусть у нас есть два сигнала, которые отличаются сдвигом.

Представим, что нам нужно использовать значение сигналов и их градиентов чтобы найти сдвиг

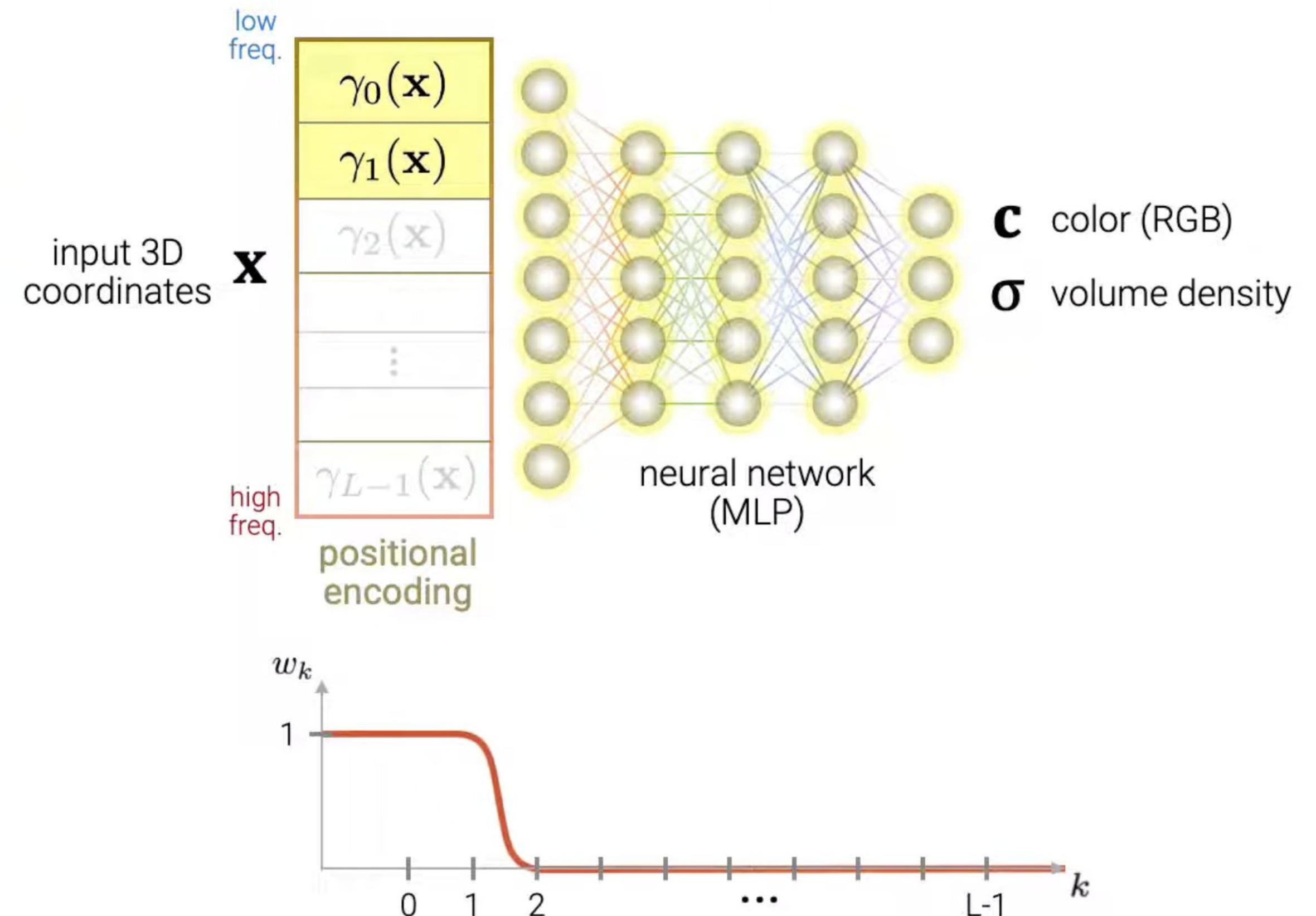


# Solution

$$\gamma_k(\mathbf{x}) = w_k \cdot [\cos(2^k \pi \mathbf{x}), \sin(2^k \pi \mathbf{x})] \in [0,1]$$

$$\text{where } w_k(\alpha) = \begin{cases} 0 & \text{if } \alpha < k \\ \frac{1 - \cos((\alpha - k)\pi)}{2} & \text{if } 0 \leq \alpha - k < 1 \\ 1 & \text{if } \alpha - k \geq 1 \end{cases}$$

$\alpha \in [0, L]$  is a controllable parameter



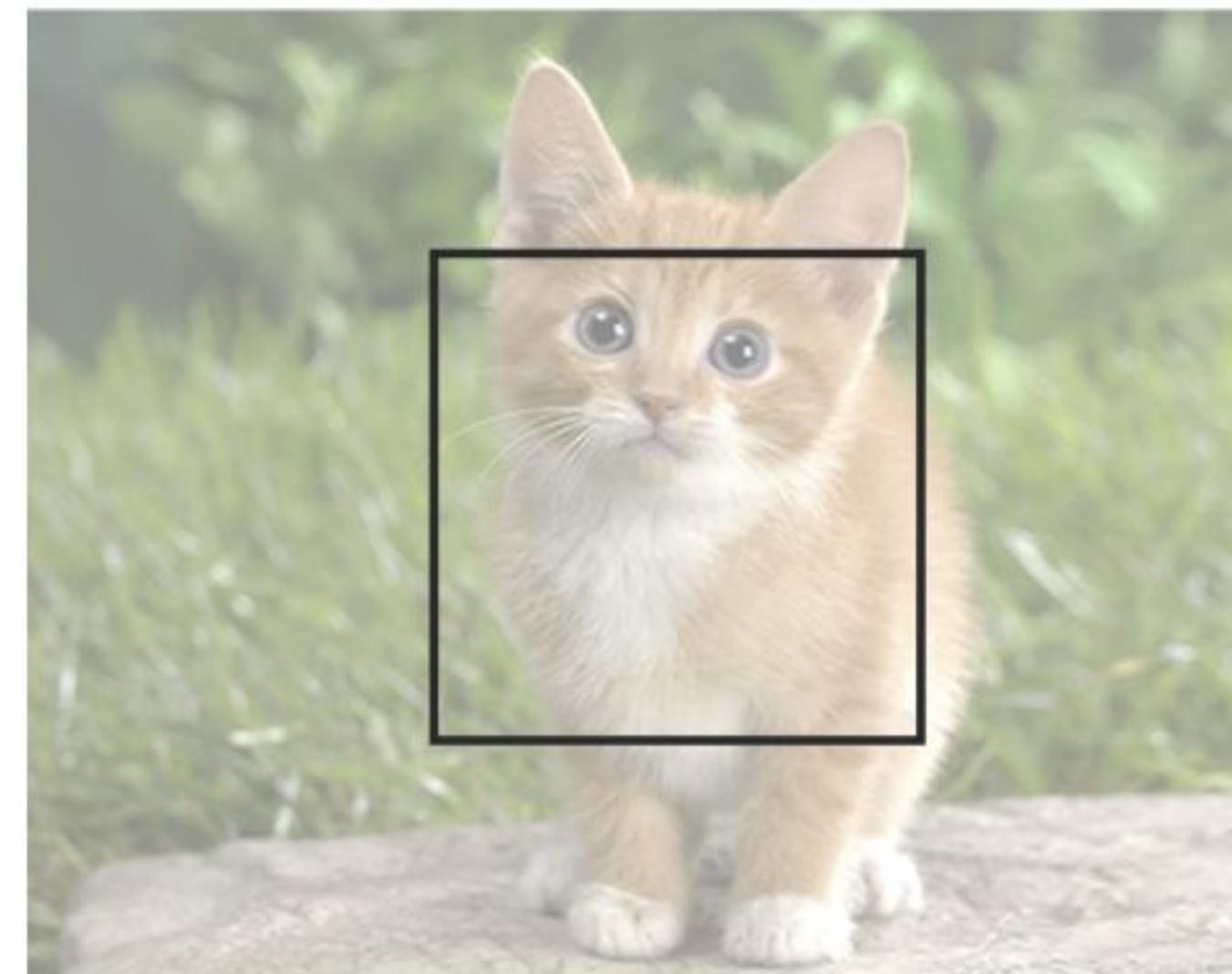
Как только все encoding активировались - получаем такой же Positional encoding, как в NeRF

# 2D Примеры

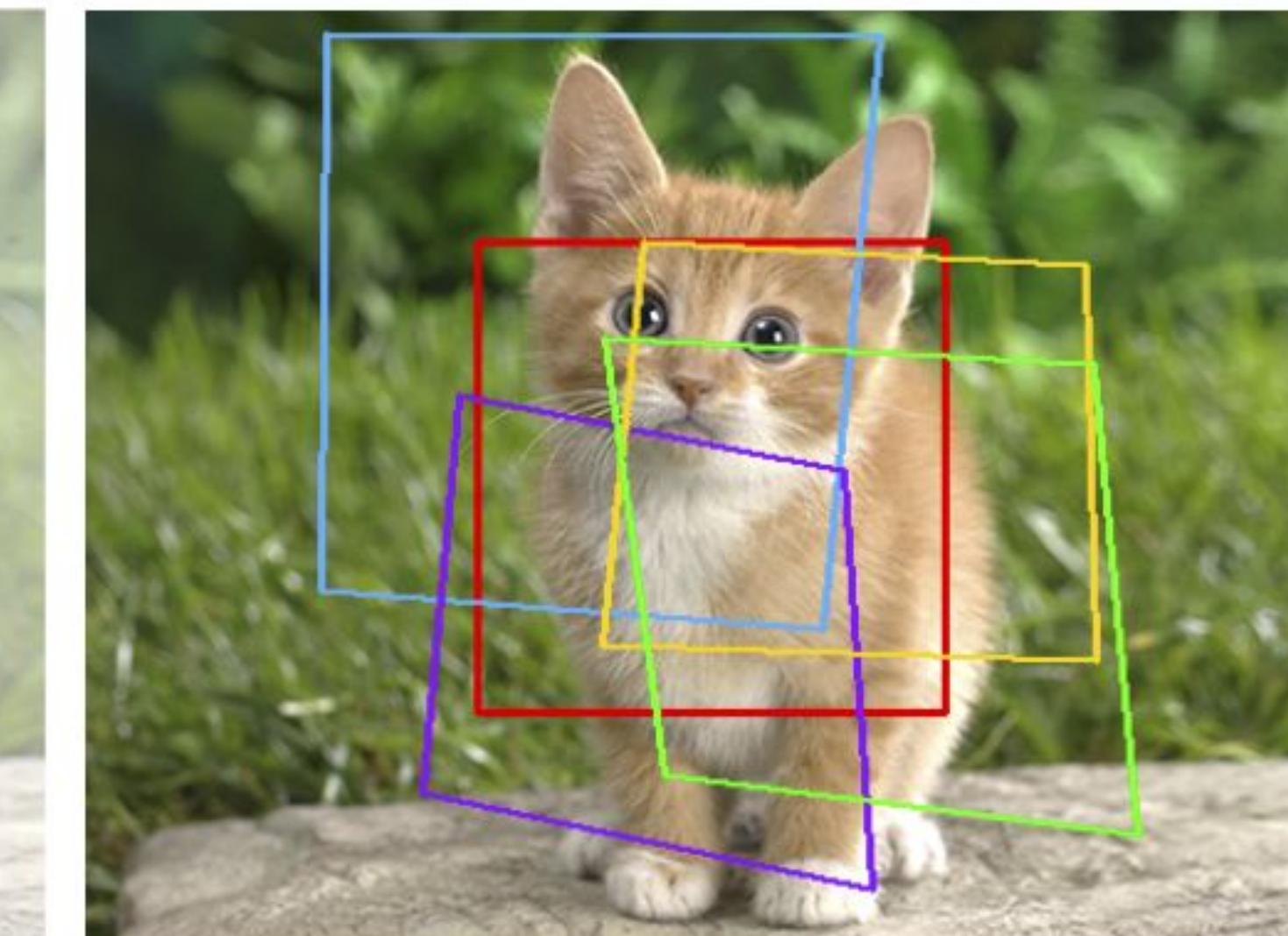
Пусть у нас есть пять патчей фотографии кота с неизвестным расположением



(a) image patches given for optimization

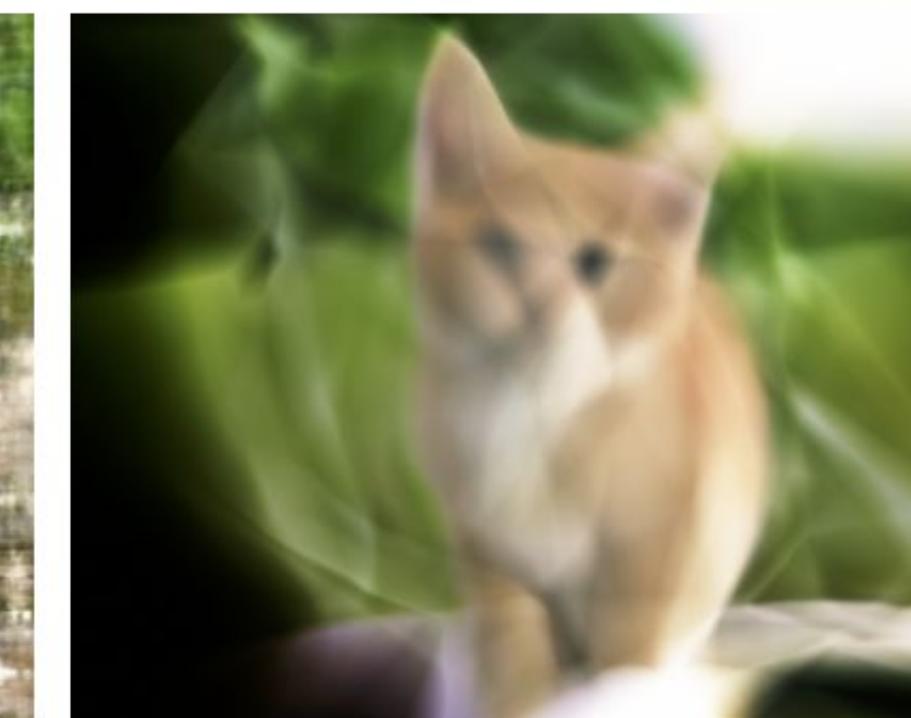
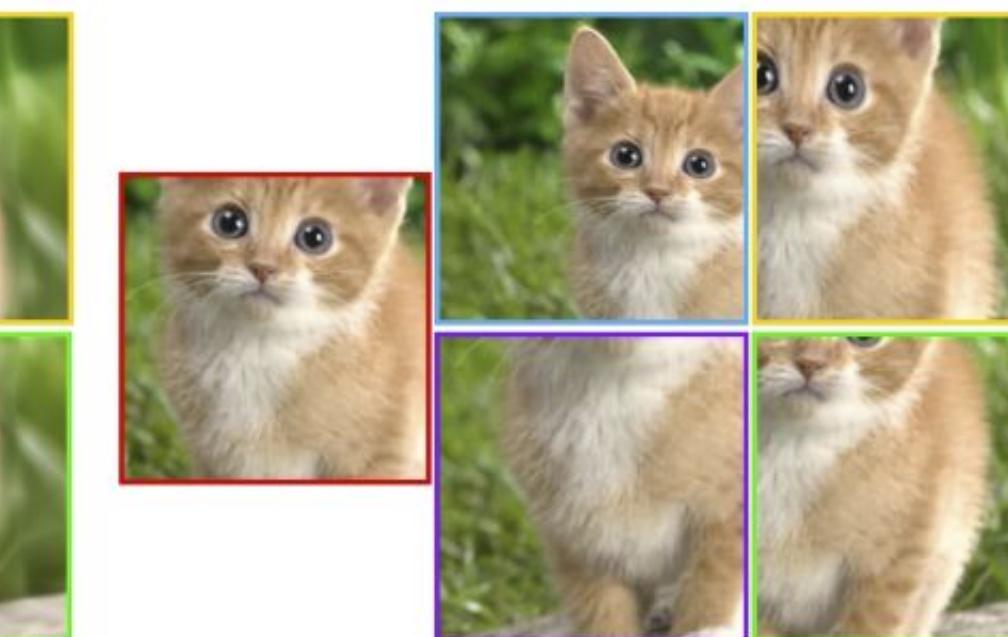
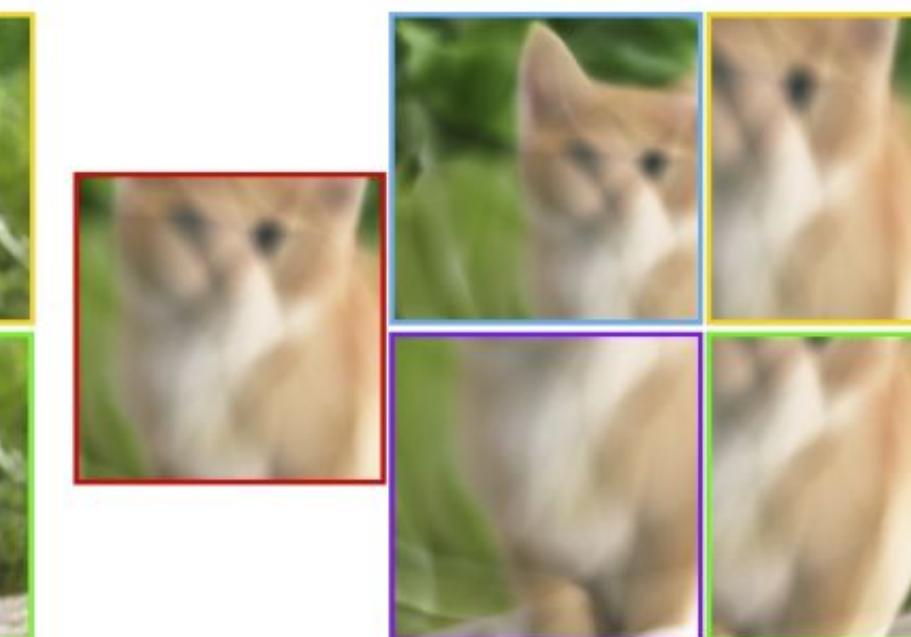
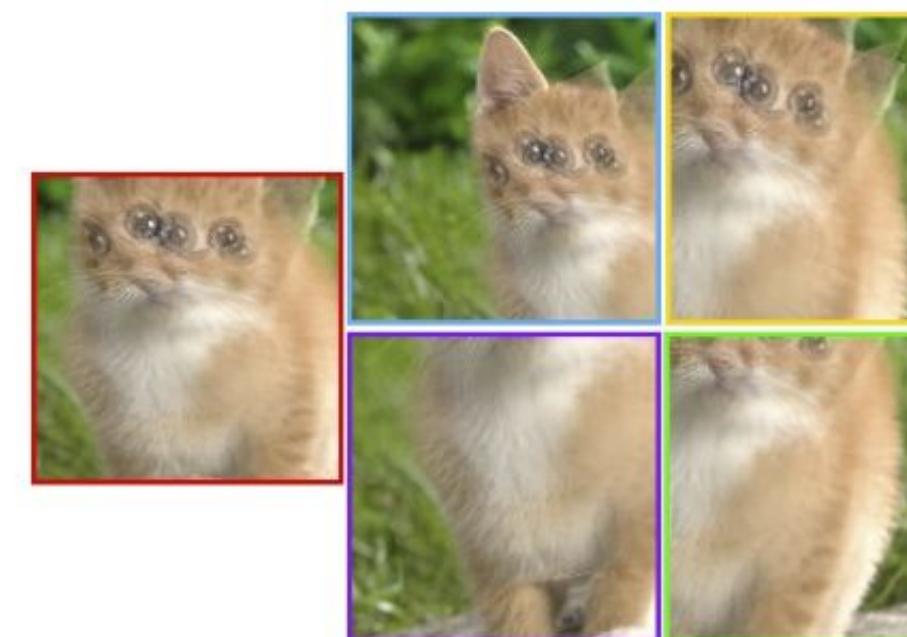
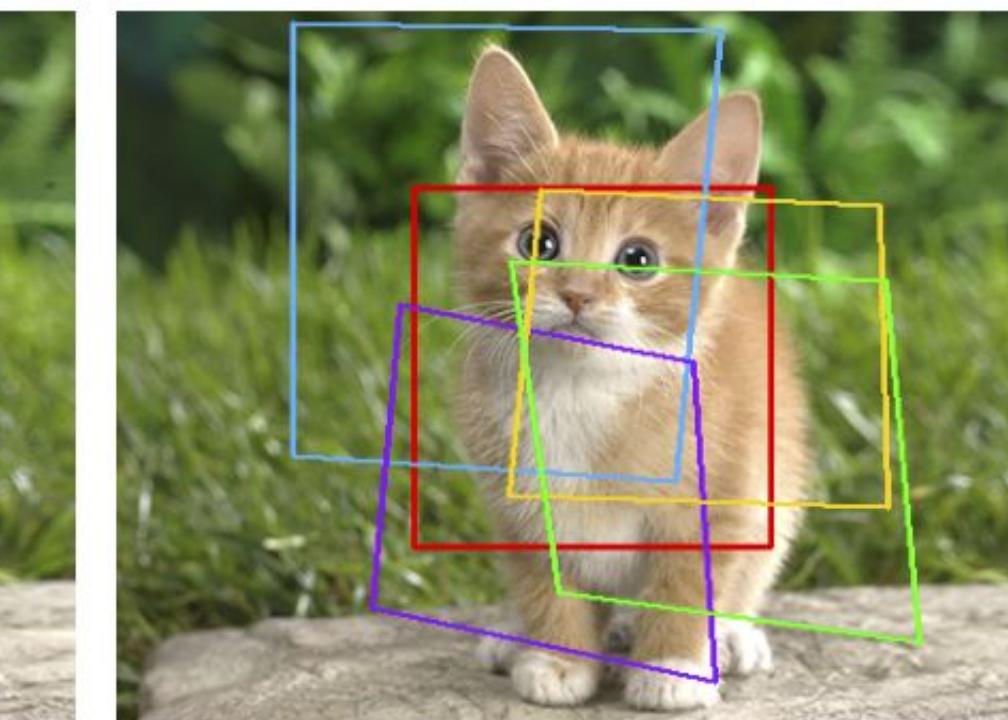
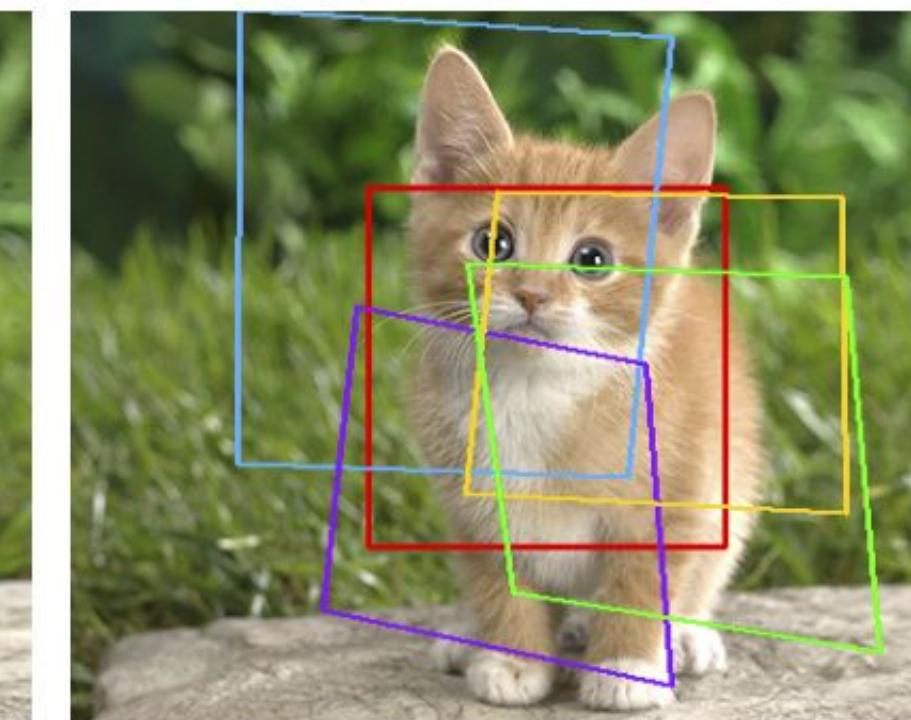
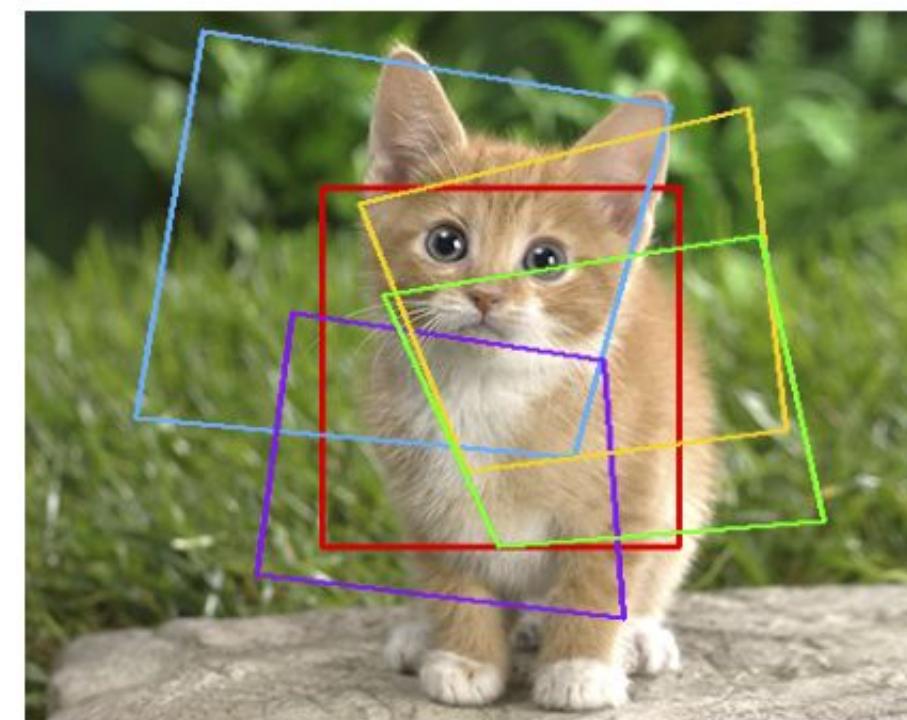


(b) initialization



(c) ground-truth warps

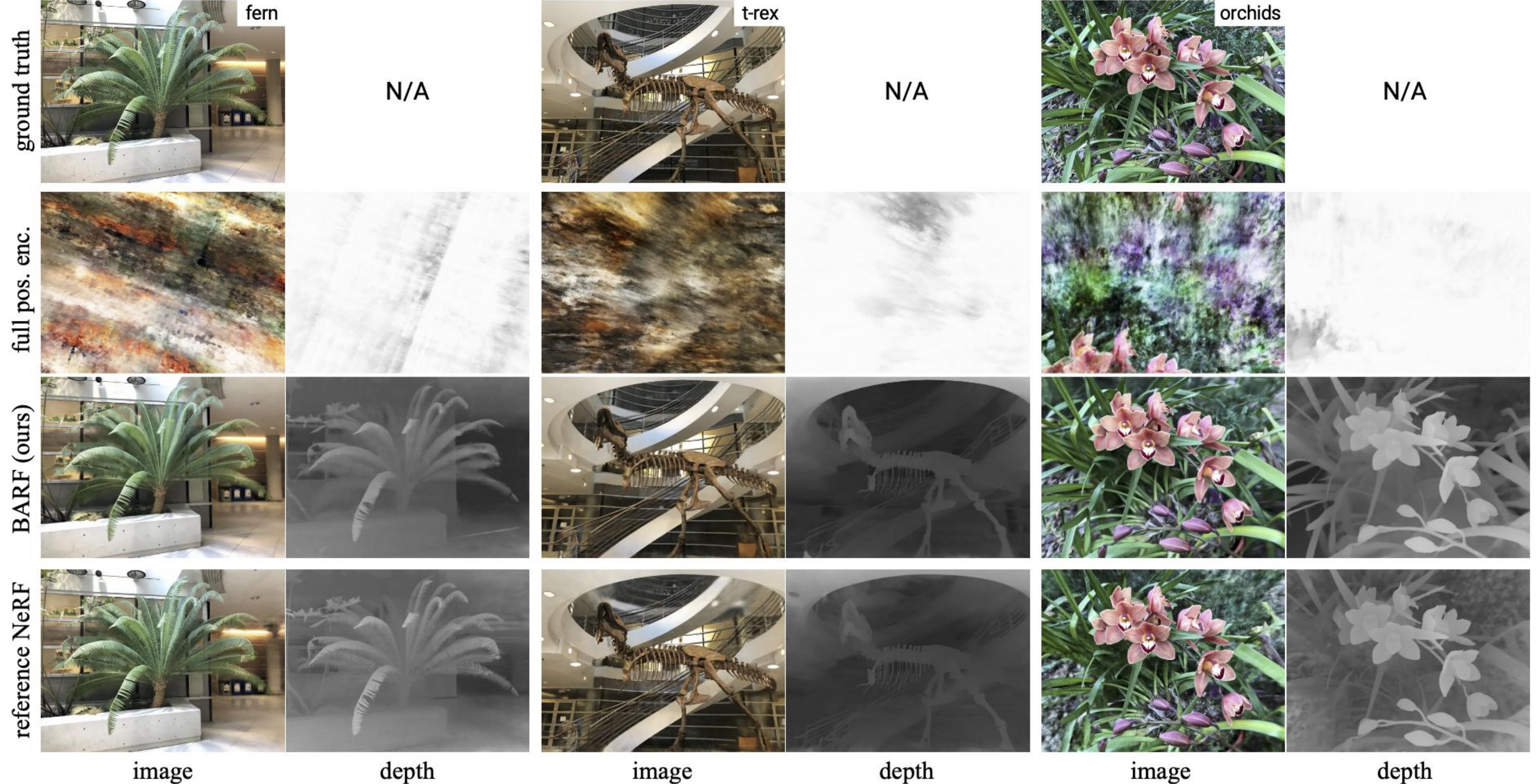
# 2D Примеры



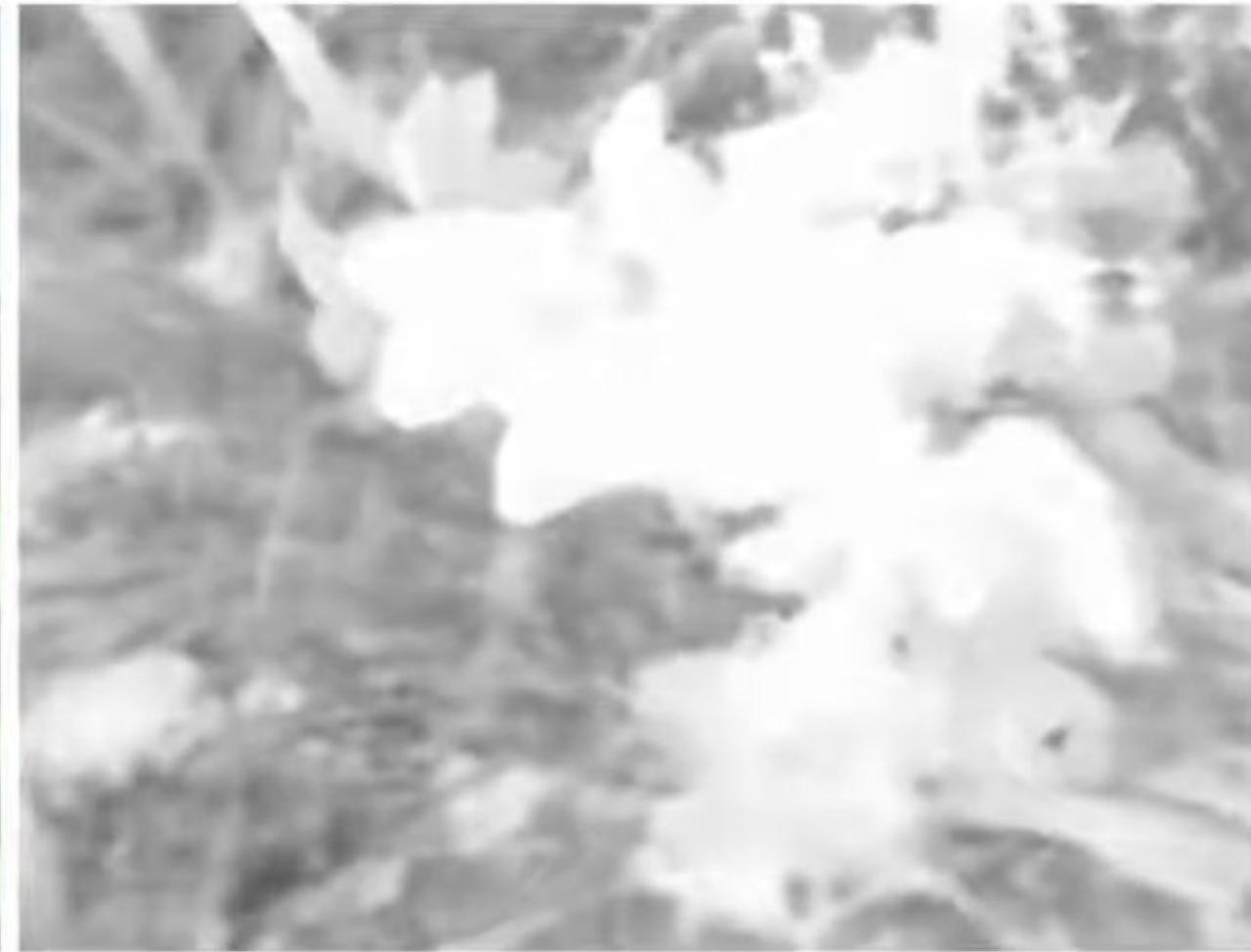
(a) naïve pos. enc.

(b) w/o pos. enc.

(c) BARF



w/ naive pos. enc.

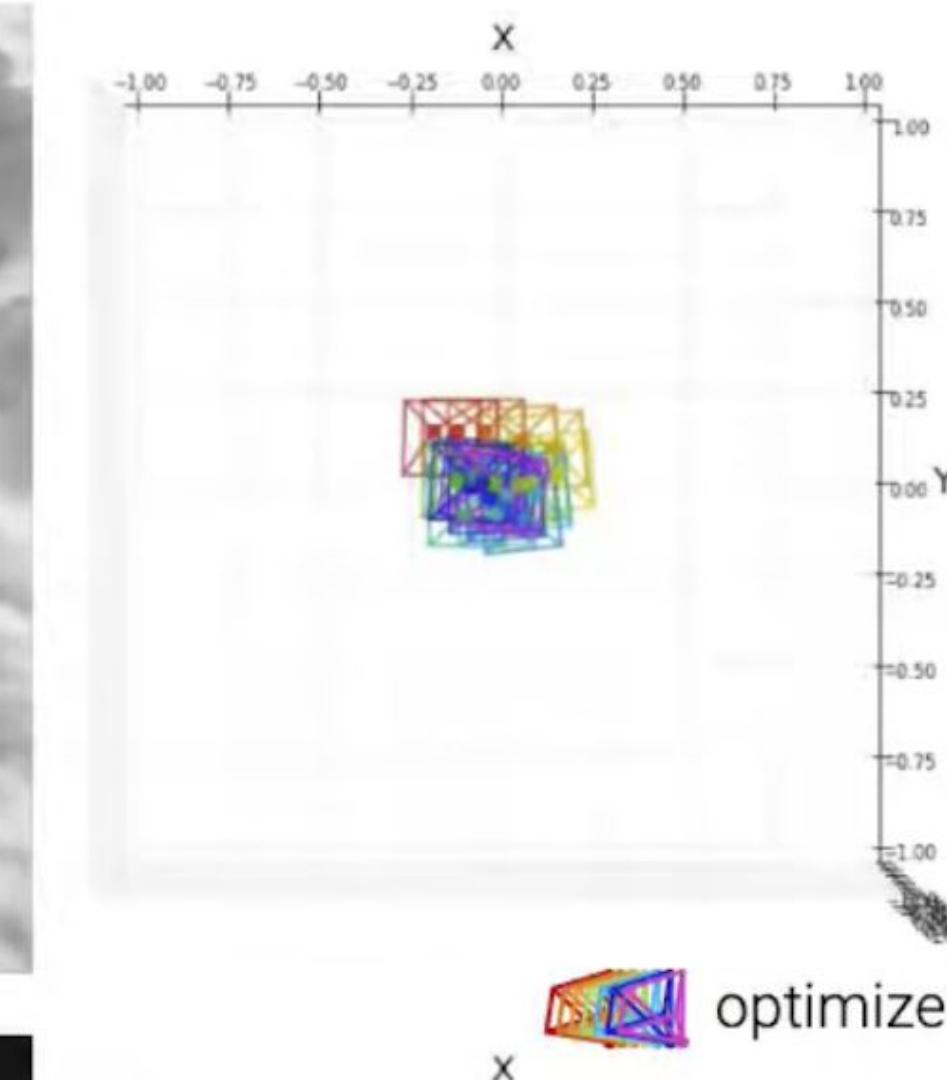


BARF (Ours)

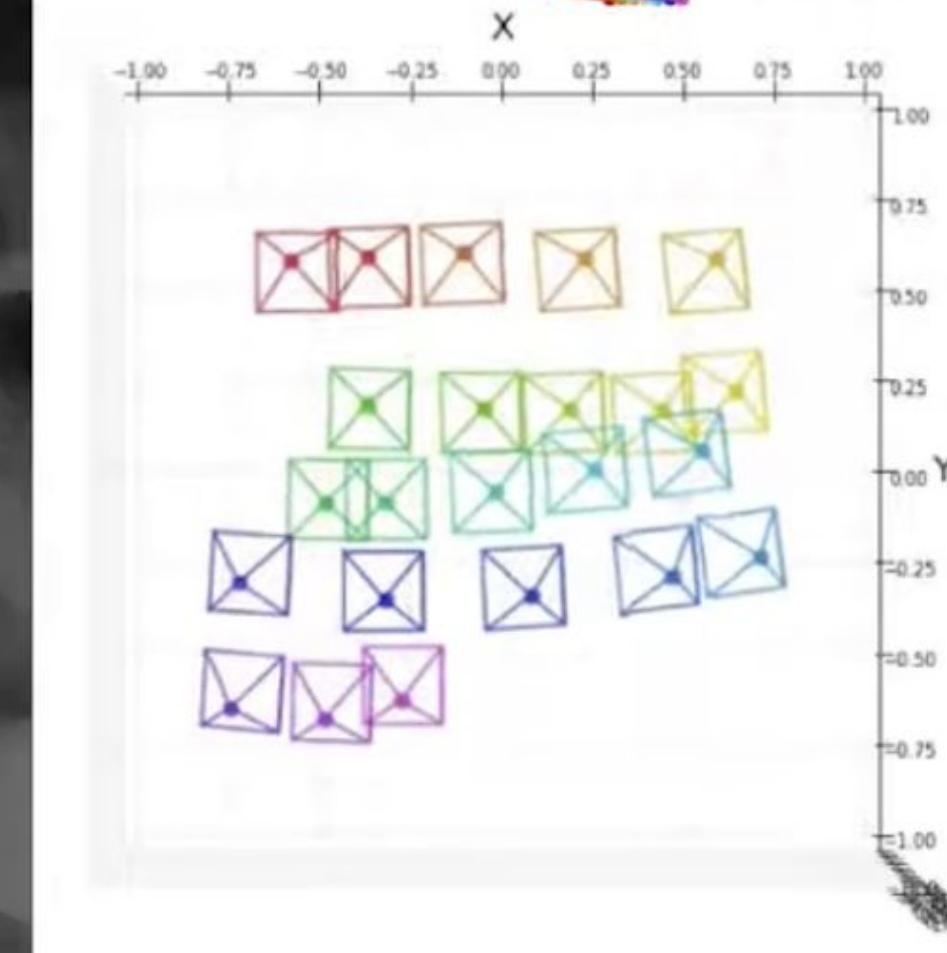


RGB

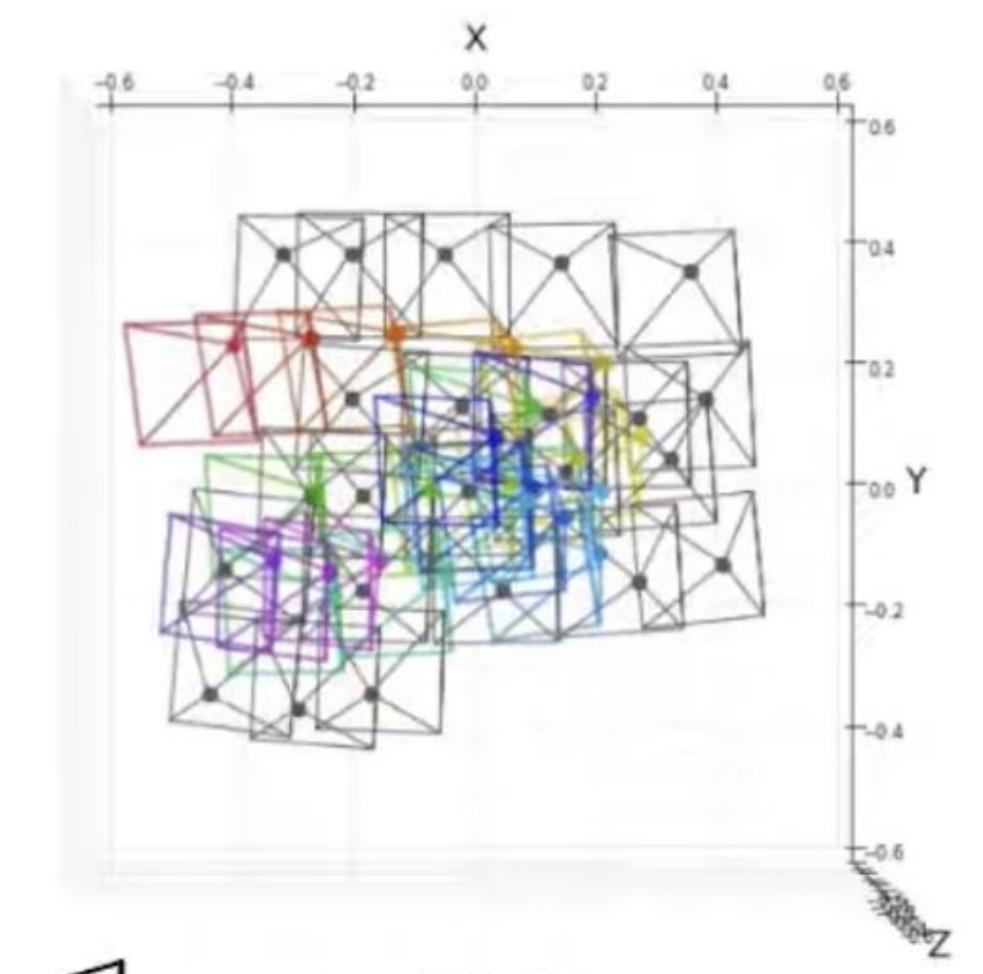
depth



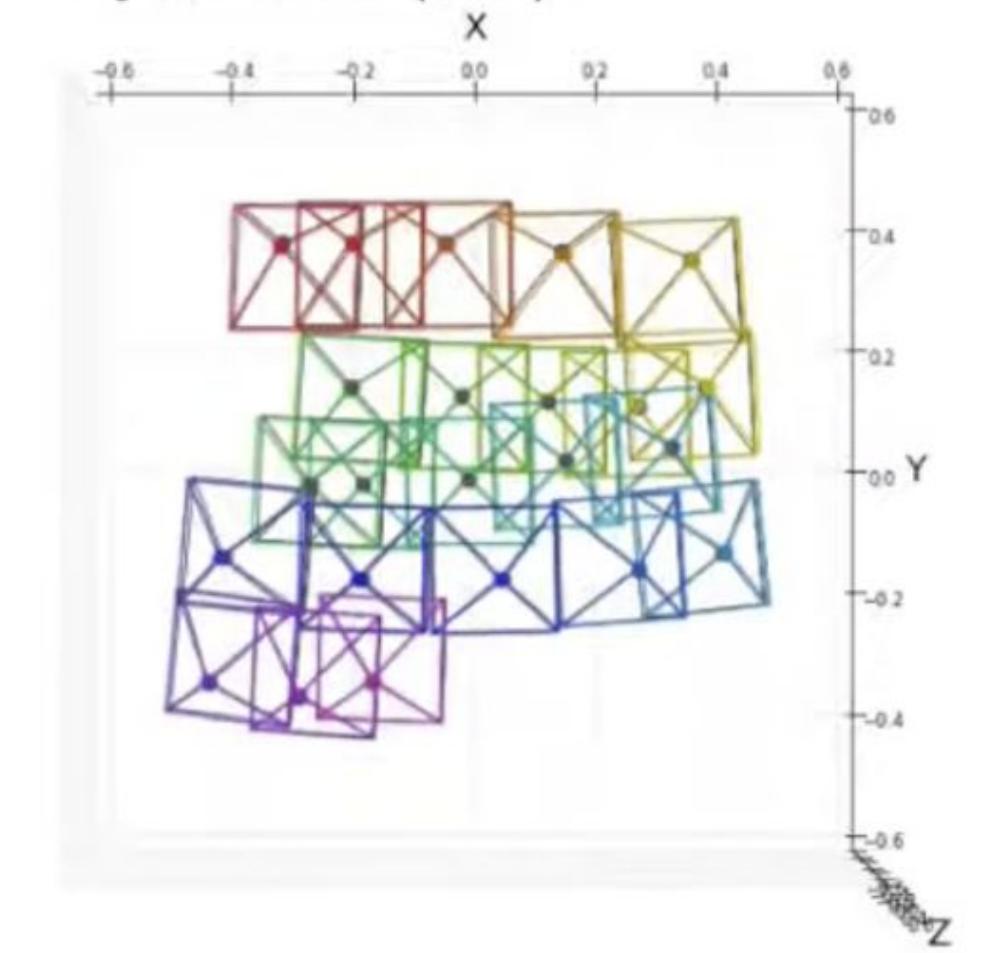
optimized



camera poses



COLMAP (SfM)



camera poses  
(aligned to COLMAP)

**Начальная инициализация расположений камер - просто в центре константная**

Scene	Camera pose registration				View synthesis quality								
	Rotation (°) ↓		Translation ↓		PSNR ↑			SSIM ↑			LPIPS ↓		
	full pos.enc.	BARF	full pos.enc.	BARF	full pos.enc.	BARF	ref. NeRF	full pos.enc.	BARF	ref. NeRF	full pos.enc.	BARF	ref. NeRF
Fern	74.452	<b>0.191</b>	30.167	<b>0.192</b>	9.81	<b>23.79</b>	23.72	0.187	<b>0.710</b>	0.733	0.853	<b>0.311</b>	0.262
Flower	2.525	<b>0.251</b>	2.635	<b>0.224</b>	17.08	<b>23.37</b>	23.24	0.344	<b>0.698</b>	0.668	0.490	<b>0.211</b>	0.244
Fortress	75.094	<b>0.479</b>	33.231	<b>0.364</b>	12.15	<b>29.08</b>	25.97	0.270	<b>0.823</b>	0.786	0.807	<b>0.132</b>	0.185
Horns	58.764	<b>0.304</b>	32.664	<b>0.222</b>	8.89	<b>22.78</b>	20.35	0.158	<b>0.727</b>	0.624	0.805	<b>0.298</b>	0.421
Leaves	88.091	<b>1.272</b>	13.540	<b>0.249</b>	9.64	<b>18.78</b>	15.33	0.067	<b>0.537</b>	0.306	0.782	<b>0.353</b>	0.526
Orchids	37.104	<b>0.627</b>	20.312	<b>0.404</b>	9.42	<b>19.45</b>	17.34	0.085	<b>0.574</b>	0.518	0.806	<b>0.291</b>	0.307
Room	173.811	<b>0.320</b>	66.922	<b>0.270</b>	10.78	<b>31.95</b>	32.42	0.278	<b>0.940</b>	0.948	0.871	<b>0.099</b>	0.080
T-rex	166.231	<b>1.138</b>	53.309	<b>0.720</b>	10.48	<b>22.55</b>	22.12	0.158	<b>0.767</b>	0.739	0.885	<b>0.206</b>	0.244
Mean	84.509	<b>0.573</b>	31.598	<b>0.331</b>	11.03	<b>23.97</b>	22.56	0.193	<b>0.722</b>	0.665	0.787	<b>0.238</b>	0.283

COLMAP  
posesCOLMAP  
posesCOLMAP  
poses

**PSNR** - оценивает уровень искажения изображения. Чем выше значение, тем больше деталей осталось

**SSIM** - измеряет сходство между двумя изображениями с точки зрения их структурной информации.

Учитывает изменения текстур, контраста и яркости и лучше соответствует восприятию качества человеком

**LPIPS** - использует глубокое обучение для оценки воспринимаемого качества изображений.

Учитывает сложные аспекты текстур, узоров и других особенностей,

которые могут быть упущены более простыми метриками, такими как PSNR и SSIM