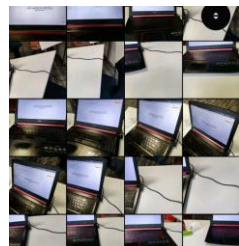


Допустим, вы захотели скачать NeRF и  
сделать нейронную реконструкцию своего  
рабочего места

# Как воспользоваться NeRFом самому

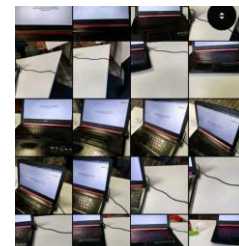
1. git clone <https://github.com/yenchenlin/nerf-pytorch.git>
2. Делаем много фотографий своего стола с разных ракурсов
3. Подаём фотографии и соответствующие позиции камер в NeRF



NeRF

# Как воспользоваться NeRFом самому

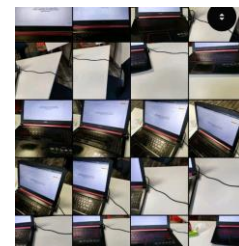
1. git clone <https://github.com/yenchenlin/nerf-pytorch.git>
2. Делаем много фотографий своего стола с разных ракурсов
3. Подаём фотографии и <sup>???</sup>соответствующие позиции камер в NeRF



NeRF

# Как воспользоваться NeRFом самому

1. git clone <https://github.com/yenchenlin/nerf-pytorch.git>
2. Делаем много фотографий своего стола с разных ракурсов
3. Подаём фотографии и соответствующие позиции камер в NeRF



NeRF



NeRF: Implementation Details:

“...we use ground truth camera poses, intrinsics, and bounds for synthetic data, and use the **COLMAP** structure-from-motion package to estimate these parameters for real data...”

# Доклад разбивается на 2 части:

1. COLMAP — реконструкция геометрии сцены **без нейронок**
2. BARF — как можно встроить реконструкцию позиций камер в NeRF

# COLMAP – Фотограмметрия / Structure from motion

20k фотографий Рима позволяют сделать такую реконструкцию **без ML**



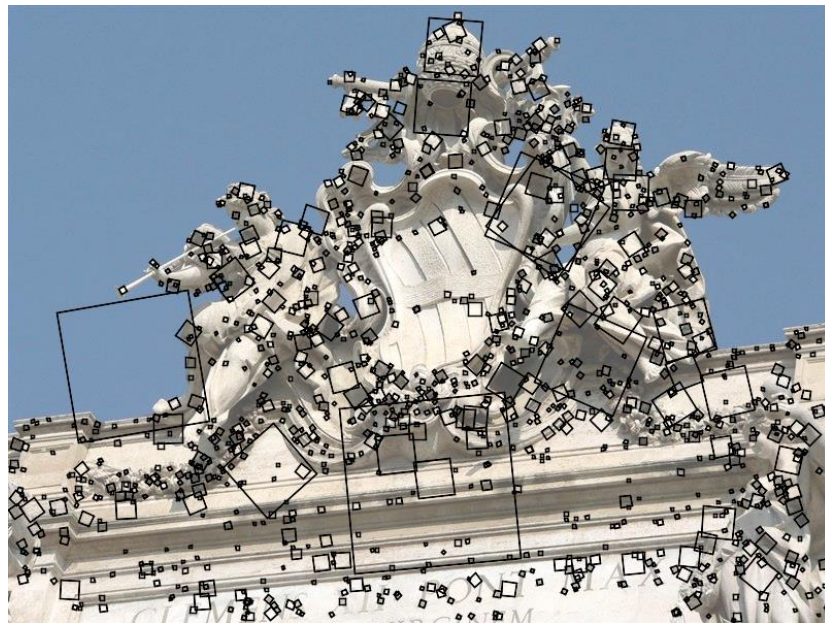
Вычисление этого занимает ~3 часа на суперкомпьютере с 256Gb RAM

Видны недостатки: много точек-выбросов, в некоторых участках сильно менее плотное облако точек, чем в других

# COLMAP

Реконструкция геометрии сцены из фотографий  
Без ML!

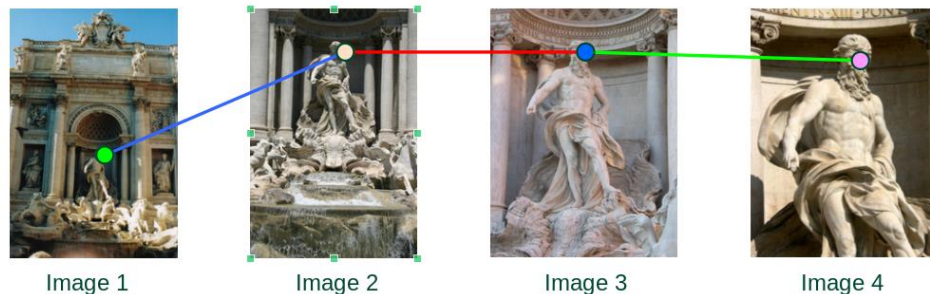
1. **Feature extraction**
2. Pairwise feature matching
3. Geometric verification
4. Bundle Adjustment



# COLMAP

Реконструкция геометрии сцены из фотографий  
Без ML!

1. Feature extraction
2. **Pairwise feature matching**
3. Geometric verification
4. Bundle Adjustment



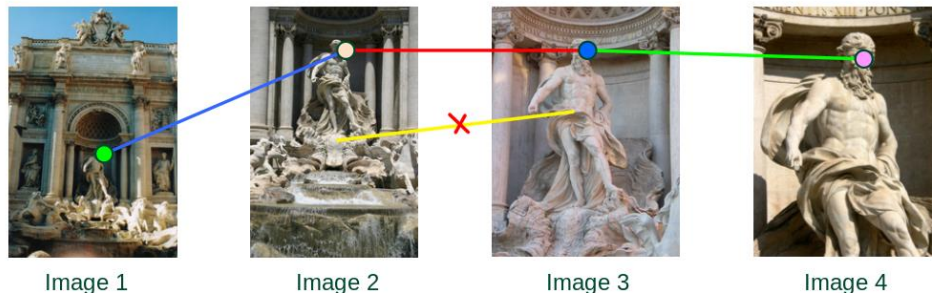
[from <http://phototour.cs.washington.edu/>]



# COLMAP

Реконструкция геометрии сцены из фотографий  
Без ML!

1. Feature extraction
2. Pairwise feature matching
3. **Geometric verification**
4. Bundle Adjustment

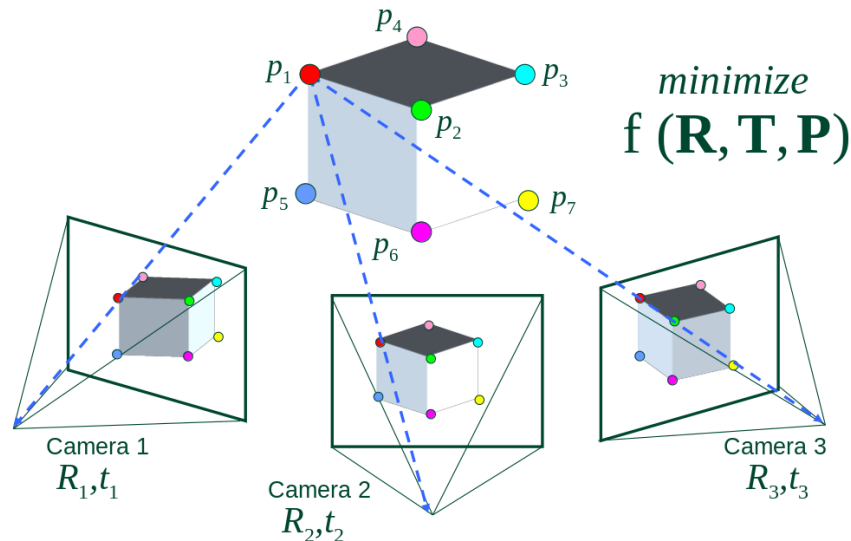


[from <http://phototour.cs.washington.edu/>]

# COLMAP

Реконструкция геометрии сцены из фотографий  
Без ML!

1. Feature extraction
2. Pairwise feature matching
3. Geometric verification
4. **Bundle Adjustment**

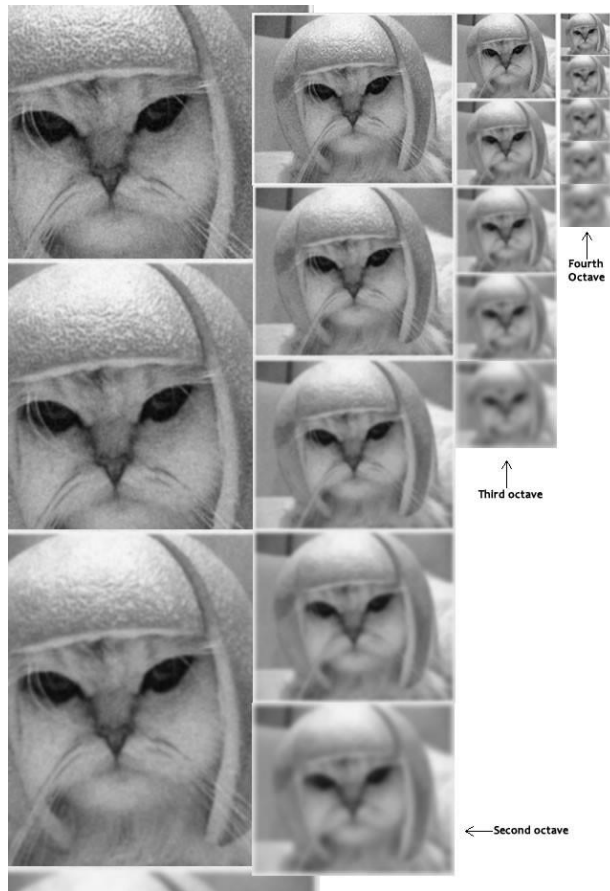


# 1. SIFT for Feature Extraction

## **SIFT** - Scale Invariant Feature Transform

Алгоритм для извлечения ключевых точек из изображений и определения их дескрипторов, которые не зависят от базовых преобразований:  
масштабирование, поворот, сдвиг цвета.

1. **Peak Selection**
2. Keypoint Localization
3. Orientation Assignment
4. Keypoint descriptor



# 1. SIFT for Feature Extraction

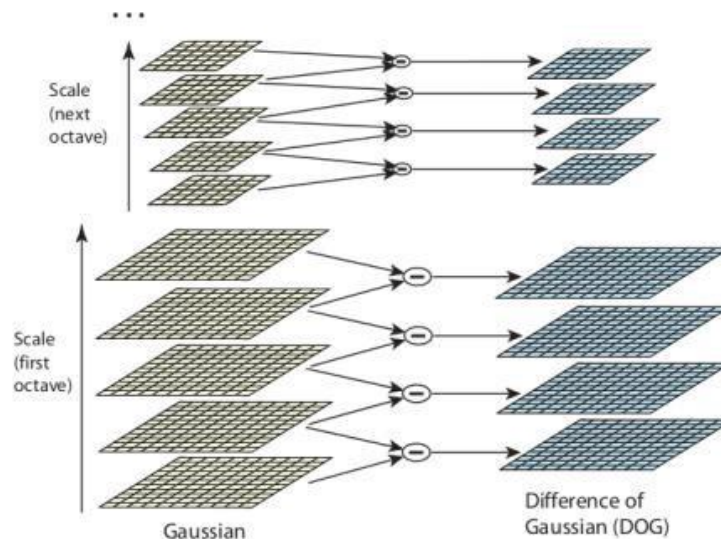
## SIFT - Scale Invariant Feature Transform

Алгоритм для извлечения ключевых точек из изображений и определения их дескрипторов, которые не зависят от базовых преобразований:  
масштабирование, поворот, сдвиг цвета.



### 1. Peak Selection

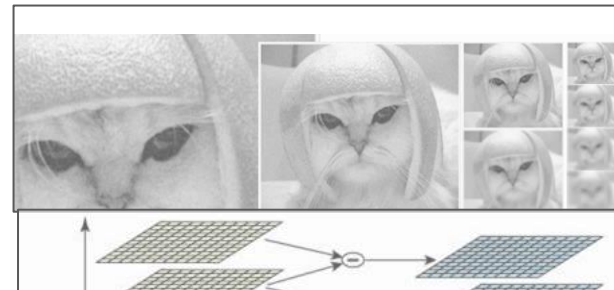
2. Keypoint Localization
3. Orientation Assignment
4. Keypoint descriptor



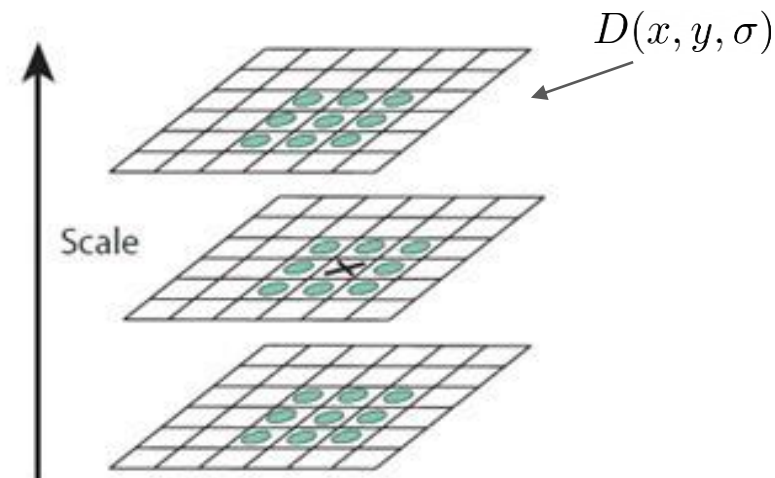
# 1. SIFT for Feature Extraction

## **SIFT** - Scale Invariant Feature Transform

Алгоритм для извлечения ключевых точек из изображений и определения их дескрипторов, которые не зависят от базовых преобразований: масштабирование, поворот, сдвиг цвета.



1. **Peak Selection**
2. Keypoint Localization
3. Orientation Assignment
4. Keypoint descriptor

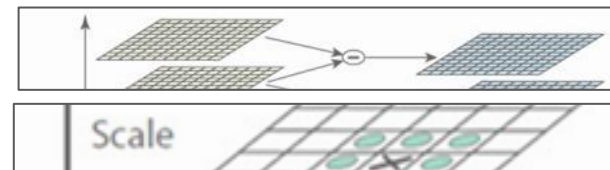


# 1. SIFT for Feature Extraction

## **SIFT** - Scale Invariant Feature Transform

Алгоритм для извлечения ключевых точек из изображений и определения их дескрипторов, которые не зависят от базовых преобразований:  
масштабирование, поворот, сдвиг цвета.

1. **Peak Selection**
2. Keypoint Localization
3. Orientation Assignment
4. Keypoint descriptor



# 1. SIFT for Feature Extraction

## SIFT - Scale Invariant Feature Transform

Алгоритм для извлечения ключевых точек из изображений и определения их дескрипторов, которые не зависят от базовых преобразований:  
масштабирование, поворот, сдвиг цвета.

1. Peak Selection
2. **Keypoint Localization**
3. Orientation Assignment
4. Keypoint descriptor

Координаты точек могли быть найдены с погрешностью.

Чтобы улучшить локализацию точек, приблизим окрестность  $D(P)$  к квадратичной функции рядом Тейлора и найдём экстремум

$P$  – sample point

$$D(P + u) \approx D(P) + (\nabla D)^T u + \frac{1}{2} u^T \nabla^2 D u$$

$$\hat{u} = -(\nabla^2 D(P))^{-1} \nabla D(P)$$

$$\hat{P} = P + \hat{u} \quad - \text{localized sample point}$$

# 1. SIFT for Feature Extraction

## SIFT - Scale Invariant Feature Transform

Алгоритм для извлечения ключевых точек из изображений и определения их дескрипторов, которые не зависят от базовых преобразований: масштабирование, поворот, сдвиг цвета.

1. Peak Selection
2. **Keypoint Localization**
3. Orientation Assignment
4. Keypoint descriptor

Reject flat surfaces:

$$|D(\hat{P})| < 0.03$$

■ Reject edges:

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$$

Let  $\alpha$  be the eigenvalue with larger magnitude and  $\beta$  the smaller.

$$\text{Tr}(\mathbf{H}) = D_{xx} + D_{yy} = \alpha + \beta,$$

$$\text{Det}(\mathbf{H}) = D_{xx}D_{yy} - (D_{xy})^2 = \alpha\beta.$$

Let  $r = \alpha/\beta$ .  
So  $\alpha = r\beta$

$$\frac{\text{Tr}(\mathbf{H})^2}{\text{Det}(\mathbf{H})} = \frac{(\alpha + \beta)^2}{\alpha\beta} = \frac{(r\beta + \beta)^2}{r\beta^2} = \frac{(r + 1)^2}{r},$$

$(r+1)^2/r$  is at a min when the 2 eigenvalues are equal.

■  $r < 10$



# 1. SIFT for Feature Extraction

## **SIFT** - Scale Invariant Feature Transform

Алгоритм для извлечения ключевых точек из изображений и определения их дескрипторов, которые не зависят от базовых преобразований:  
масштабирование, поворот, сдвиг цвета.

1. Peak Selection
2. **Keypoint Localization**
3. Orientation Assignment
4. Keypoint descriptor



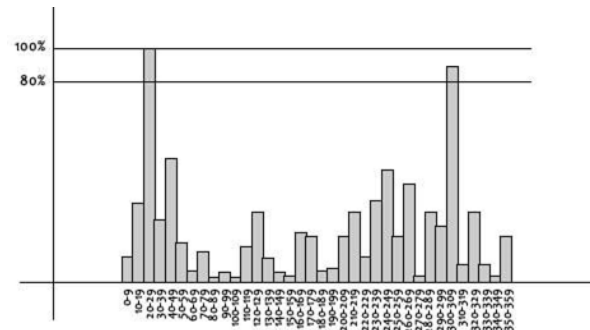
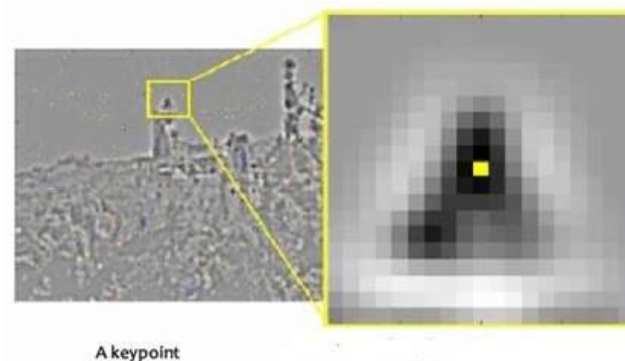
# 1. SIFT for Feature Extraction

## SIFT - Scale Invariant Feature Transform

Алгоритм для извлечения ключевых точек из изображений и определения их дескрипторов, которые не зависят от базовых преобразований: масштабирование, поворот, сдвиг цвета.

1. Peak Selection
2. Keypoint Localization
3. **Orientation Assignment**
4. Keypoint descriptor

Считаем все градиенты в окрестности точки



# 1. SIFT for Feature Extraction

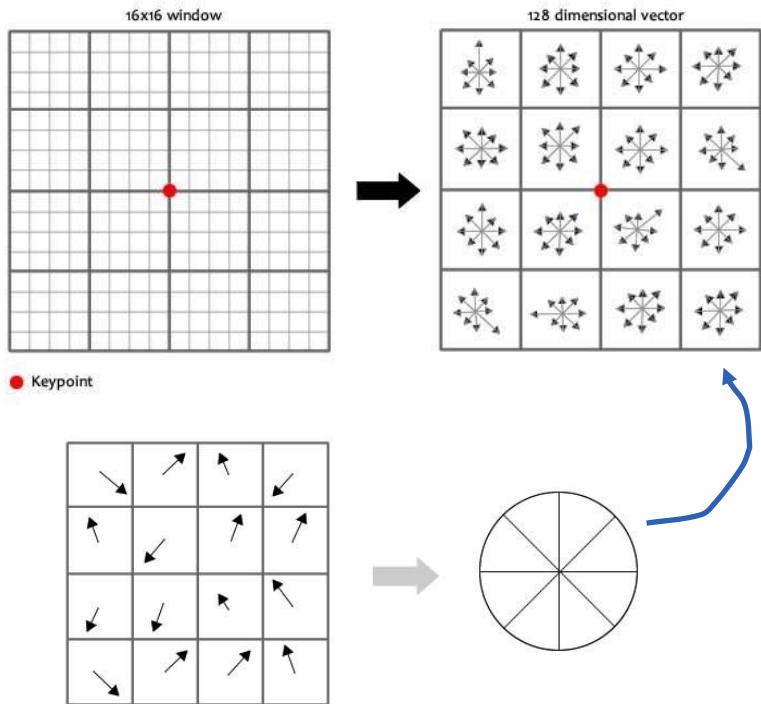
## SIFT - Scale Invariant Feature Transform

Алгоритм для извлечения ключевых точек из изображений и определения их дескрипторов, которые не зависят от базовых преобразований:  
масштабирование, поворот, сдвиг цвета.

1. Peak Selection
2. Keypoint Localization
3. Orientation Assignment
4. **Keypoint descriptor**

$\mathcal{I} = \{I_i | i = 1 \dots N_I\}$  - images

$\mathcal{F}_i = \{(P_i, f_i) | i = 1 \dots N_{F_i}\}$  - features

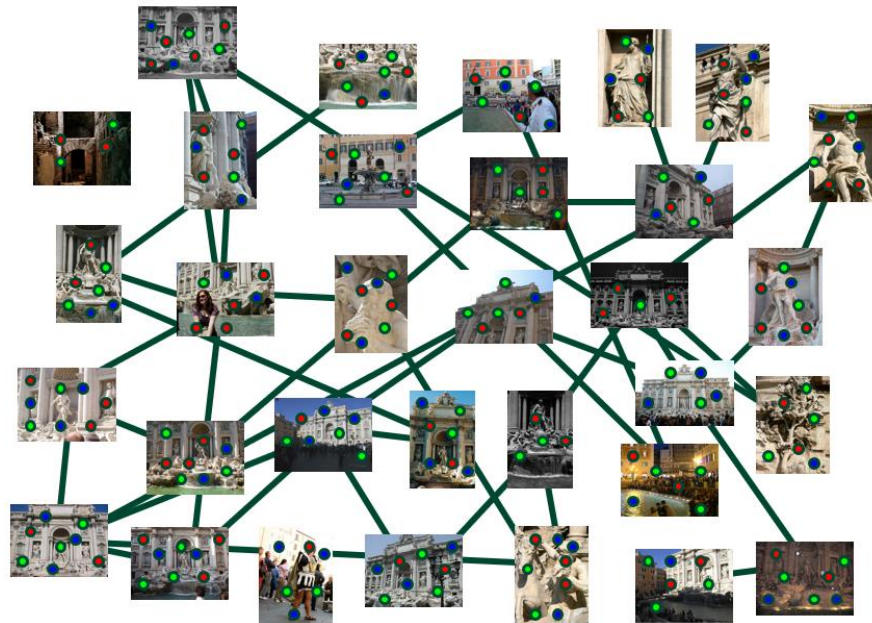


## 2. Pairwise feature matching

После получения точек и их дескрипторов, ищем близкие дескрипторы между картинками  $|f_i - f_j| < c$

Однако наивное решение этой задачи имеет сложность  $O(N_I^2 N_{F_i}^2)$  которая неприемлема при работе с 5к картинок

Один из подходов — строить kd-дерево для дескрипторов точек каждой картинки и для каждой пары картинок  $I, J$  делать поиск совпадений, ограничивая каждый запрос проходом по 200 бакетам.



### 3. Geometric verification

Поскольку дескрипторы опираются только на вид картинки в определённой окрестности, возможны коллизии — разные элементы сцены будут иметь схожий вид, соответственно и схожие дескрипторы SIFT.

Т.е. Pairwise feature matching мог дать лишние пары

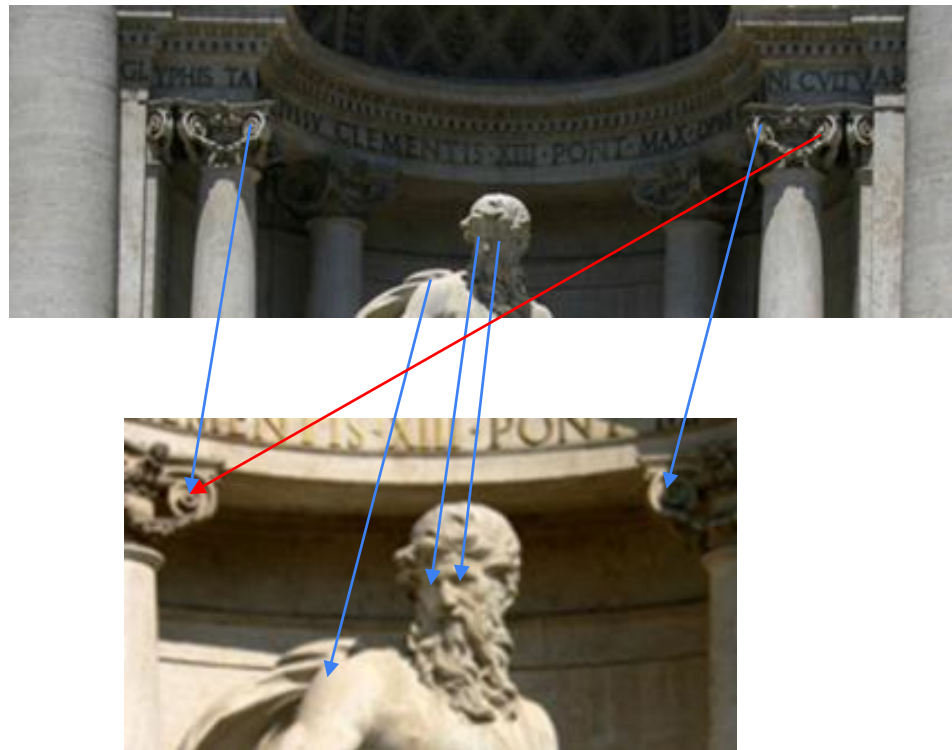


[from <http://phototour.cs.washington.edu/>]

### 3. Geometric verification

Для двух картинок  $I$ ,  $J$  с большим количеством совпадений, пытаемся посчитать проектирующее преобразование  $\mathcal{H} : \{P_{I,1}, \dots\} \mapsto \{P_{J,1}, \dots\}$

точек картинки  $I$  в точки картинки  $J$ , сохраняя совпадающие дескрипторами точки.



[from <http://phototour.cs.washington.edu/>]

### 3. Geometric verification

Для двух картинок  $I$ ,  $J$  с большим количеством совпадений, пытаемся посчитать проектирующее преобразование  $\mathcal{H} : \{P_{I,1}, \dots\} \mapsto \{P_{J,1}, \dots\}$

точек картинки  $I$  в точки картинки  $J$ , сохраняя совпадающие дескрипторами точки.

Используем алгоритм RANSAC для избавления от выбросов/шума:

1. N раз:

I) Посчитаем  $\mathcal{H}$  для случайного подмножества точек из пары картинок

II) Оцениваем качество на всех точках

2. Берём  $\mathcal{H}$  с наилучшей общей точностью.

3. Удаляем все пары точек на которых  $\mathcal{H}$  даёт слишком большую погрешность



[from <http://phototour.cs.washington.edu/>]

## 4. Bundle Adjustment

 $X_k$ 

- Позиции точек в 3D пространстве (неизвестный параметр)

 $P_c$ 

- Параметры камер (неизвестный параметр)

 $\pi(P_c, X_k)$ 

- Координаты отображения точки  $X_k$  на изображение камеры

 $x_j$ 

- Координаты точки в 2D пространстве для какой-то камеры

Функционал соответствия параметров друг другу:

$$L(X, P, x) = \sum_c^{N_I} \sum_i^{N_{F_i}} \rho \left( \left\| \pi(P_c, X_{k_{c,i}}) - x_{c,i} \right\|_2^2 \right)$$



## 4. Bundle Adjustment

$$L(X, P, x) = \sum_c^{N_I} \sum_i^{N_{F_i}} \rho \left( \left\| \underbrace{\pi(P_c, X_{k_{c,i}})}_{\text{Проекция соответствующей 3D точки на камеру}} - \underbrace{x_{c,i}}_{\text{Известные правильные координаты точки в пространстве 2D-картинок}} \right\|_2^2 \right)$$

Суммирование  
по картинкам

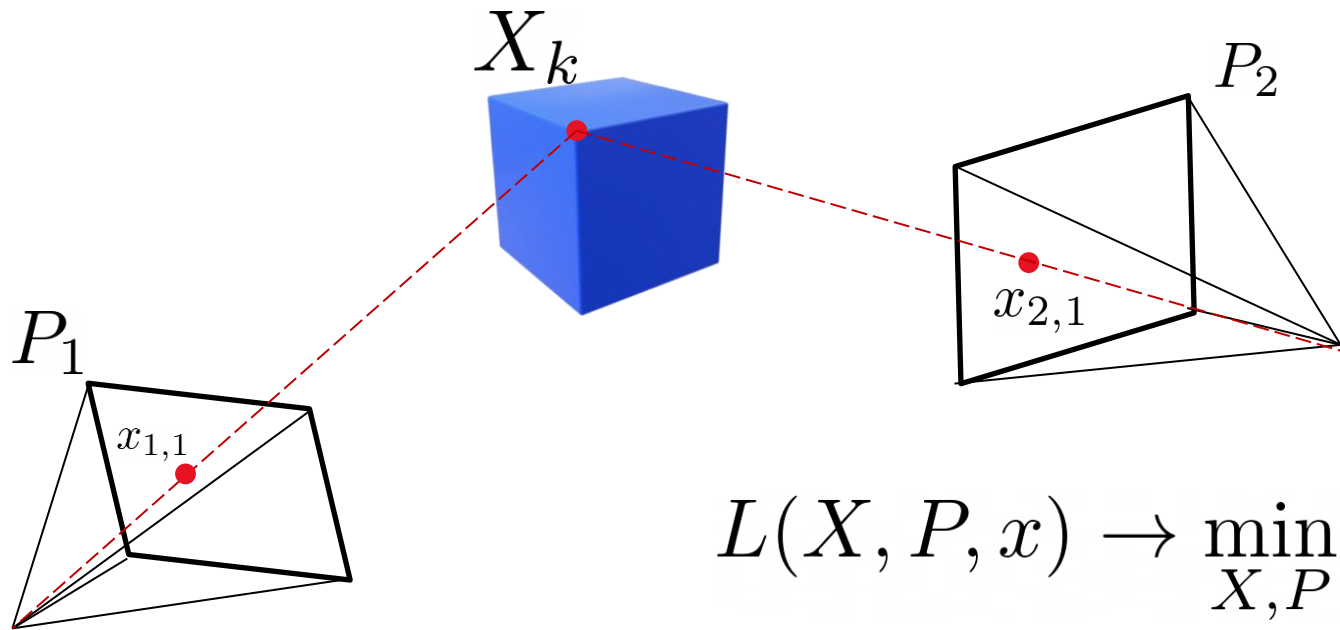
Суммирование  
по точкам с  
картинки

Проекция  
соответствующей 3D  
точки на камеру

Известные правильные  
координаты точки в  
пространстве 2D-картинок

$\rho$  - функция, снижающая значения для потенциальных выбросов/шума

## 4. Bundle Adjustment



## 4. Bundle Adjustment

1. Добавляем в набор рассматриваемых изображений  $I_1, I_2$  с большим количеством совпадающих точек (скорее всего, фотографии, сделанные с небольшим сдвигом)
2.  $L(X, P, x) \rightarrow \min$   
( $X, P, x$  взятые из набора обрабатываемых изображений)
3. Добавляем в набор  $I_t$  с наибольшим пересечением в множестве точек с имеющимся набором. Если же все изображения обработаны, Bundle Adjustment завершает работу



+



[from <http://phototour.cs.washington.edu/>]

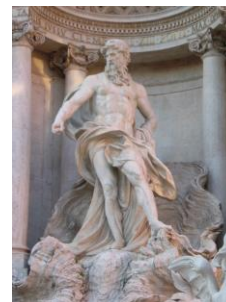
## 4. Bundle Adjustment

1. Добавляем в набор рассматриваемых изображений  $I_1, I_2$  с большим количеством совпадающих точек (скорее всего, фотографии, сделанные с небольшим сдвигом)



2.  $L(X, P, x) \rightarrow \min$   
( $X, P, x$  взятые из набора обрабатываемых изображений)

+



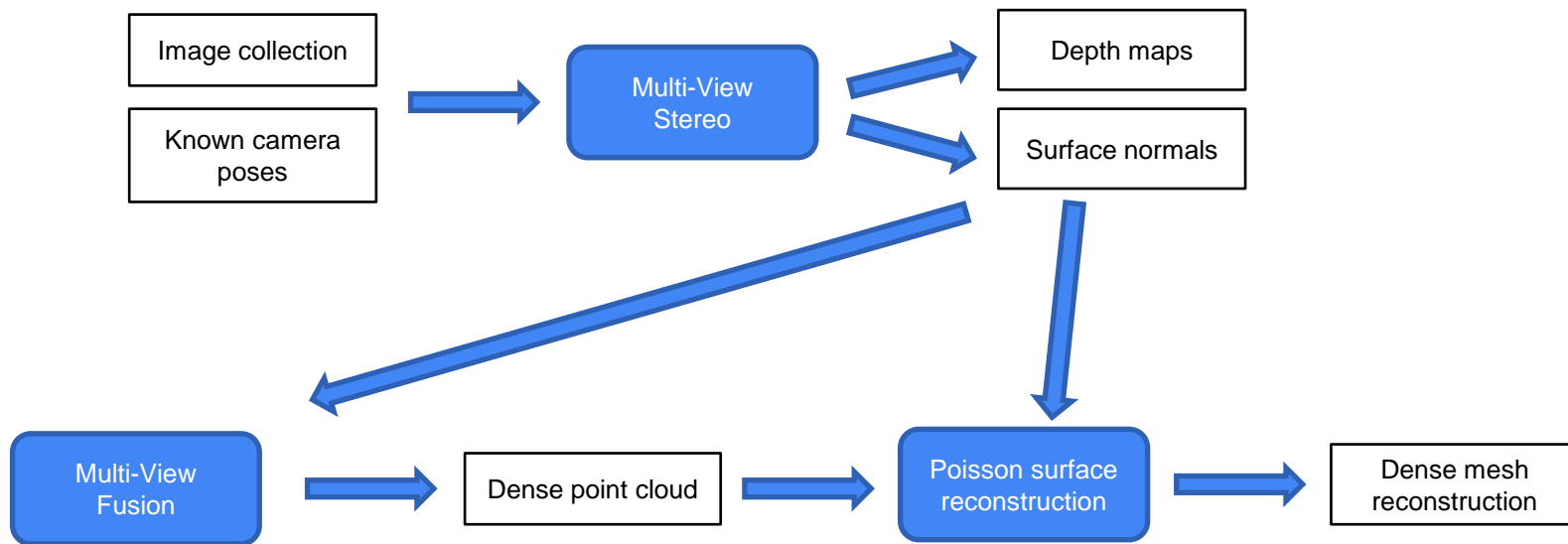
[from <http://phototour.cs.washington.edu/>]

3. Добавляем в набор  $I_t$  с наибольшим пересечением в множестве точек с имеющимся набором. Если же все изображения обработаны, Bundle Adjustment завершает работу

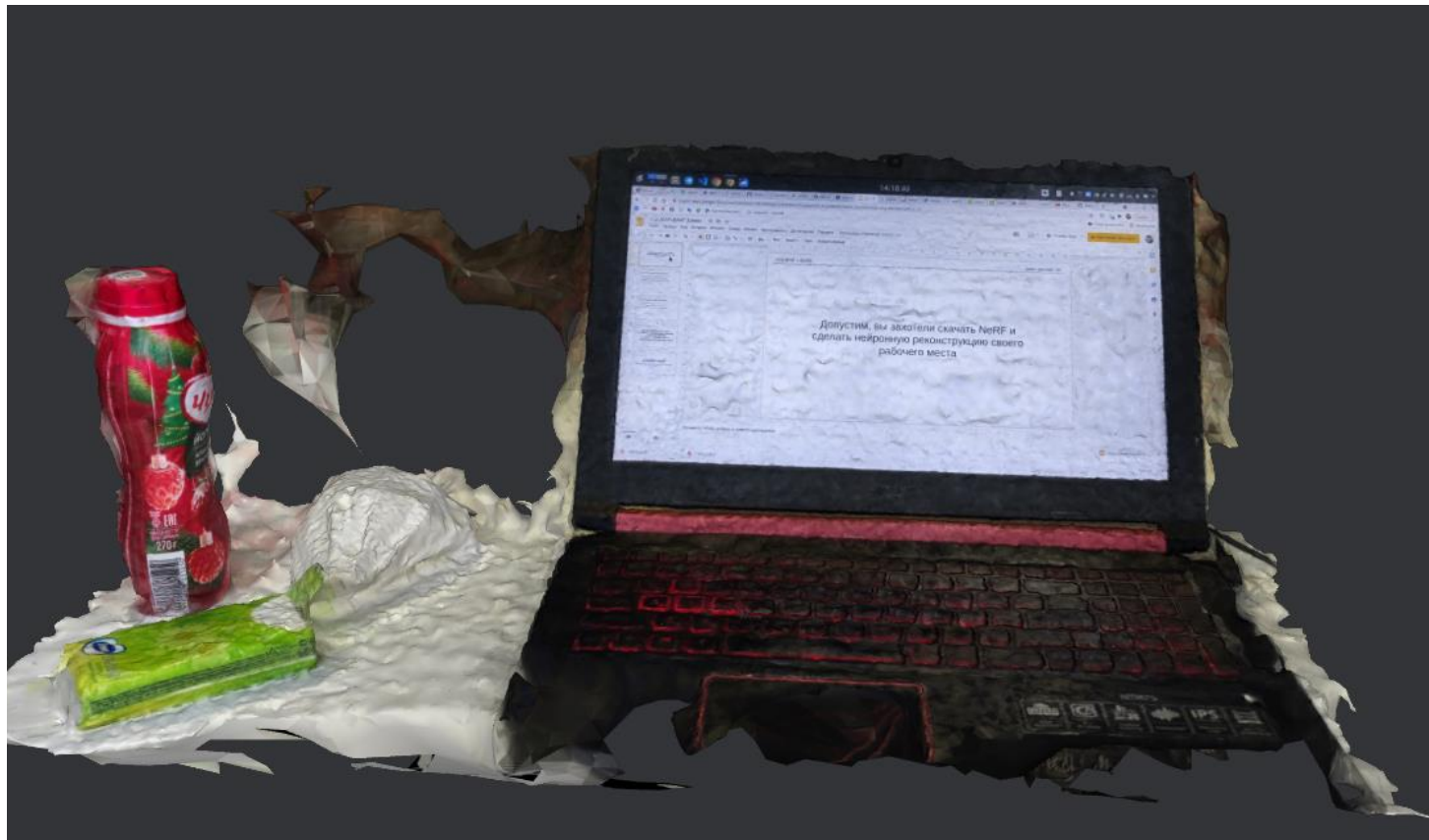
Наиболее популярен для оптимизации функционала ВА алгоритм Левенберга — Марквардта, в подробности которого я вдаваться не буду

# Mesh building

COLMAP также позволяет построить Mesh после получения позиций камер

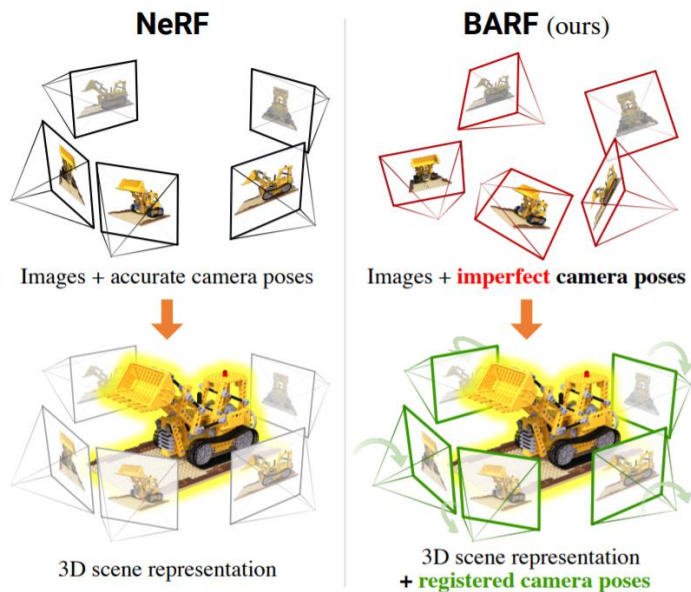


Если хотите так же (или лучше), попробуйте Meshroom



# BARF – Bundle-Adjusting NeRF

Как добавить реконструкцию позиций камер в NeRF и забыть про COLMAP раз и навсегда



# BARF – Bundle-Adjusting NeRF

Нельзя просто сказать что  $P_c$  - теперь тоже параметры NeRF'а, которые будем оптимизировать через Gradient Descent?

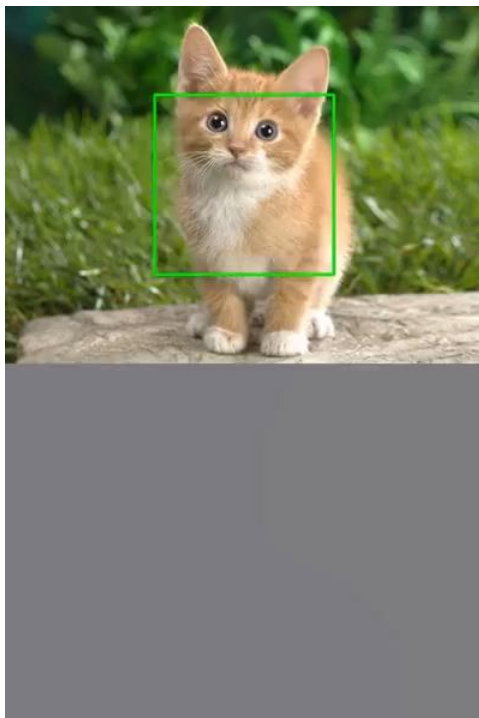
Рассмотрим на примере NeRFa в 2D: возьмём картинку котика и возьмём случайные куски. Положения этих кусков алгоритму известны не будут



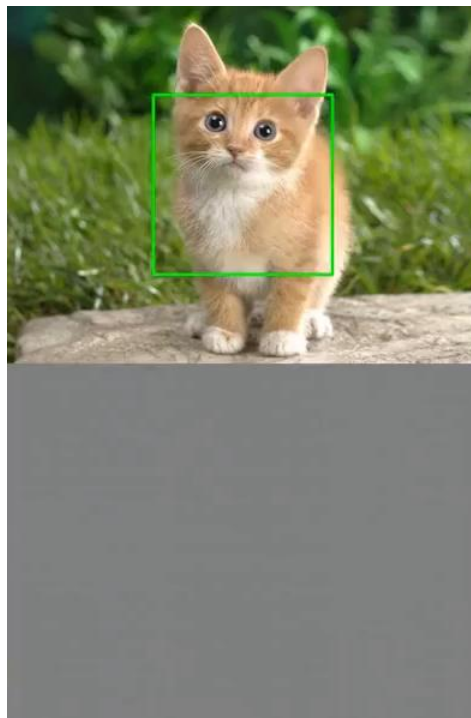
[from <https://chenhsuanlin.bitbucket.io/bundle-adjusting-NeRF/>]



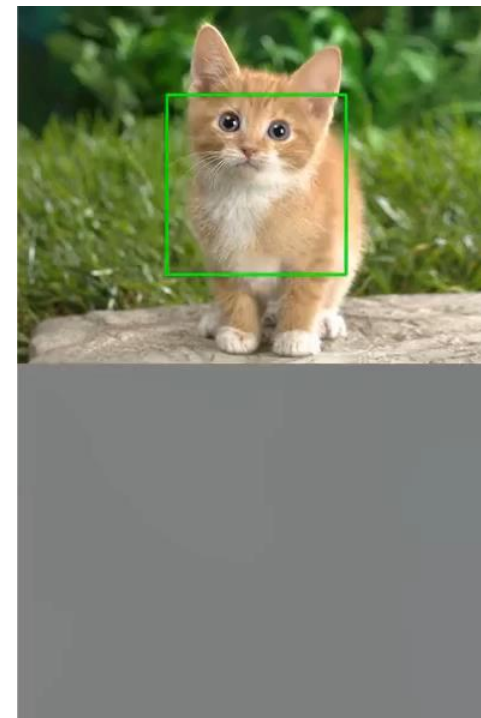
# BARF – Bundle-Adjusting NeRF



w/o positional encoding



with positional encoding

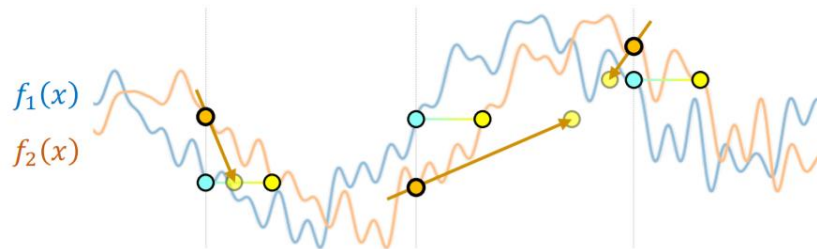


BARF

# BARF – Bundle-Adjusting NeRF

Поскольку обратное распространение ошибки в параметры камеры проходит через positional encodings, градиент ошибки по отношению к параметрам камеры может быть не очень гладким.

Тогда может быть сложно искать оптимальные параметры камеры, т.к. на значение градиента сильно влияют высокочастотные positional encodings.

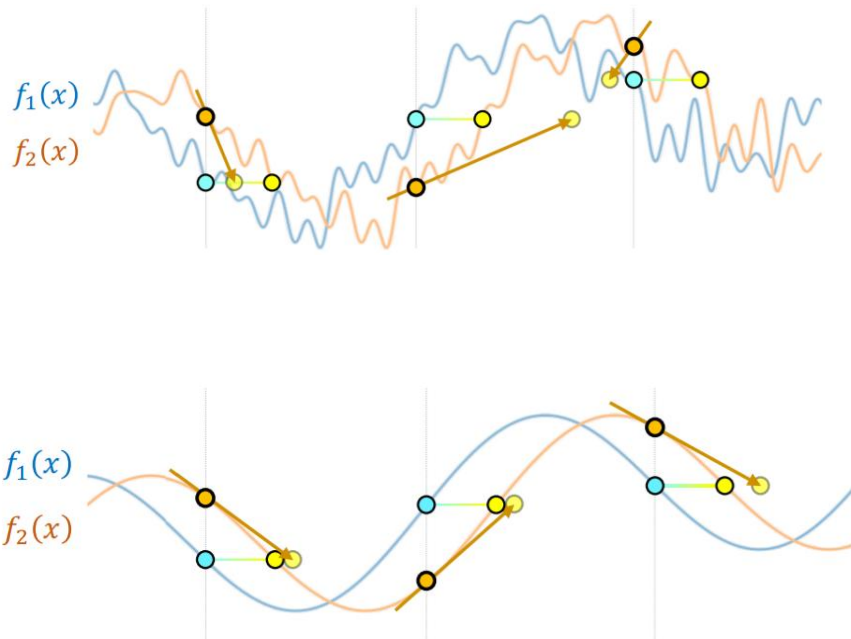


# BARF – Bundle-Adjusting NeRF

Поскольку обратное распространение ошибки в параметры камеры проходит через positional encodings, градиент ошибки по отношению к параметрам камеры может быть не очень гладким.

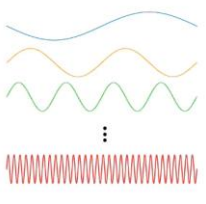
Тогда может быть сложно искать оптимальные параметры камеры, т.к. на значение градиента сильно влияют высокочастотные positional encodings.

Градиенты хорошо бы распространялись если бы эти высокие частоты не мешали при оптимизации параметров камеры.



# BARF – Bundle-Adjusting NeRF

NeRF:

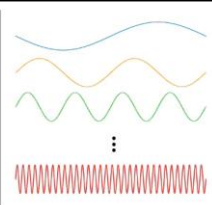
$$\gamma(\mathbf{x}) = \begin{bmatrix} \gamma_0(\mathbf{x}) \\ \gamma_1(\mathbf{x}) \\ \gamma_2(\mathbf{x}) \\ \vdots \\ \gamma_{L-1}(\mathbf{x}) \end{bmatrix}$$




- Color  
- Density

# BARF – Bundle-Adjusting NeRF

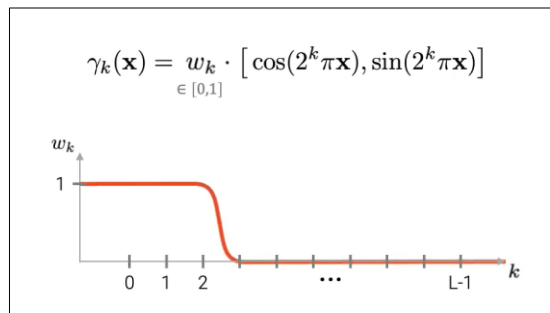
NeRF:

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




- Color  
- Density

BARF:

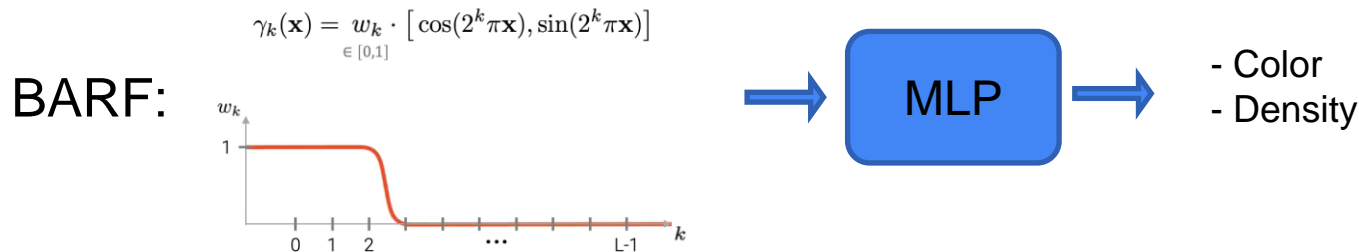


- Color  
- Density

$w_k(\alpha)$

Зануляет все positional encodings с частотами больше  $2^\alpha$

# BARF – Bundle-Adjusting NeRF

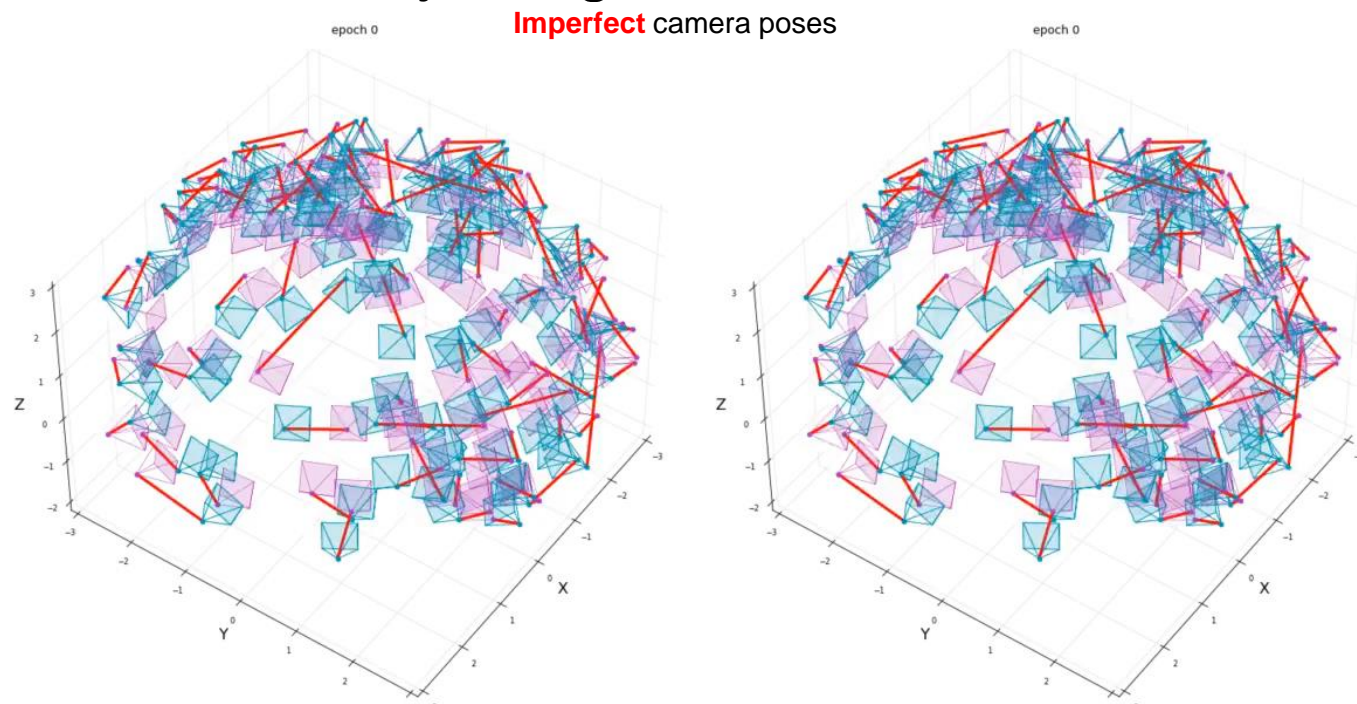


$w_k(\alpha)$  Зануляет все positional encodings с частотами больше  $2^\alpha$

Начинаем обучение с  $\alpha = 1$ , в течение первых  $K$  эпох постепенно повышаем до  $L$ , где  $L$  – количество синусойд в positional encodings, т.е.  $w_k(\alpha)$  перестают действовать

Таким образом, позиции камер учатся сначала когда используются только низкие частоты с хорошими градиентами, и со временем, когда камеры уже успели подобраться, подключаются все positional encodings чтобы NeRF мог спокойно дообучиться.

# BARF – Bundle-Adjusting NeRF

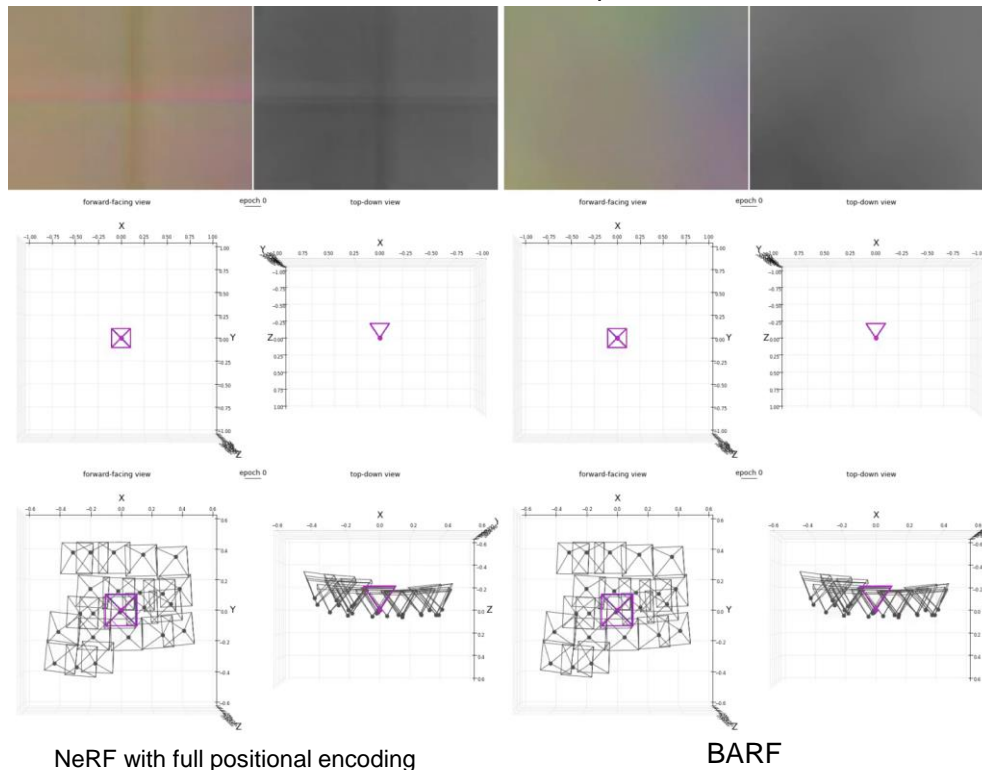


NeRF with full positional encoding

BARF

# BARF – Bundle-Adjusting NeRF

Unknown camera poses





# ИСТОЧНИКИ

SIFT - <https://medium.com/data-breach/introduction-to-sift-scale-invariant-feature-transform-65d7f3a72d40>  
<https://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf>

Phototourism (большинство хороших визуализаций из их презентации) -  
<http://phototour.cs.washington.edu/>

Structure From Motion overview -

[https://openaccess.thecvf.com/content\\_cvpr\\_2016/papers/Schonberger\\_Structure-From-Motion\\_Revisited\\_CVPR\\_2016\\_paper.pdf](https://openaccess.thecvf.com/content_cvpr_2016/papers/Schonberger_Structure-From-Motion_Revisited_CVPR_2016_paper.pdf)

Intro to Structure from motion - <https://kb.unavco.org/kb/file.php?id=810>

BARF - <https://chenhsuanlin.bitbucket.io/bundle-adjusting-NeRF/>