Летние стажировки по фотограмметрии в Agisoft

Agisoft Metashape

polarnick@agisoft.com

Полярный Николай

Потенциальный диплом/статья:

1) **NERF** для построения ортофотоплана - т.е. адаптация в 2.5D
2) Улучшение дескриптора **SIFT** для **LIDAR** облаков точек
3) Реализовать быстрый **MRF** оптимизатор для задачи текстурирования (идея)
4) Упаковка текстурного атласа (математика/геометрия/олимп. прогр.)
5) Воспроизвести, изучить, адаптировать для больших случаев: PermutoSDF

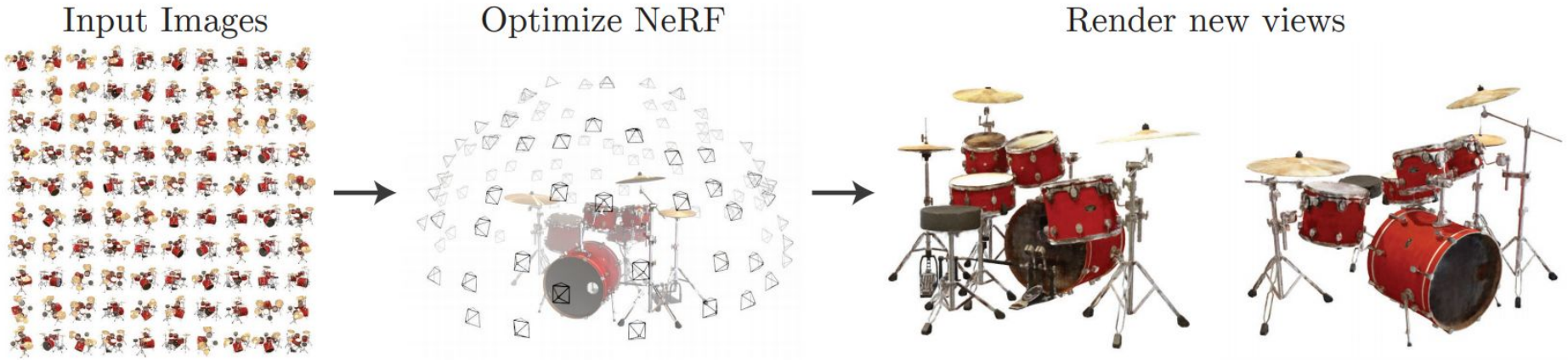
Другое:

6) Улучшение текстуры - насыщенность/resharpening
7) Super-resolution тепловых снимков и мультиспектральных спутников
8) Раскраска вершин модели по фотографиям (Vulkan API)
9) Ретуширование ортофотоплана (healing brush)

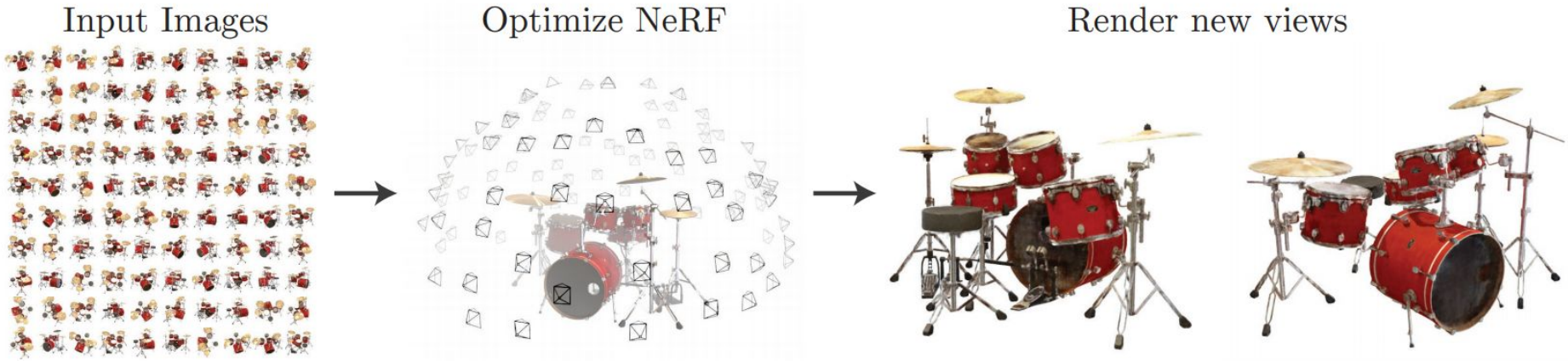
Python:

10) Сделать Python-скрипт для удаления людей с фотографий
11) Внедрение Python-прототипа нейронки (с дообучением) в C++ приложение
12) Сделать Python-скрипт для возможности использовать нейронные методы реконструкции карт глубины, сравнить SGM и PatchMatch с нейронным SOTA

1) **NERF** для построения ортофотоплана - т.е. адаптация в 2.5D

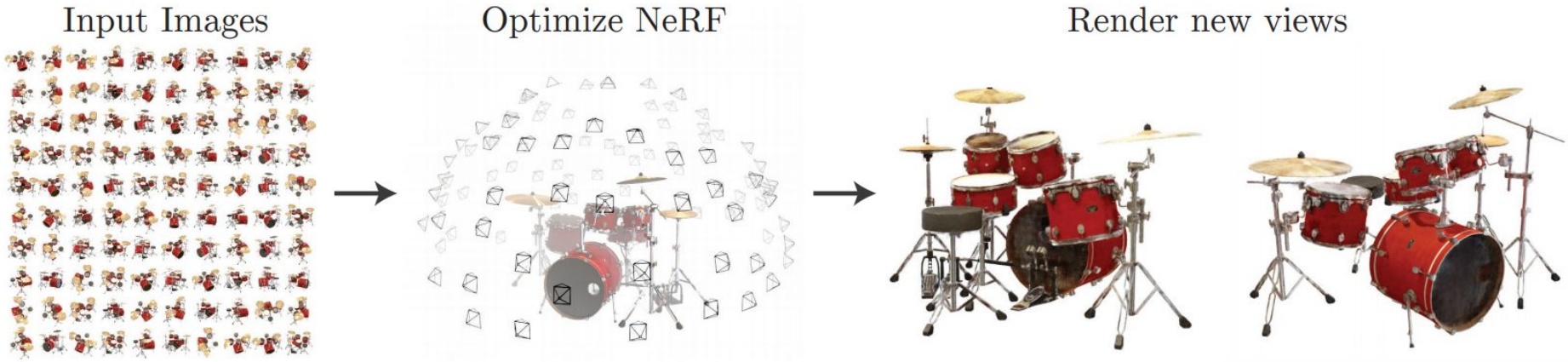


1) **NERF** для построения ортофотоплана - т.е. адаптация в 2.5D



Что **NERF** делает хорошо? “Фотографии” сцены с нового ракурса (**novel views**).

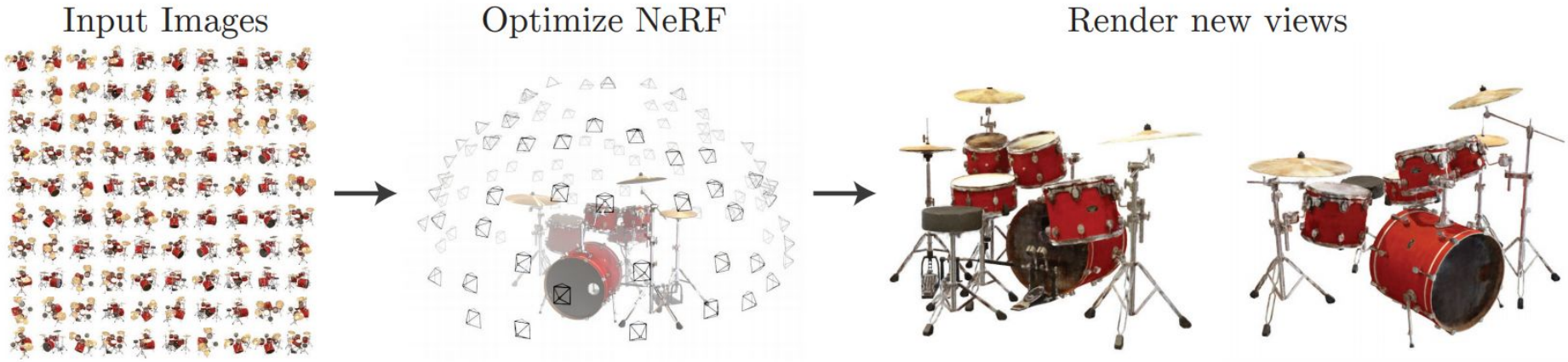
1) **NERF** для построения ортофотоплана - т.е. адаптация в 2.5D



Что **NERF** делает хорошо? “Фотографии” сцены с нового ракурса (**novel views**).

Где в фотограмметрии нужно генерировать **novel view**?

1) **NERF** для построения ортофотоплана - т.е. адаптация в 2.5D

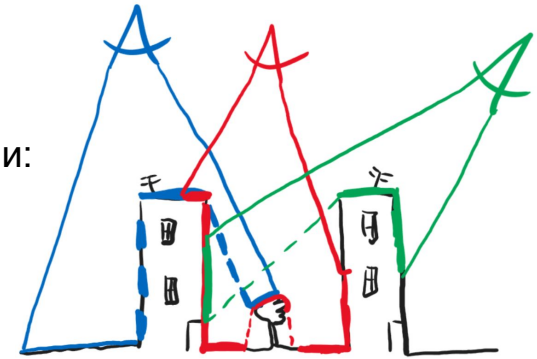


Что **NERF** делает хорошо? “Фотографии” сцены с нового ракурса (**novel views**).

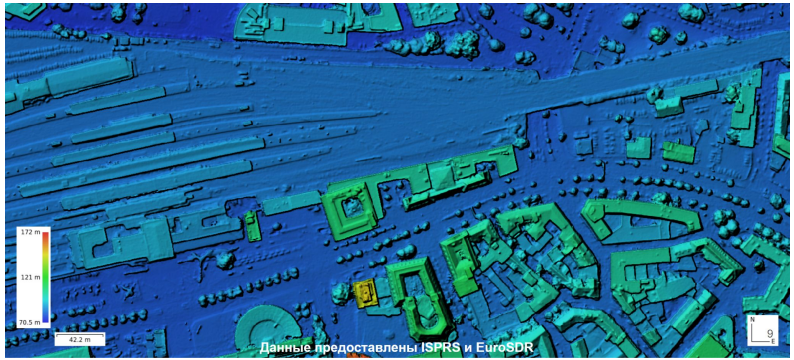
Где в фотограмметрии нужно генерировать **novel view**? Вид сверху! Ортофото!

1) **NERF** для построения ортофотоплана - т.е. адаптация в 2.5D

Проецировали:

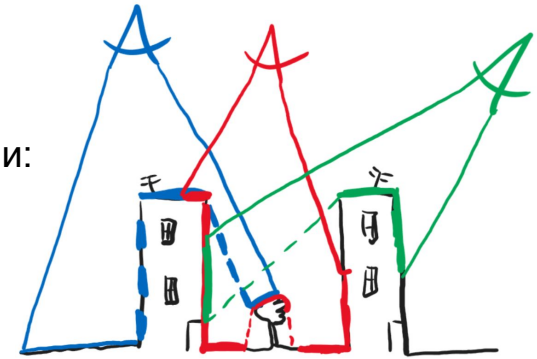


на **DEM**:

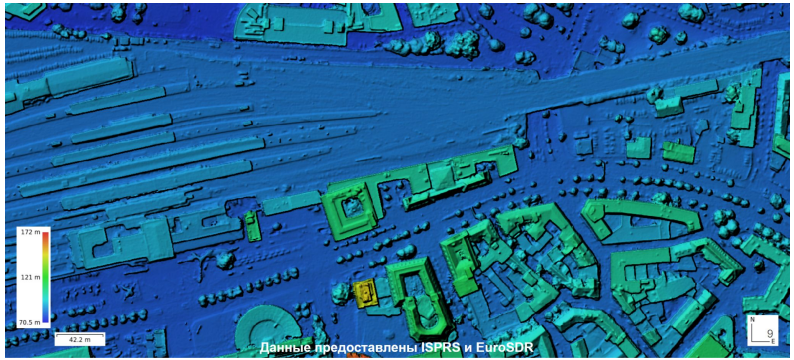


1) **NERF** для построения ортофотоплана - т.е. адаптация в 2.5D

Проецировали:

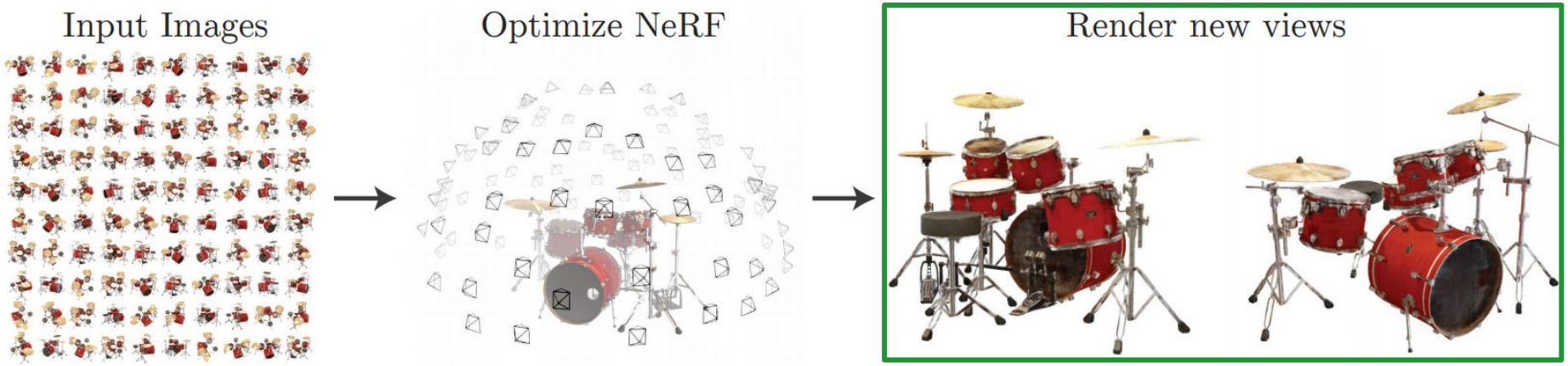


на **DEM**:

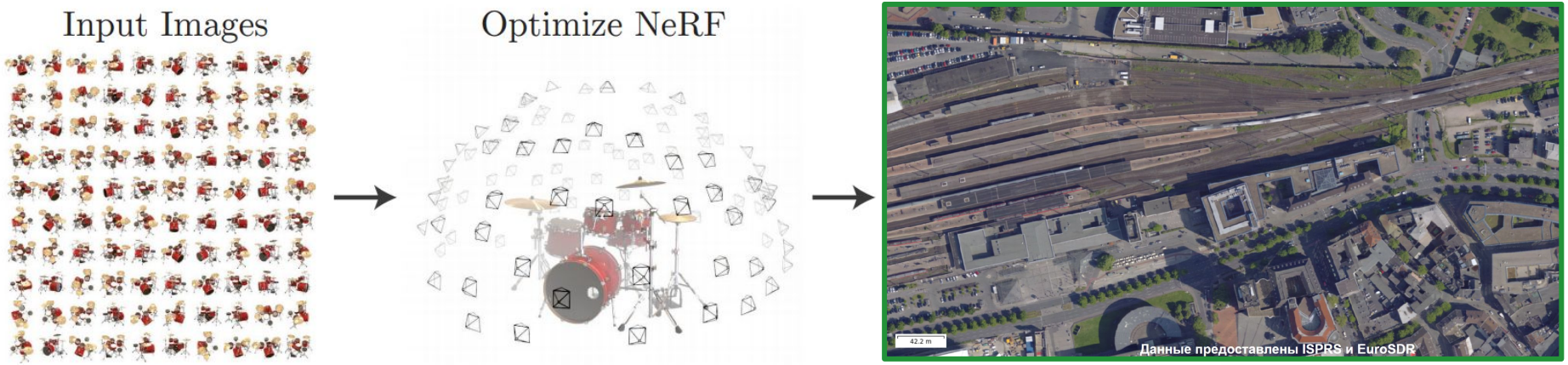


Получали ортофотоплан (вид сверху):

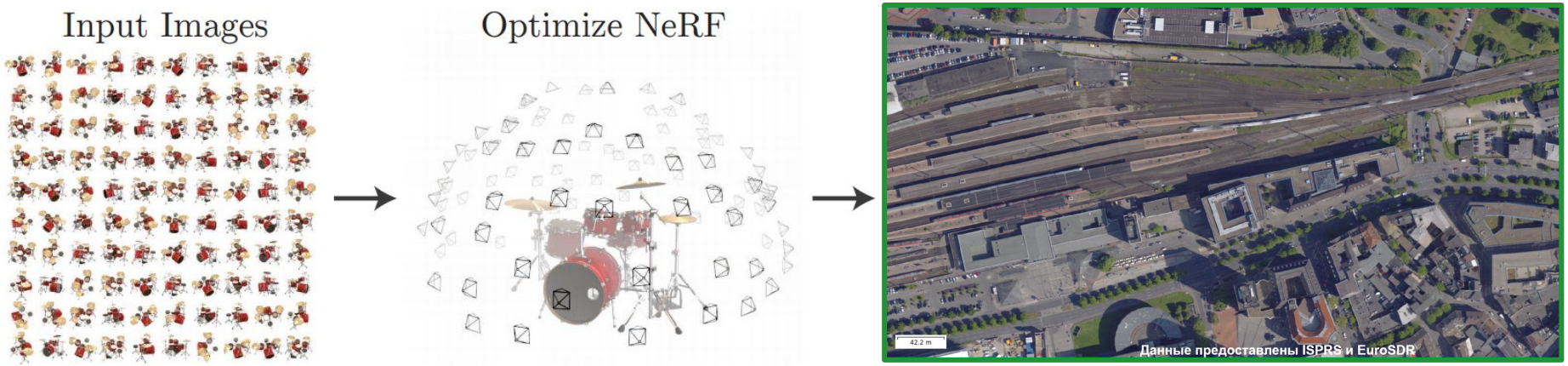
1) **NERF** для построения ортофотоплана - т.е. адаптация в 2.5D



1) **NERF** для построения ортофотоплана - т.е. адаптация в 2.5D

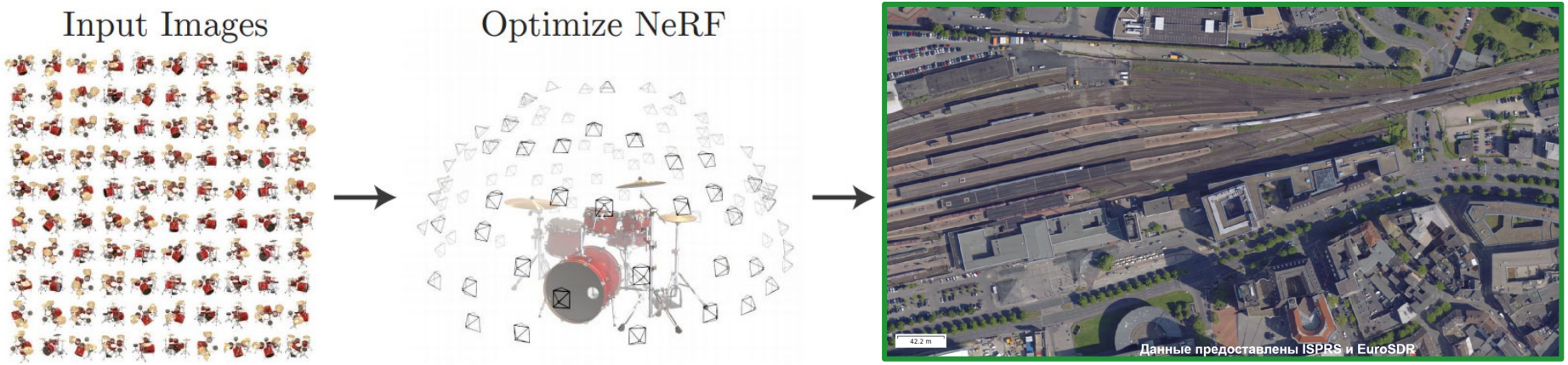


1) **NERF** для построения ортофотоплана - т.е. адаптация в 2.5D

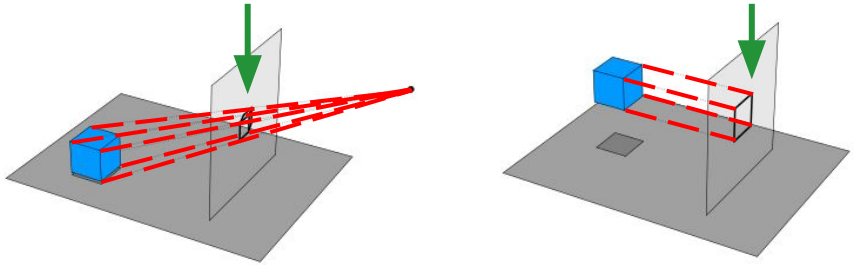


1) Воспроизвести NERF на небольшом аэрофото датасете

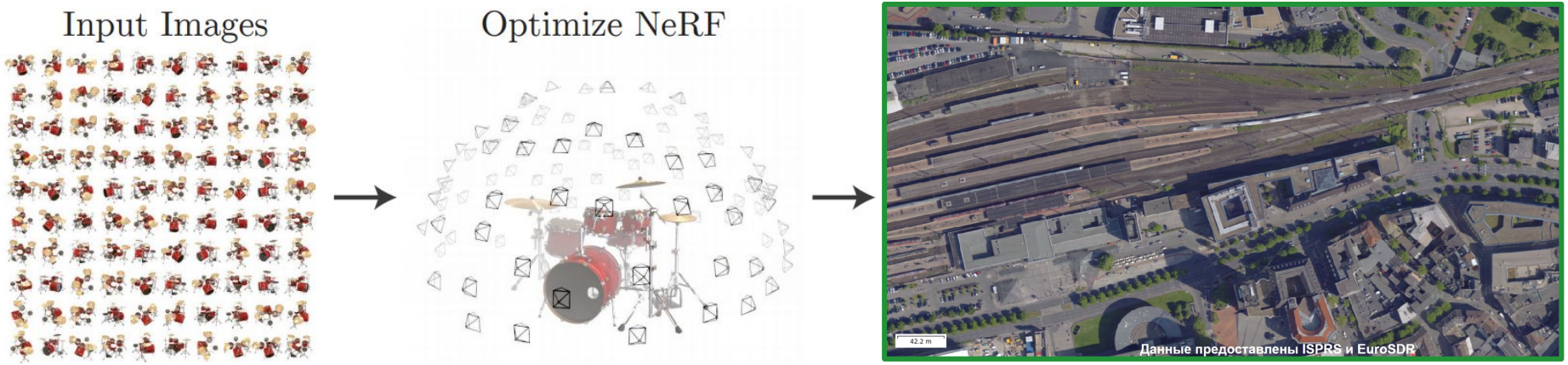
1) **NERF** для построения ортофотоплана - т.е. адаптация в 2.5D



1) Воспроизвести NERF на небольшом аэрофото датасете
2) Заменить модель камеры novel view с перспективной на ортопроекцию

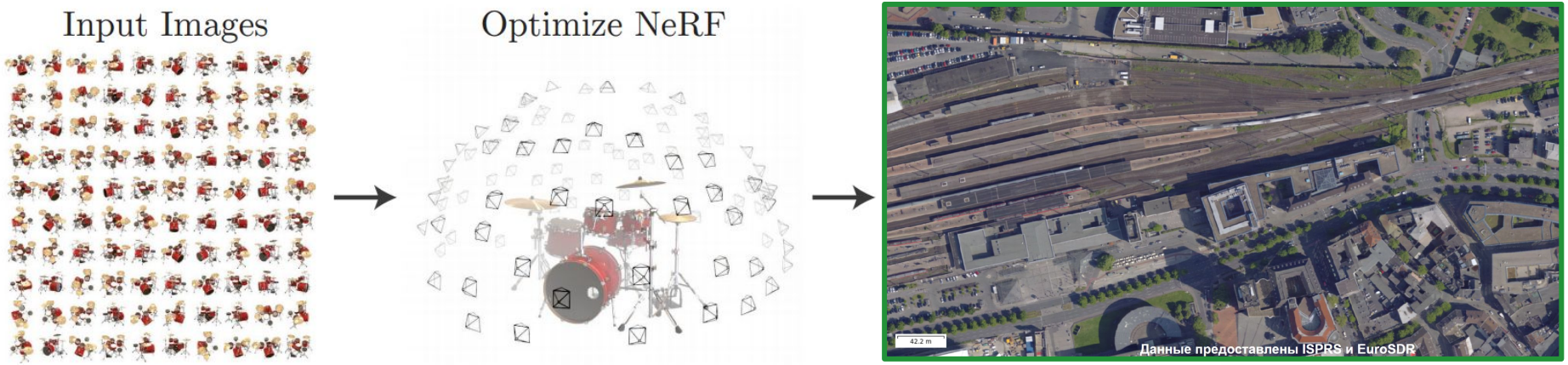


1) **NERF** для построения ортофотоплана - т.е. адаптация в 2.5D



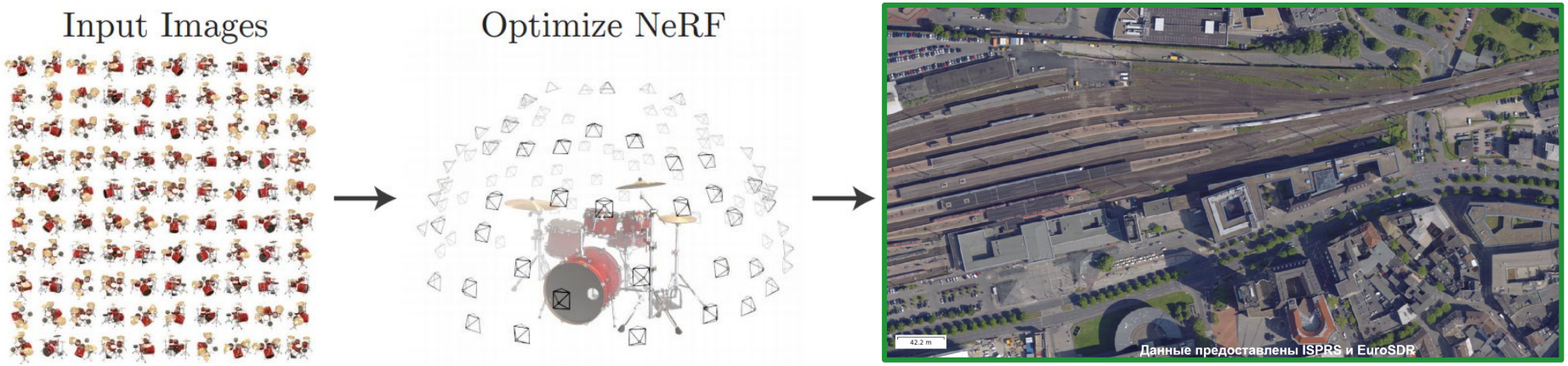
1) Воспроизвести NERF на небольшом аэрофото датасете
2) Заменить модель камеры novel view с перспективной на ортопроекцию
3) Учесть специфику 2.5D пространства:
 - ускорить обработку заменив по-воксельную работу на по-пиксельную

1) **NERF** для построения ортофотоплана - т.е. адаптация в 2.5D



1) Воспроизвести NERF на небольшом аэрофото датасете
2) Заменить модель камеры novel view с перспективной на ортопроекцию
3) Учесть специфику 2.5D пространства:
 - ускорить обработку заменив по-воксельную работу на по-пиксельную
 - в каждом пикселе ищем высоту пикселя + цвета для разных ракурсов

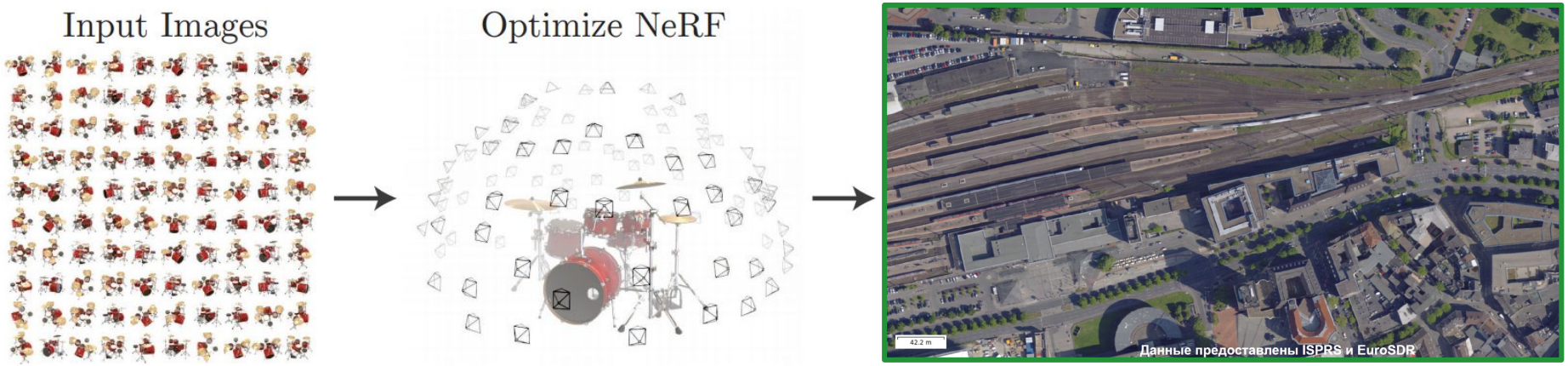
1) **NERF** для построения ортофотоплана - т.е. адаптация в 2.5D



1) Воспроизвести NERF на небольшом аэрофото датасете
2) Заменить модель камеры novel view с перспективной на ортопроекцию
3) Учесть специфику 2.5D пространства:
 - ускорить обработку заменив по-воксельную работу на по-пиксельную
 - в каждом пикселе ищем высоту пикселя + цвета для разных ракурсов
 - **coarse-to-fine** схема как в **tSGM** (прореженная регулярная решетка)

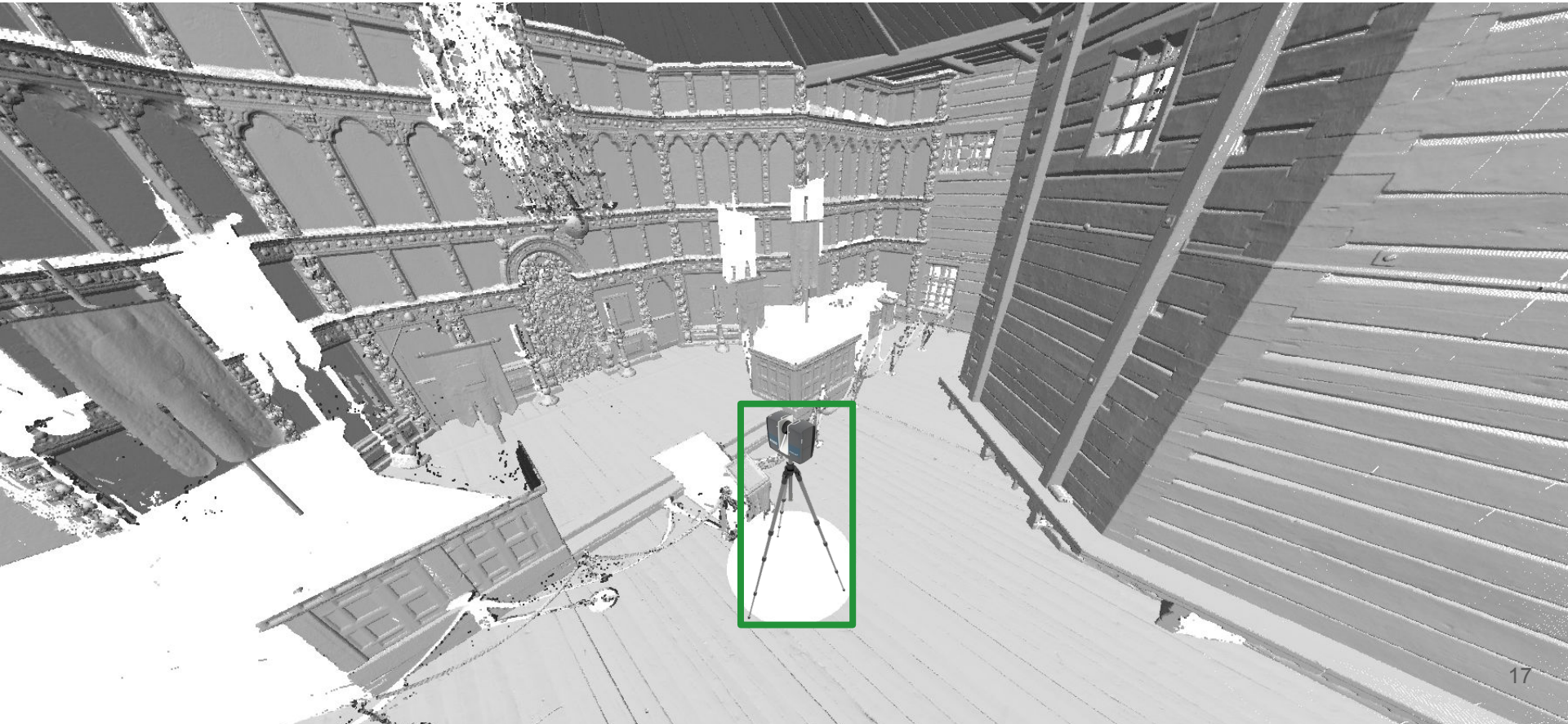
Вопросы?

1) **NERF** для построения ортофотоплана - т.е. адаптация в 2.5D



1) Воспроизвести NERF на небольшом аэрофото датасете
2) Заменить модель камеры novel view с перспективной на ортопроекцию
3) Учесть специфику 2.5D пространства:
 - ускорить обработку заменив по-воксельную работу на по-пиксельную
 - в каждом пикселе ищем высоту пикселя + цвета для разных ракурсов
 - **coarse-to-fine** схема как в **tSGM** (прореженная регулярная решетка)

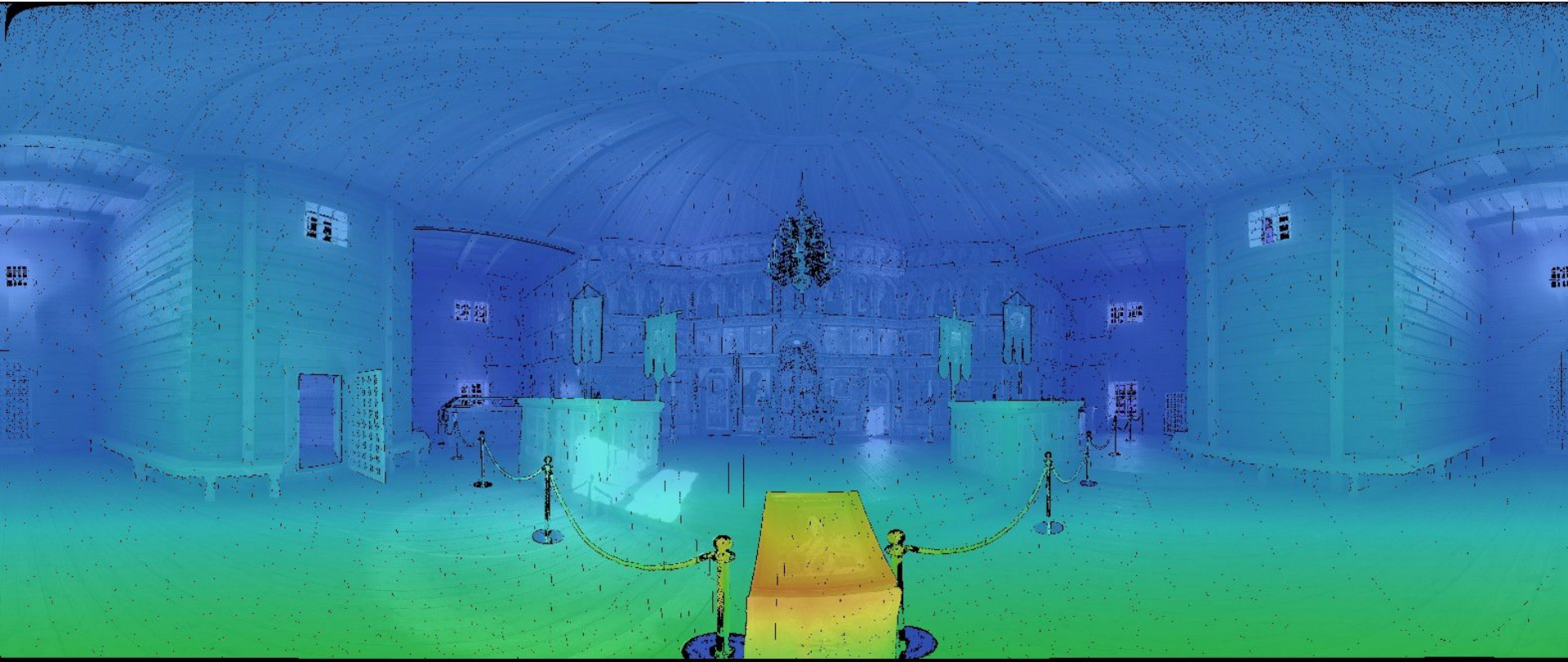
2) Улучшение дескриптора **SIFT** для **LIDAR** облаков точек



2) Улучшение дескриптора **SIFT** для **LIDAR** облаков точек

LIDAR легко представить в виде 360-градусов сферической камеры

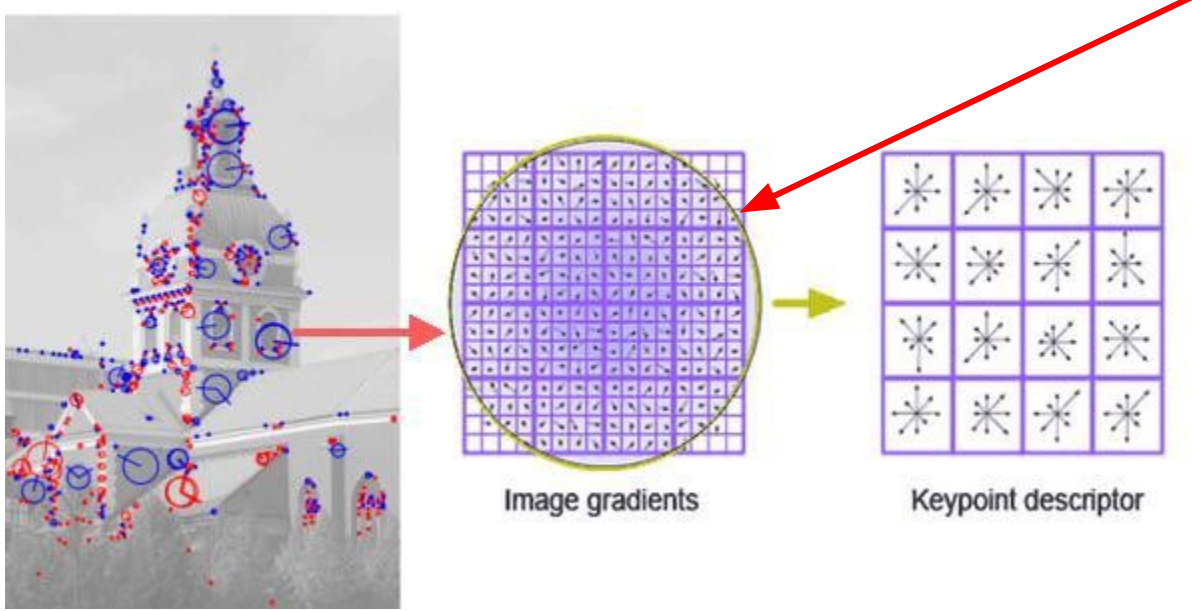
2) Улучшение дескриптора **SIFT** для **LIDAR** облаков точек



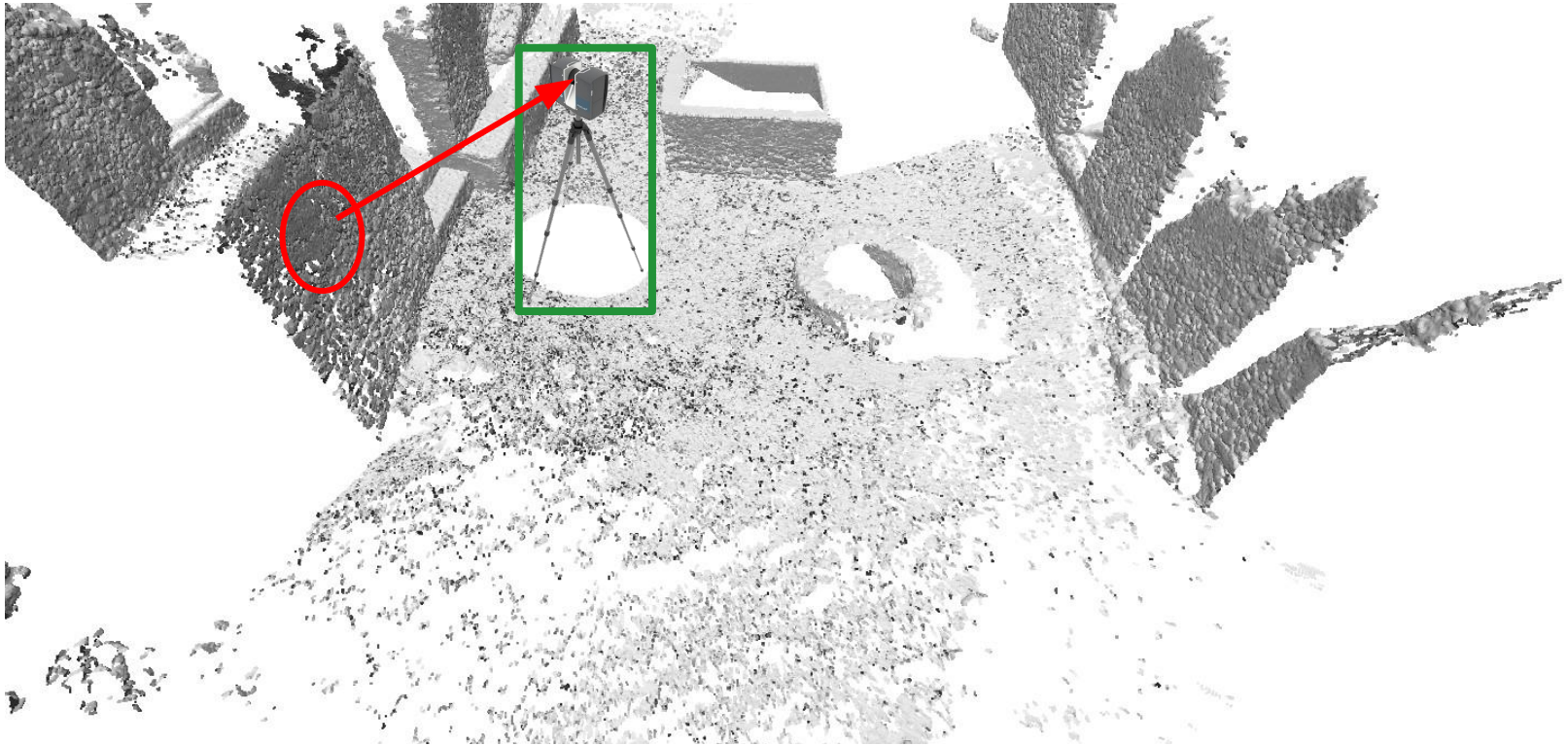
LIDAR легко представить в виде 360-градусов сферической камеры

2) Улучшение дескриптора **SIFT** для **LIDAR** облаков точек

Не знаем перспективное искажение в фотографиях - дескриптор по окружности

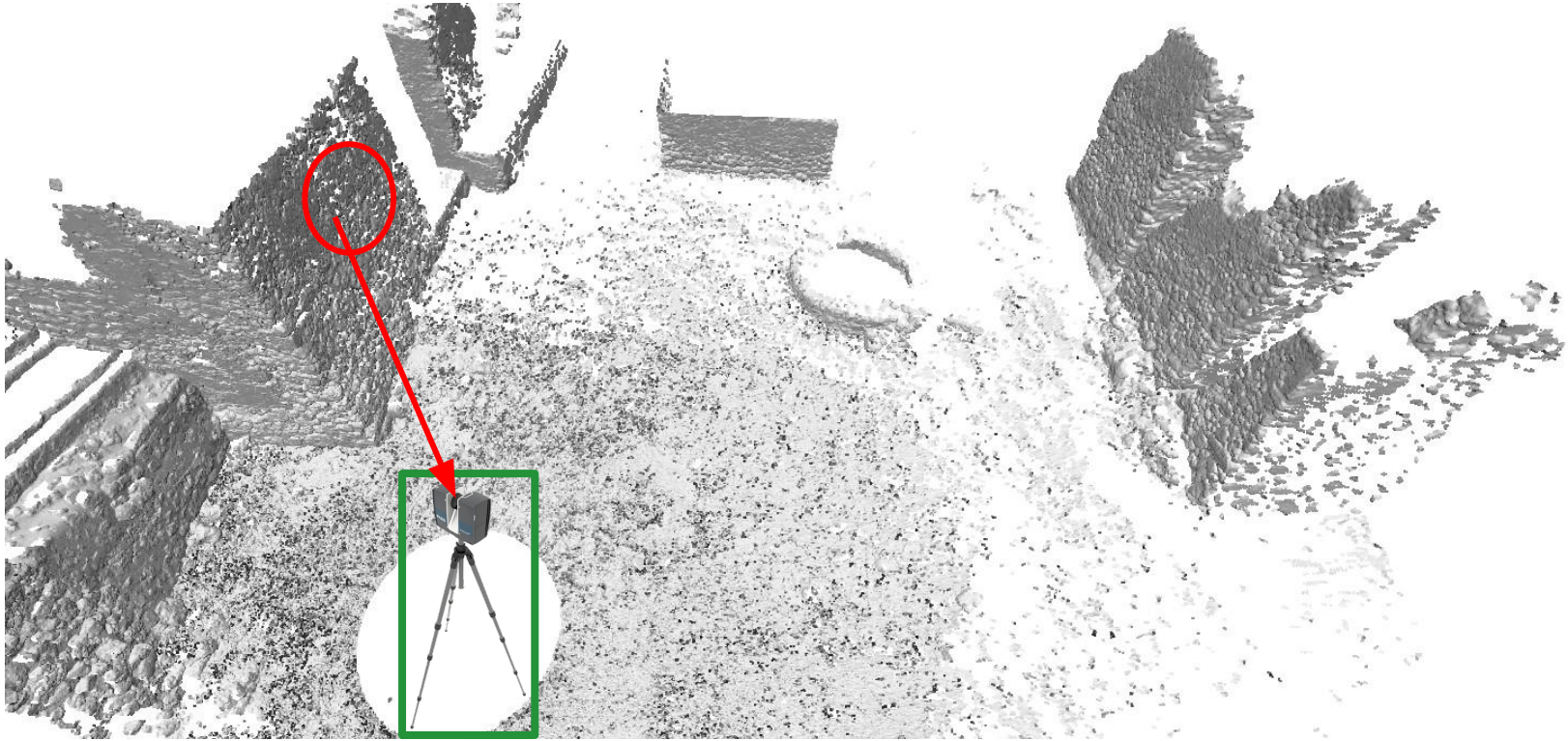


2) Улучшение дескриптора **SIFT** для **LIDAR** облаков точек



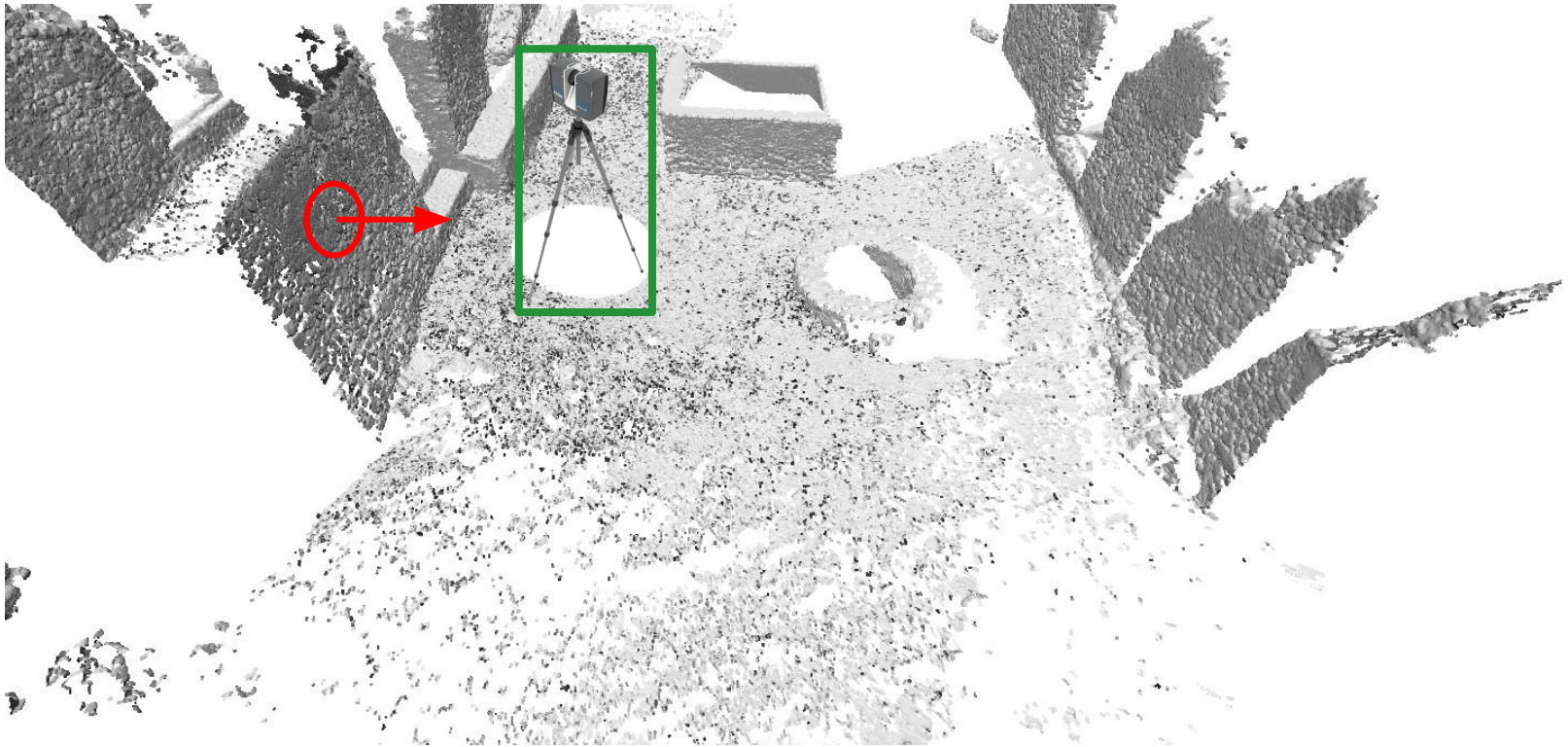
Дескриптор по окружности - не учитывает перспективное искажение

2) Улучшение дескриптора **SIFT** для **LIDAR** облаков точек



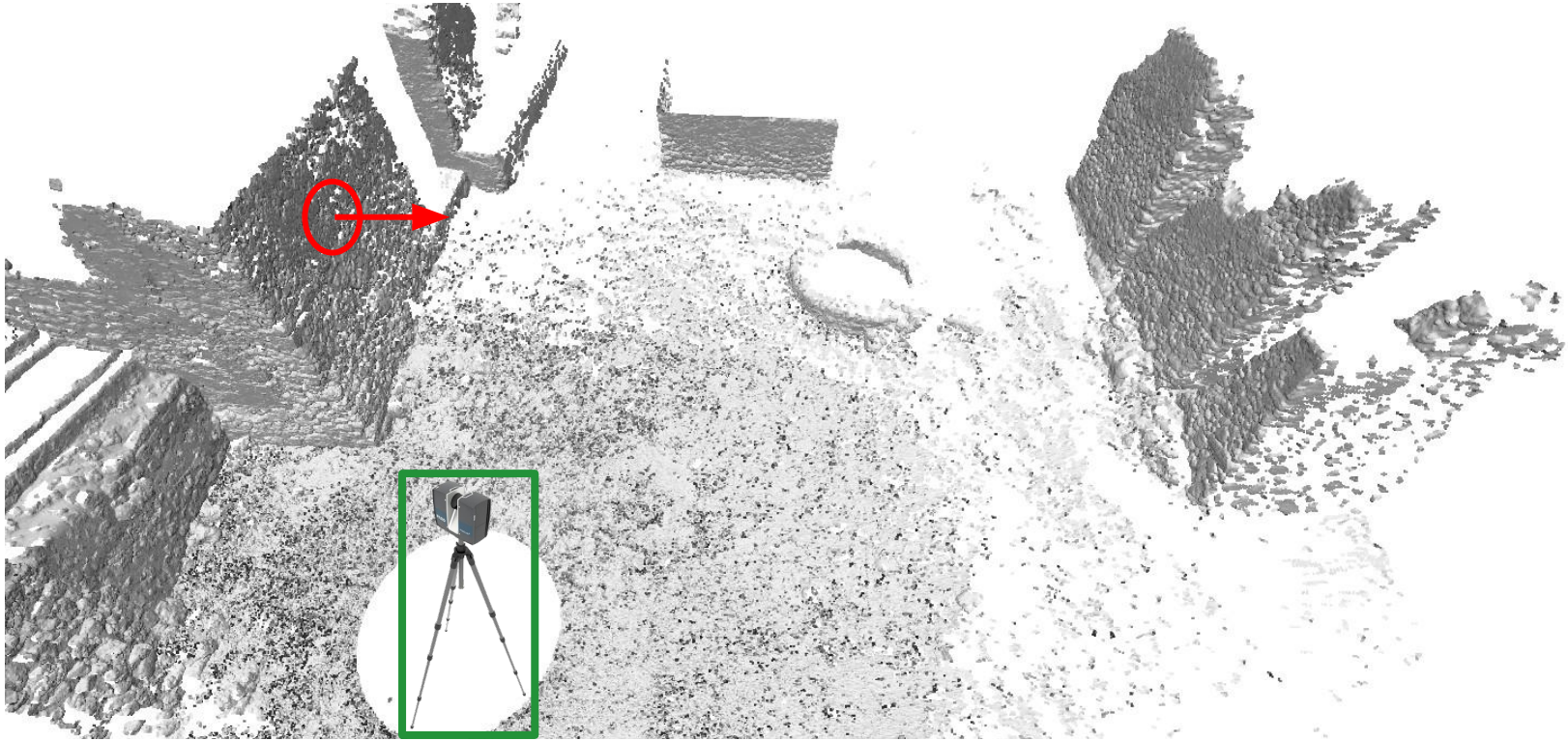
Дескриптор по окружности - не учитывает перспективное искажение

2) Улучшение дескриптора **SIFT** для **LIDAR** облаков точек



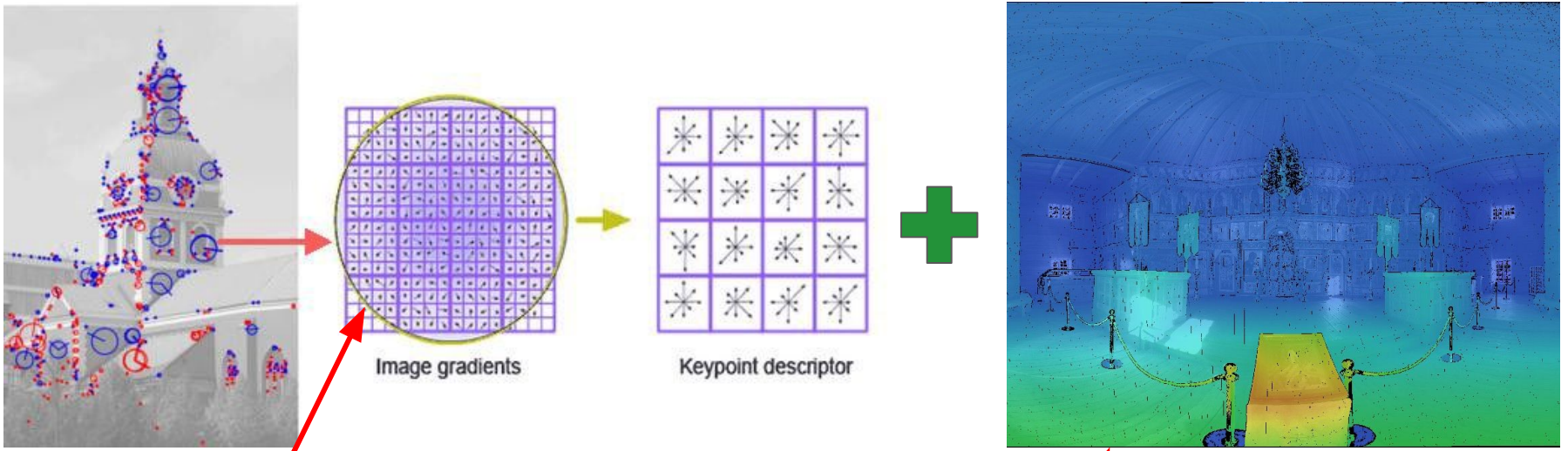
Благодаря глубине знаем **нормаль** - учли перспективное искажение

2) Улучшение дескриптора **SIFT** для **LIDAR** облаков точек



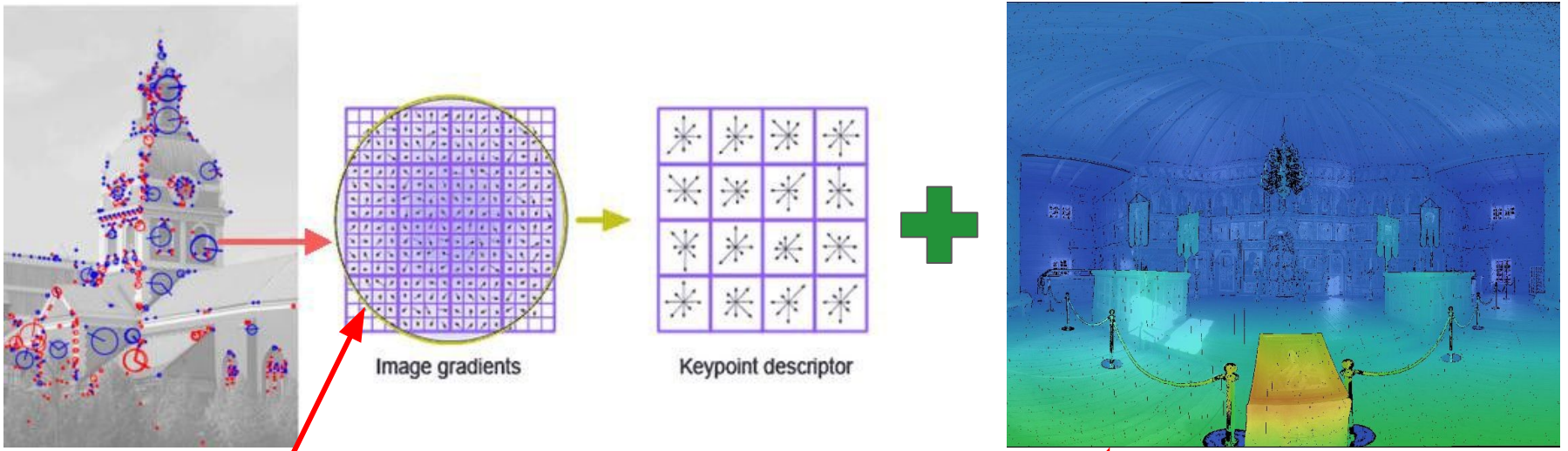
Благодаря глубине знаем **нормаль** - учли перспективное искажение

2) Улучшение дескриптора **SIFT** для **LIDAR** облаков точек



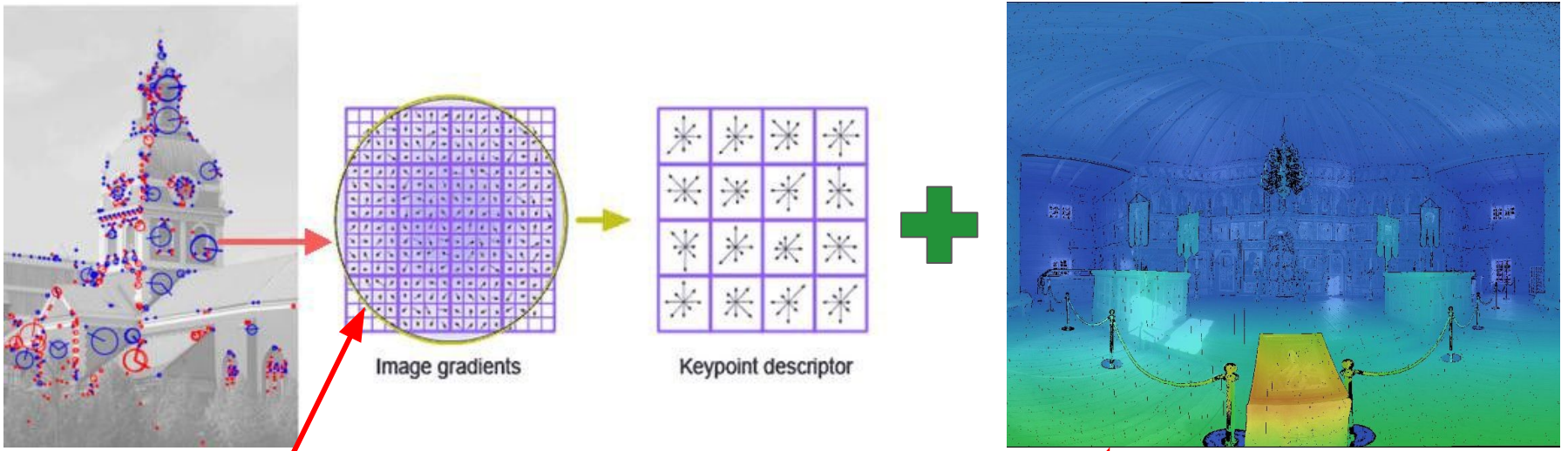
1) В существующем алгоритме построения дескриптора заменить окрестность с регулярной окружности на скорректированную (с учетом перспективных искажений)

2) Улучшение дескриптора **SIFT** для **LIDAR** облаков точек



1) В существующем алгоритме построения дескриптора заменить окрестность с регулярной окружности на скорректированную (с учетом перспективных искажений)
2) Интересует качество выравнивания **LIDAR vs LIDAR** и **LIDAR vs RGB** фото

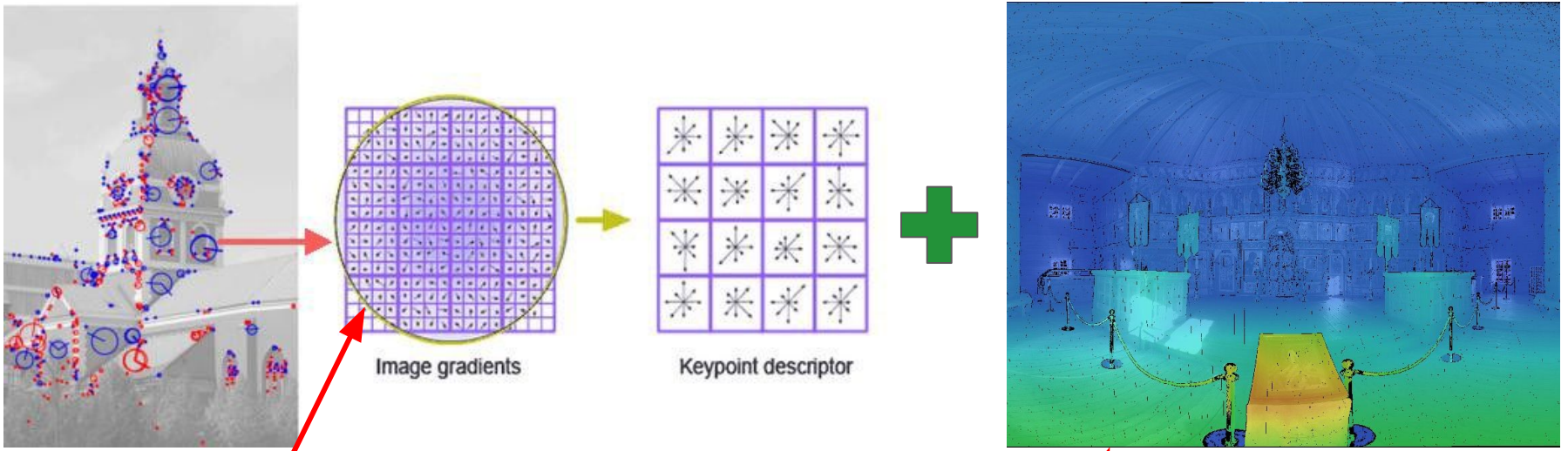
2) Улучшение дескриптора **SIFT** для **LIDAR** облаков точек



1) В существующем алгоритме построения дескриптора заменить окрестность с регулярной окружности на скорректированную (с учетом перспективных искажений)
2) Интересует качество выравнивания **LIDAR vs LIDAR** и **LIDAR vs RGB** фото
3) Найти статьи с идеями для матчинга RGB-D изображений и исследовать перспективные

Вопросы?

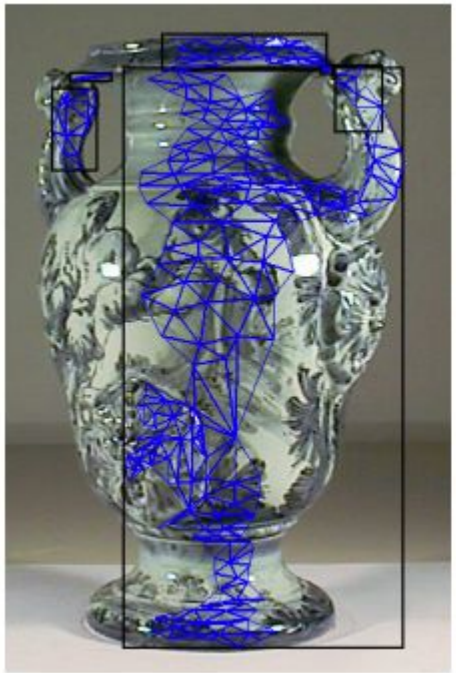
2) Улучшение дескриптора **SIFT** для **LIDAR** облаков точек



1) В существующем алгоритме построения дескриптора заменить окрестность с регулярной окружности на скорректированную (с учетом перспективных искажений)
2) Интересует качество выравнивания **LIDAR vs LIDAR** и **LIDAR vs RGB** фото
3) Найти статьи с идеями для матчинга RGB-D изображений и исследовать перспективные

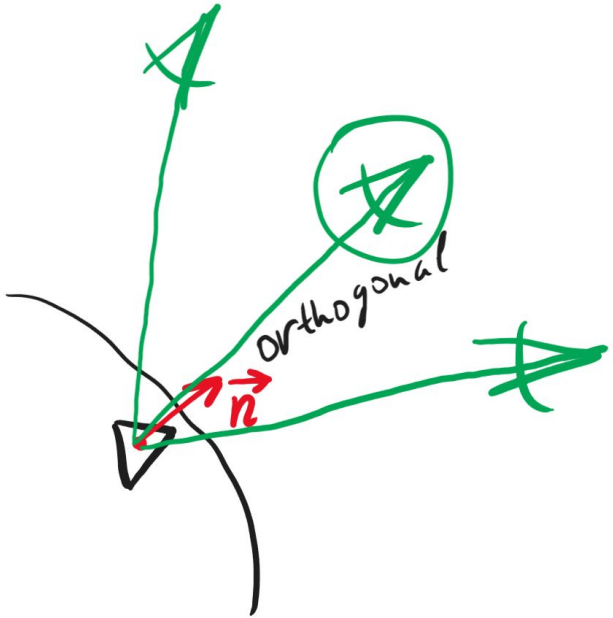
3) Реализовать быстрый **MRF** оптимизатор для задачи текстурирования (идея)

Для текстурирования каждого треугольника нужно выбрать лучшую камеру:



3) Реализовать быстрый **MRF** оптимизатор для задачи текстурирования (идея)

Для текстурирования каждого треугольника нужно выбрать лучшую камеру:
1) MRF функционал учитывает индивидульную лучшевизну камеры для треугольника.



3) Реализовать быстрый **MRF** оптимизатор для задачи текстурирования (идея)

Для текстурирования каждого треугольника нужно выбрать лучшую камеру:
1) MRF функционал учитывает индивидульную лучшевизну камеры для треугольника.
2) MRF функционал учитывает попарную лучшевизну камер для смежных треугольников.

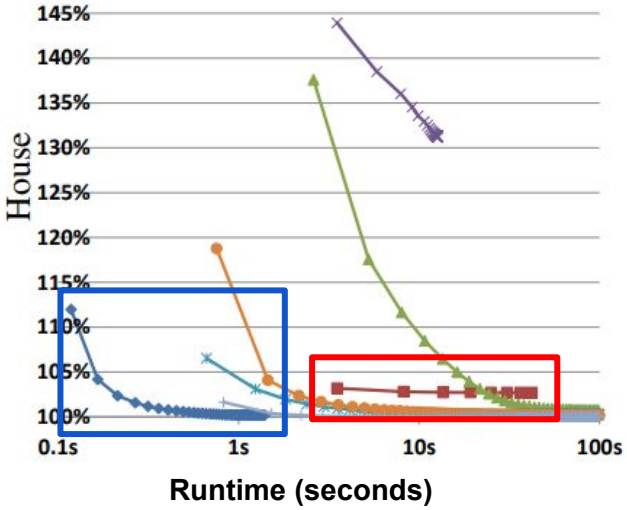
3) Реализовать быстрый **MRF** оптимизатор для задачи текстурирования (идея)

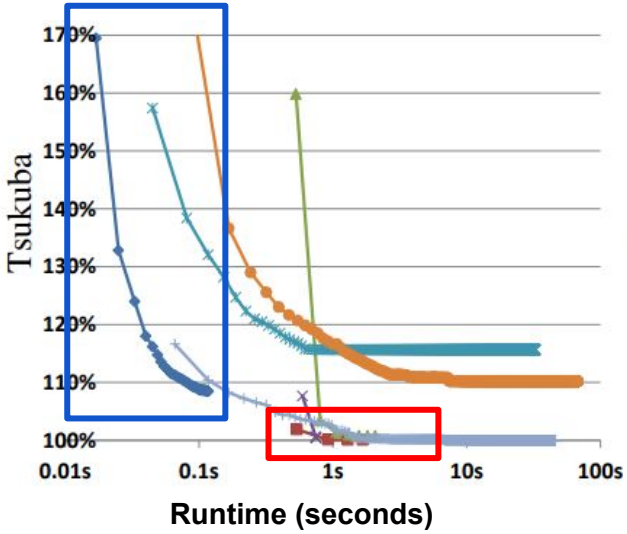
Можно решать через alpha-expansion, но это жадина + не параллелится.

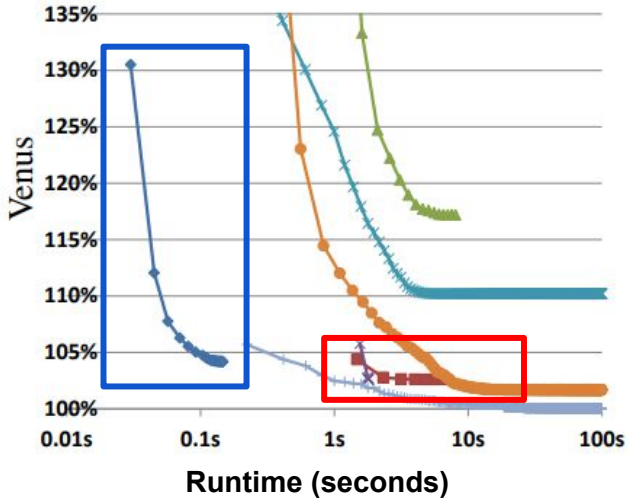
3) Реализовать быстрый **MRF** оптимизатор для задачи текстурирования (идея)

Можно решать через alpha-expansion, но это жадина + не параллелится.

Есть статья быстро решающая задачу на регулярной решетке:



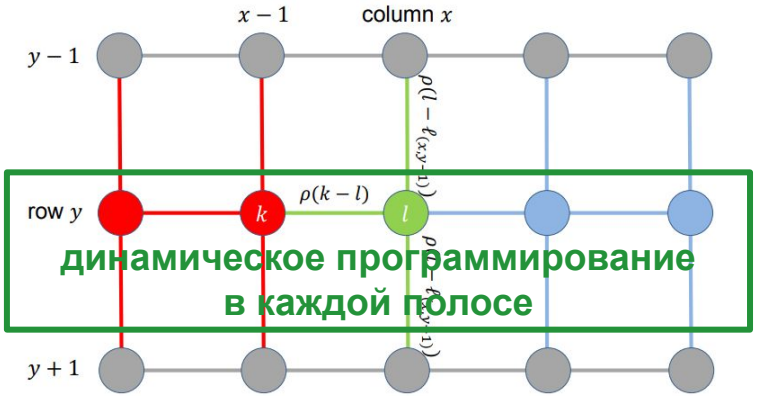




3) Реализовать быстрый **MRF** оптимизатор для задачи текстурирования (идея)

Можно решать через alpha-expansion, но это жадина + не параллелится.

Есть статья быстро решающая задачу на регулярной решетке:

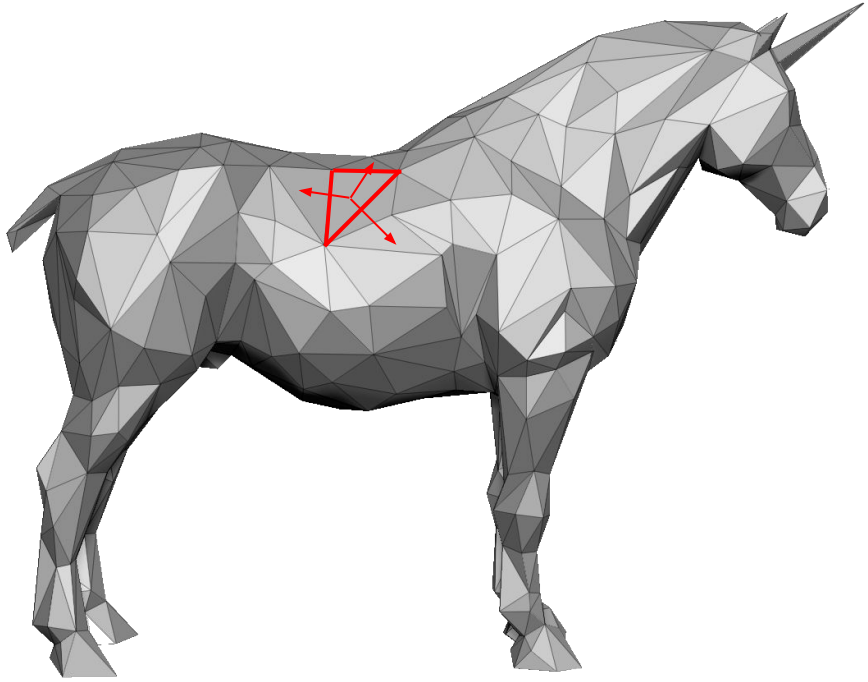


(остальные вершины заморожены)

Figure 1. Illustration of the dynamic programming algorithm. Label assignments to the grey nodes are fixed. Cumulative costs $\mathbf{C}$ for the red nodes have been computed and the algorithm is now processing the green node. To compute the cumulative cost $\mathbf{C}(x, y, l)$, the algorithm must minimize over terms involving possible assignments k to the previous node. Blue nodes will be processed subsequently.

3) Реализовать быстрый **MRF** оптимизатор для задачи текстурирования (идея)

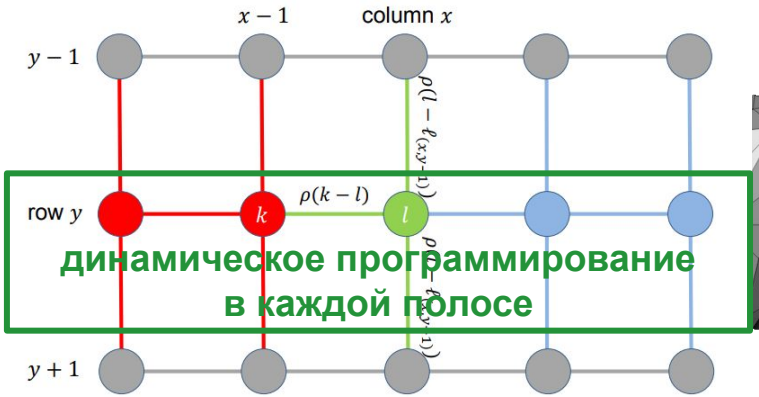
Можно решать через alpha-expansion, но это жадина + не параллелится.
Есть статья быстро решающая задачу. **Но у нас нерегулярная решетка:**



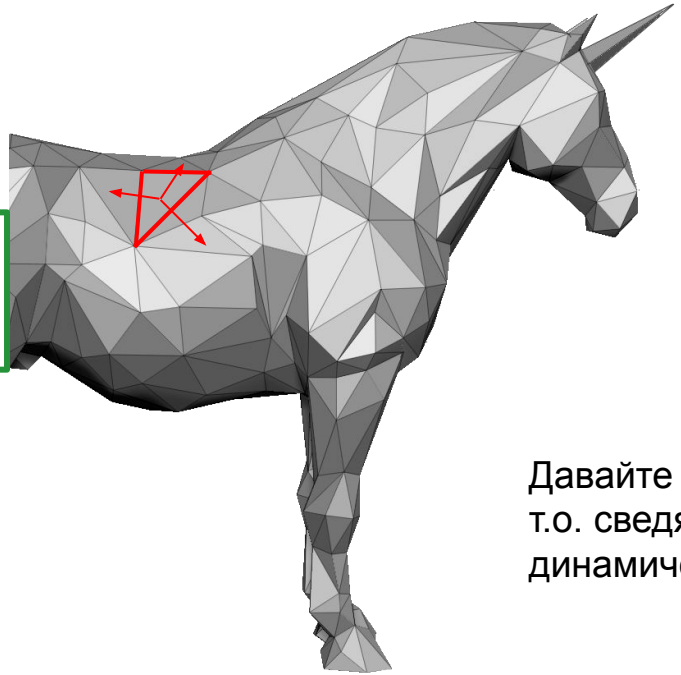
3) Реализовать быстрый **MRF** оптимизатор для задачи текстурирования (идея)

Можно решать через alpha-expansion, но это жадина + не параллелится.

Есть статья быстро решающая задачу. **Но у нас нерегулярная решетка:**



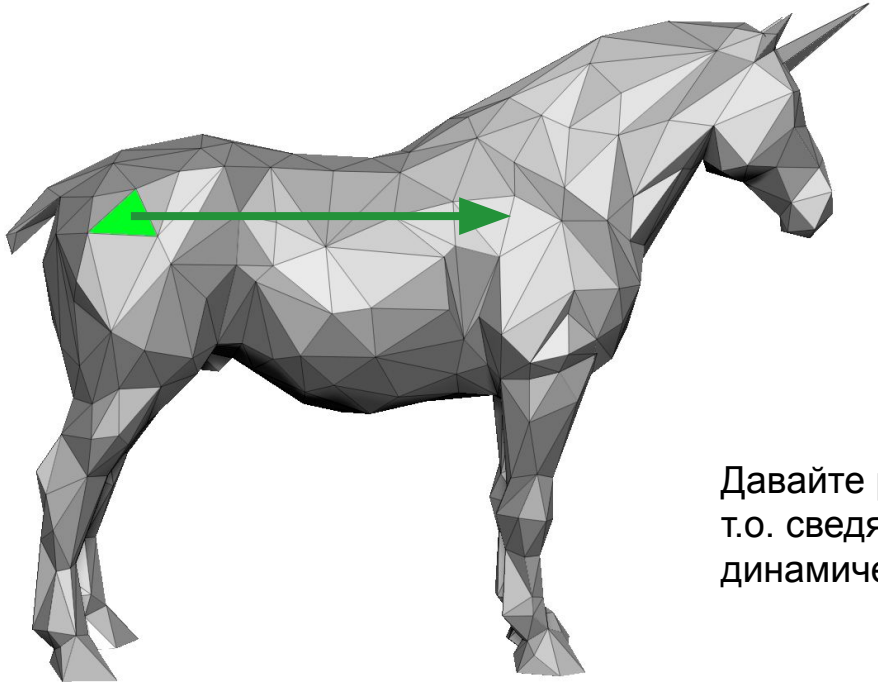
(остальные вершины заморожены)



Давайте разобьем на полосы,
т.о. сведя в каждой полосе задачу к
динамическому программированию.

3) Реализовать быстрый **MRF** оптимизатор для задачи текстурирования (идея)

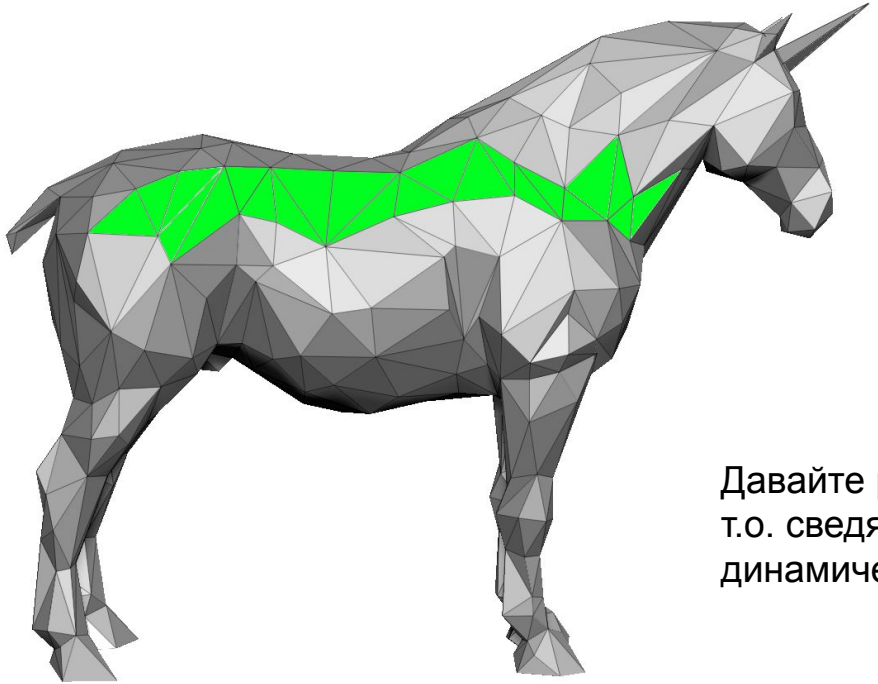
Можно решать через alpha-expansion, но это жадина + не параллелится.
Есть статья быстро решающая задачу. **Но у нас нерегулярная решетка:**



Давайте разобьем на полосы,
т.о. сведя в каждой полосе задачу к динамическому программированию.

3) Реализовать быстрый **MRF** оптимизатор для задачи текстурирования (идея)

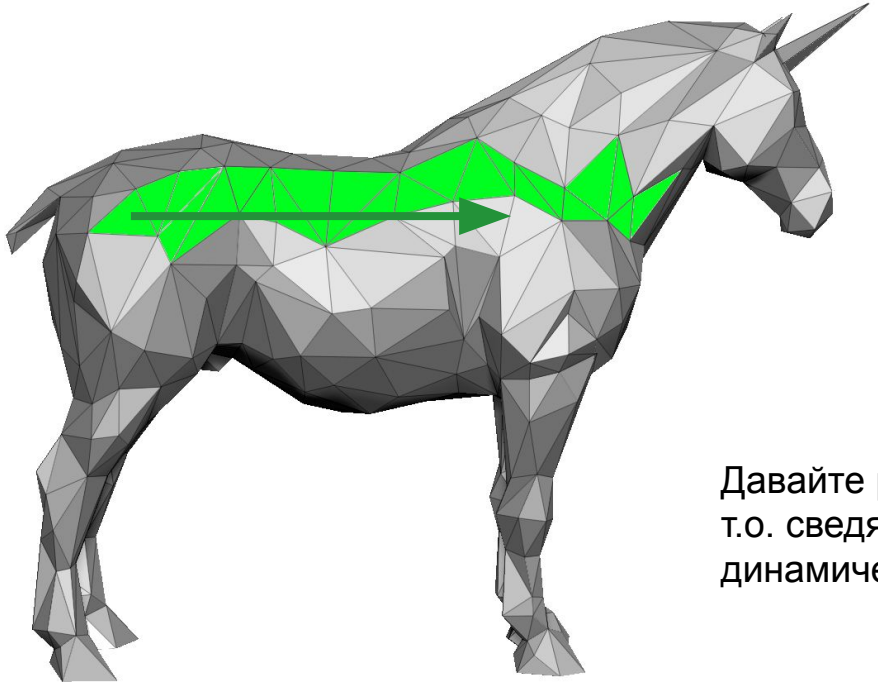
Можно решать через alpha-expansion, но это жадина + не параллелится.
Есть статья быстро решающая задачу. **Но у нас нерегулярная решетка:**



Давайте разобьем на полосы,
т.о. сведя в каждой полосе задачу к динамическому программированию.

3) Реализовать быстрый **MRF** оптимизатор для задачи текстурирования (идея)

Можно решать через alpha-expansion, но это жадина + не параллелится.
Есть статья быстро решающая задачу. **Но у нас нерегулярная решетка:**

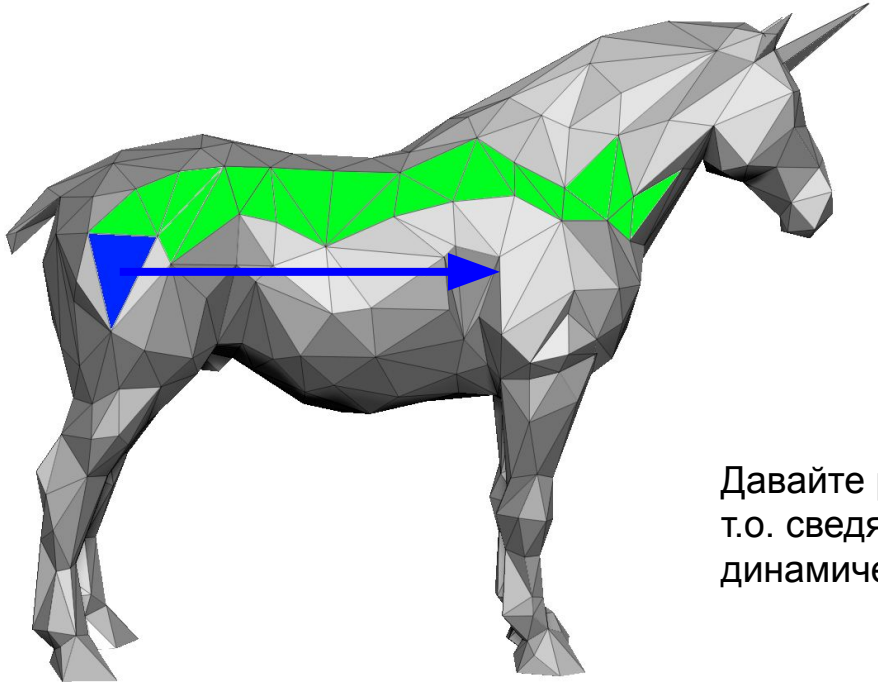


Давайте разобьем на полосы,
т.о. сведя в каждой полосе задачу к динамическому программированию.

3) Реализовать быстрый **MRF** оптимизатор для задачи текстурирования (идея)

Можно решать через alpha-expansion, но это жадина + не параллелится.

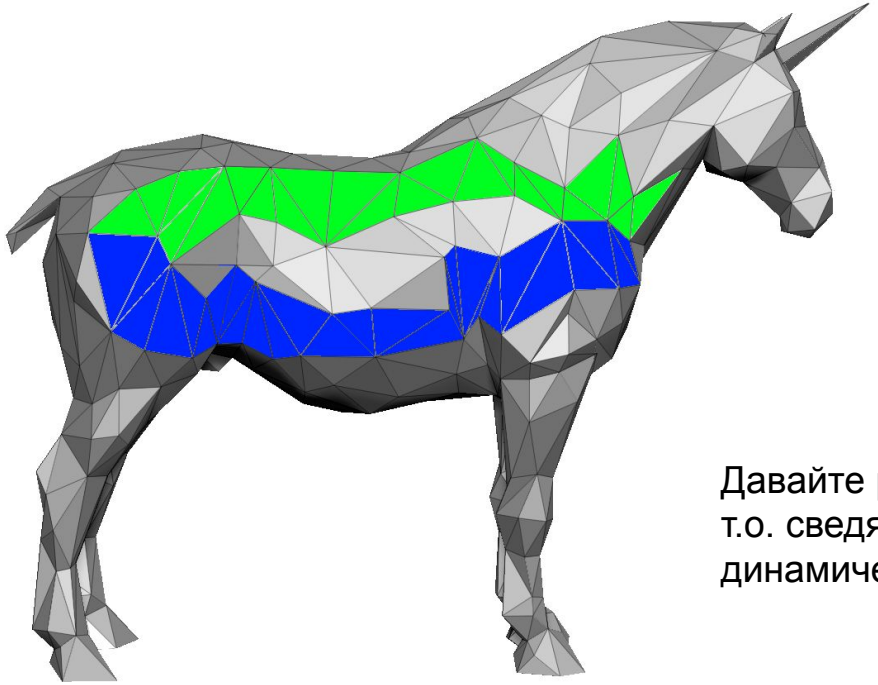
Есть статья быстро решающая задачу. **Но у нас нерегулярная решетка:**



Давайте разобьем на полосы,
т.о. сведя в каждой полосе задачу к динамическому программированию.

3) Реализовать быстрый **MRF** оптимизатор для задачи текстурирования (идея)

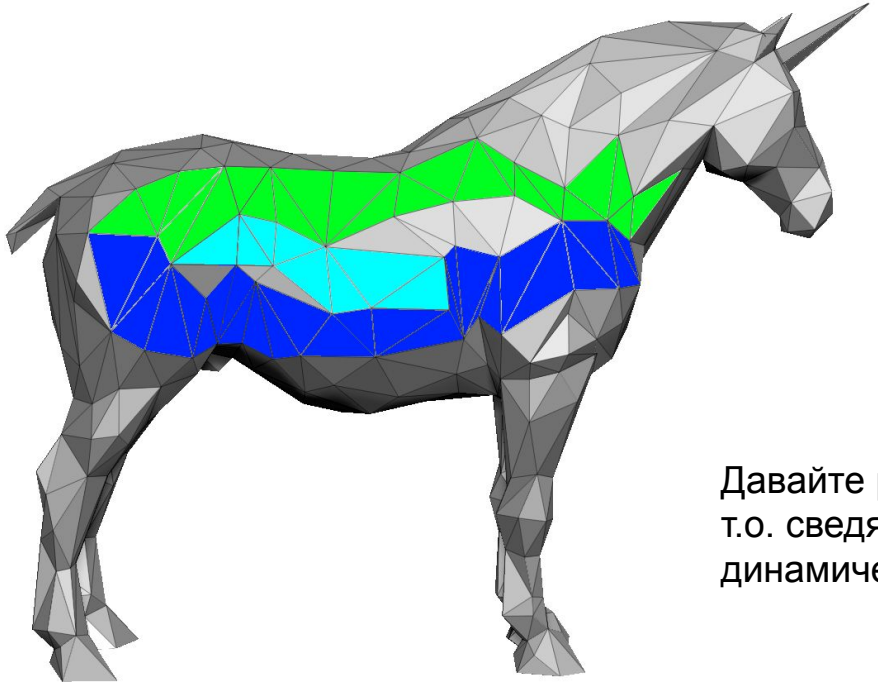
Можно решать через alpha-expansion, но это жадина + не параллелится.
Есть статья быстро решающая задачу. **Но у нас нерегулярная решетка:**



Давайте разобьем на полосы,
т.о. сведя в каждой полосе задачу к динамическому программированию.

3) Реализовать быстрый **MRF** оптимизатор для задачи текстурирования (идея)

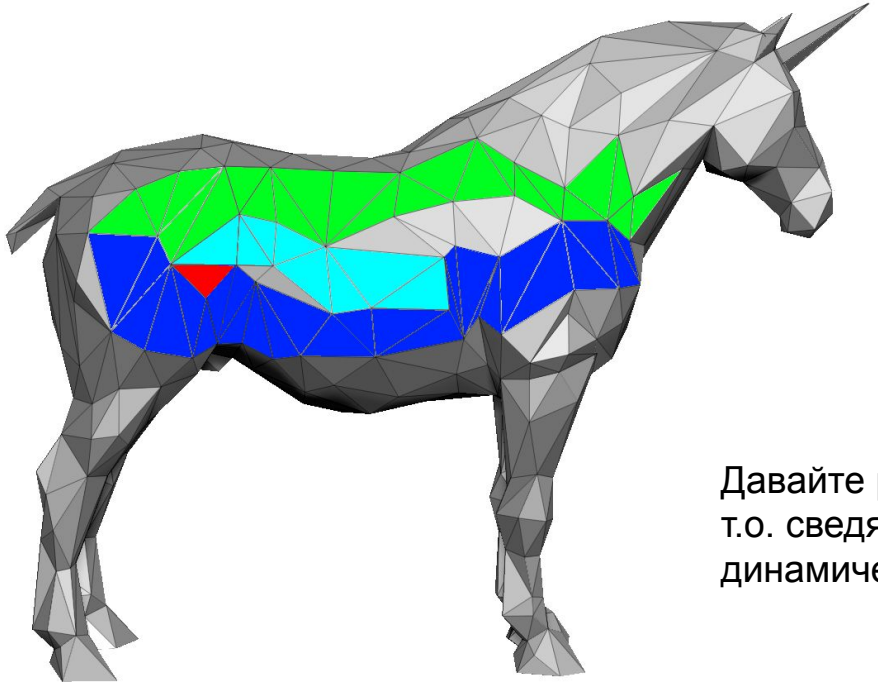
Можно решать через alpha-expansion, но это жадина + не параллелится.
Есть статья быстро решающая задачу. **Но у нас нерегулярная решетка:**



Давайте разобьем на полосы,
т.о. сведя в каждой полосе задачу к динамическому программированию.

3) Реализовать быстрый **MRF** оптимизатор для задачи текстурирования (идея)

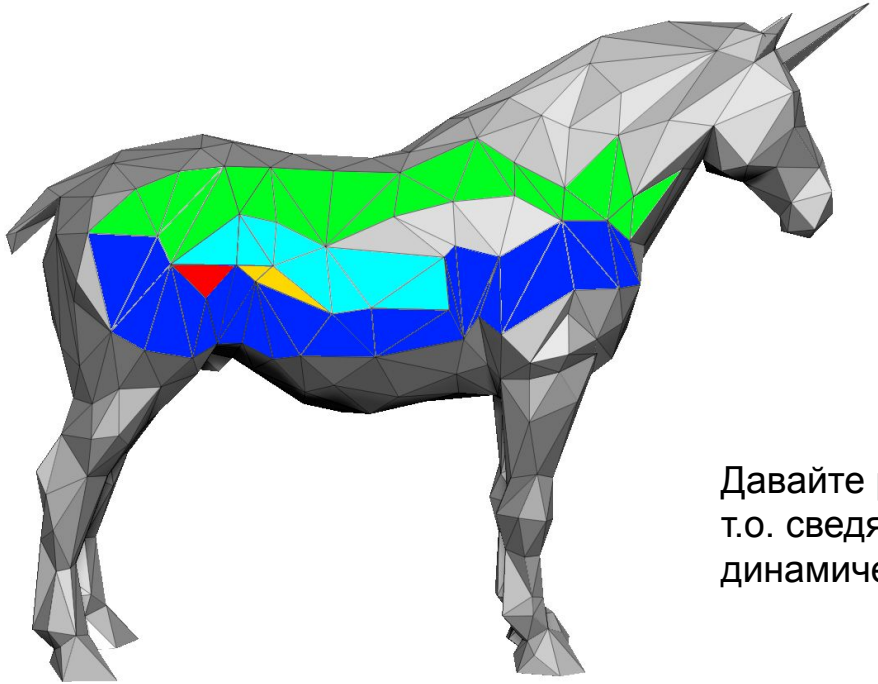
Можно решать через alpha-expansion, но это жадина + не параллелится.
Есть статья быстро решающая задачу. **Но у нас нерегулярная решетка:**



Давайте разобьем на полосы,
т.о. сведя в каждой полосе задачу к динамическому программированию.

3) Реализовать быстрый **MRF** оптимизатор для задачи текстурирования (идея)

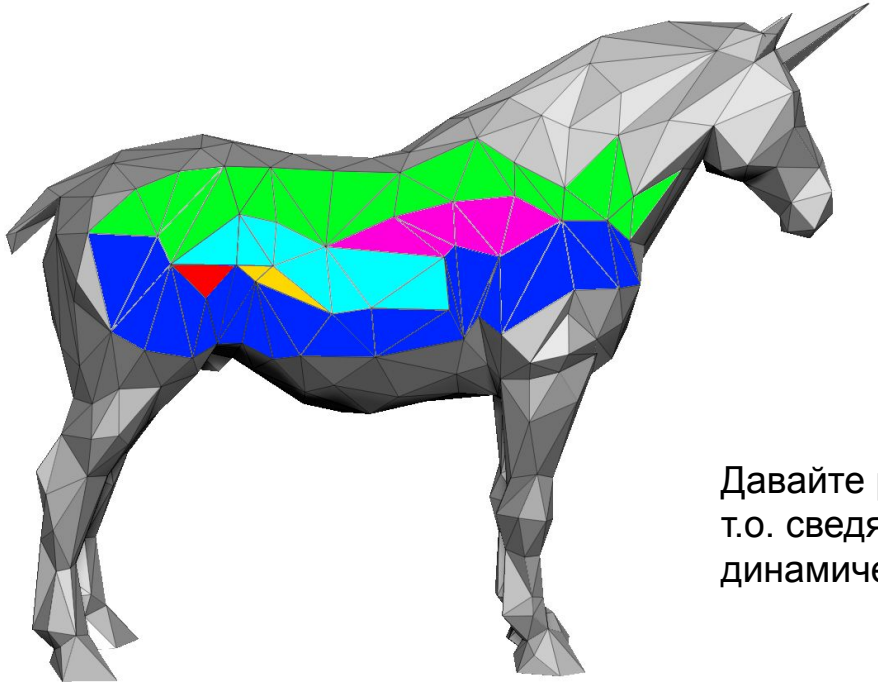
Можно решать через alpha-expansion, но это жадина + не параллелится.
Есть статья быстро решающая задачу. **Но у нас нерегулярная решетка:**



Давайте разобьем на полосы,
т.о. сведя в каждой полосе задачу к динамическому программированию.

3) Реализовать быстрый **MRF** оптимизатор для задачи текстурирования (идея)

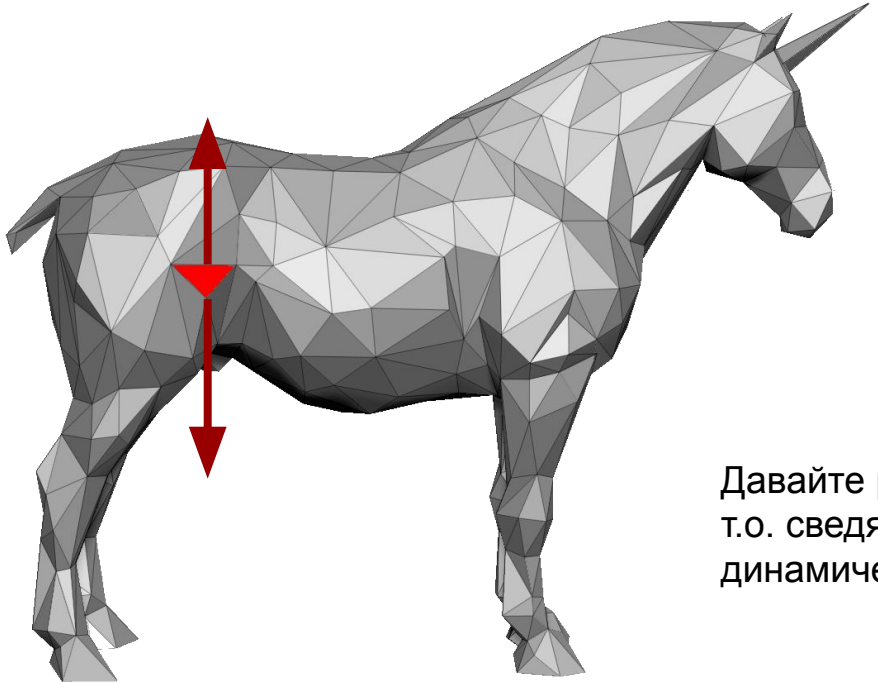
Можно решать через alpha-expansion, но это жадина + не параллелится.
Есть статья быстро решающая задачу. **Но у нас нерегулярная решетка:**



Давайте разобьем на полосы,
т.о. сведя в каждой полосе задачу к динамическому программированию.

3) Реализовать быстрый **MRF** оптимизатор для задачи текстурирования (идея)

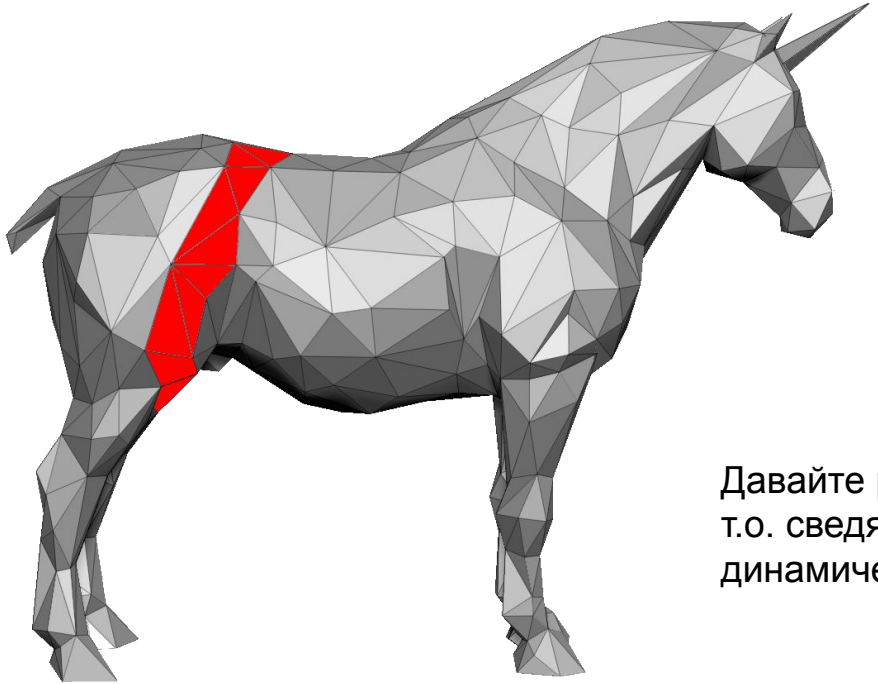
Можно решать через alpha-expansion, но это жадина + не параллелится.
Есть статья быстро решающая задачу. **Но у нас нерегулярная решетка:**



Давайте разобьем на полосы,
т.о. сведя в каждой полосе задачу к динамическому программированию.

3) Реализовать быстрый **MRF** оптимизатор для задачи текстурирования (идея)

Можно решать через alpha-expansion, но это жадина + не параллелится.
Есть статья быстро решающая задачу. **Но у нас нерегулярная решетка:**

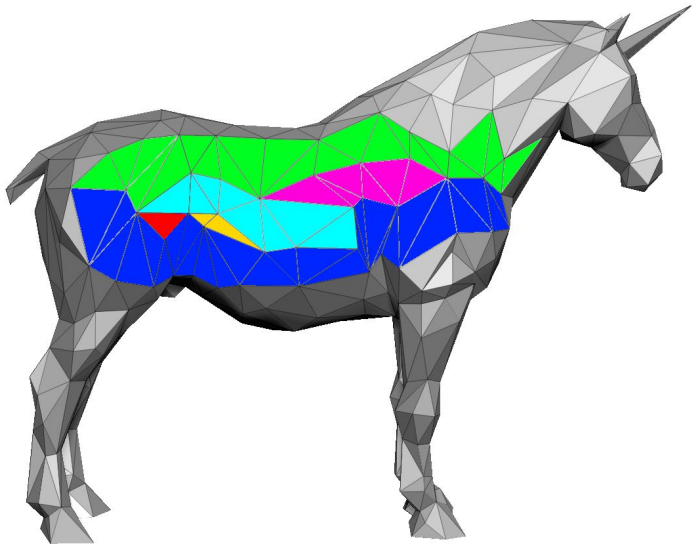


Давайте разобьем на полосы,
т.о. сведя в каждой полосе задачу к динамическому программированию.

3) Реализовать быстрый **MRF** оптимизатор для задачи текстурирования (идея)

Можно решать через alpha-expansion, но это жадина + не параллелится.
Есть статья быстро решающая задачу.

1) Реализовать алгоритм разбиения поверхности на полоски



3) Реализовать быстрый **MRF** оптимизатор для задачи текстурирования (идея)

Можно решать через alpha-expansion, но это жадина + не параллелится.
Есть статья быстро решающая задачу.

1) Реализовать алгоритм разбиения поверхности на полоски
2) Адаптировать подход из статьи на эти полоски

3) Реализовать быстрый **MRF** оптимизатор для задачи текстурирования (идея)

Можно решать через alpha-expansion, но это жадина + не параллелится.
Есть статья быстро решающая задачу.

1) Реализовать алгоритм разбиения поверхности на полоски
2) Адаптировать подход из статьи на эти полоски
3) Сравниться с **alpha-expansion**

3) Реализовать быстрый **MRF** оптимизатор для задачи текстурирования (идея)

Можно решать через alpha-expansion, но это жадина + не параллелится.
Есть статья быстро решающая задачу.

1) Реализовать алгоритм разбиения поверхности на полоски
2) Адаптировать подход из статьи на эти полоски
3) Сравниться с **alpha-expansion**
4) Ускорять и оптимизировать на огромных случаях (потенциально на GPU)

Вопросы?

3) Реализовать быстрый **MRF** оптимизатор для задачи текстурирования (идея)

Можно решать через alpha-expansion, но это жадина + не параллелится.
Есть статья быстро решающая задачу.

1) Реализовать алгоритм разбиения поверхности на полоски
2) Адаптировать подход из статьи на эти полоски
3) Сравниться с **alpha-expansion**
4) Ускорять и оптимизировать на огромных случаях (потенциально на GPU)

4) Упаковка текстурного атласа (математика/геометрия/олимп. прогр.)

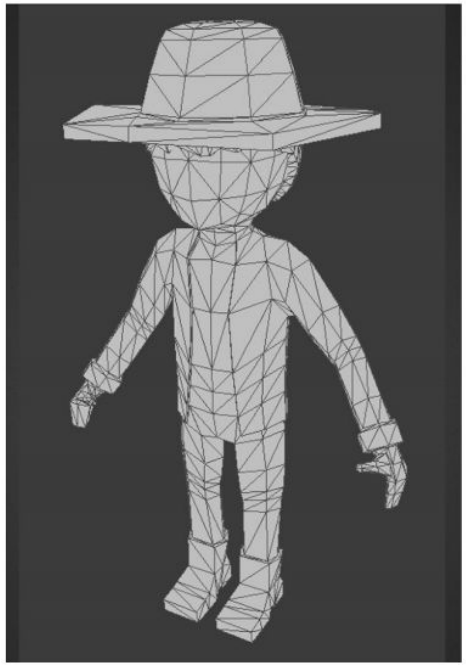
3D модель

Текстурный атлас

Текстурированная 3D модель

4) Упаковка текстурного атласа (математика/геометрия/олимп. прогр.)

3D модель

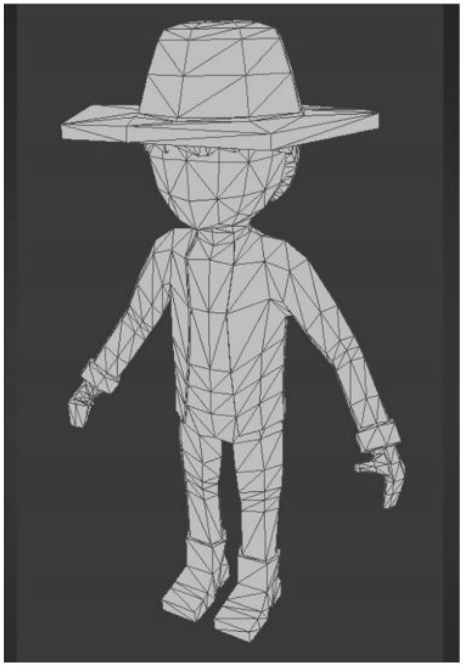


Нарезка на кусочки
(параметризация чартов)

Текстурный атлас

4) Упаковка текстурного атласа (математика/геометрия/олимп. прогр.)

3D модель



Нарезка на кусочки
(параметризация чартов)

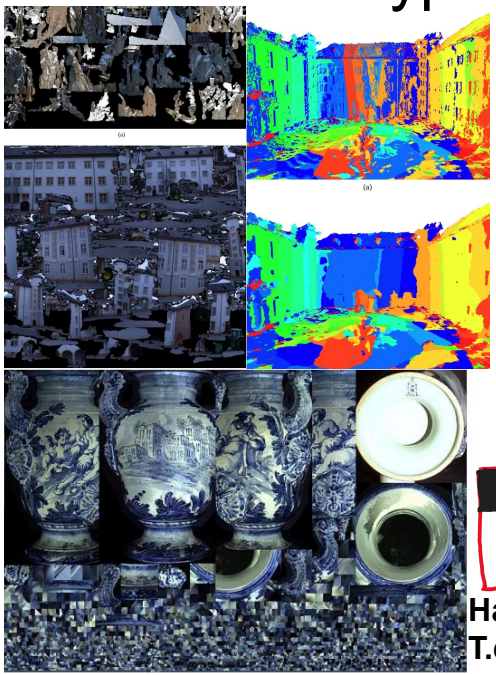
Плотное расположение
кусочков в атласе
(упаковка чартов)

Текстурный атлас

4) Упаковка текстурного атласа (математика/геометрия/олимп. прогр.)

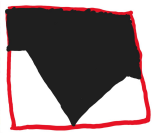
~~3D модель~~

Natural текстурный атлас



~~Нарезка на кусочки (параметризация чартов)~~

Плотное расположение кусочков в атласе (упаковка чартов)



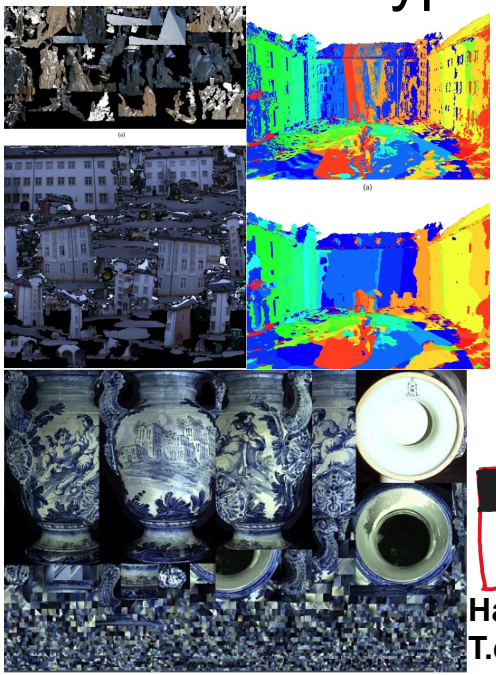
На базе высокочастотных весов
Т.е. на базе карты “победитель ли я?”

Текстурный атлас

4) Упаковка текстурного атласа (математика/геометрия/олимп. прогр.)

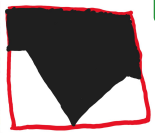
~~3D модель~~

Natural текстурный атлас



~~Нарезка на кусочки (параметризация чартов)~~

Плотное расположение кусочков в атласе (упаковка чартов)



На базе высокочастотных весов
Т.е. на базе карты “победитель ли я?”

Текстурный атлас

4) Упаковка текстурного атласа (математика/геометрия/олимп. прогр.)

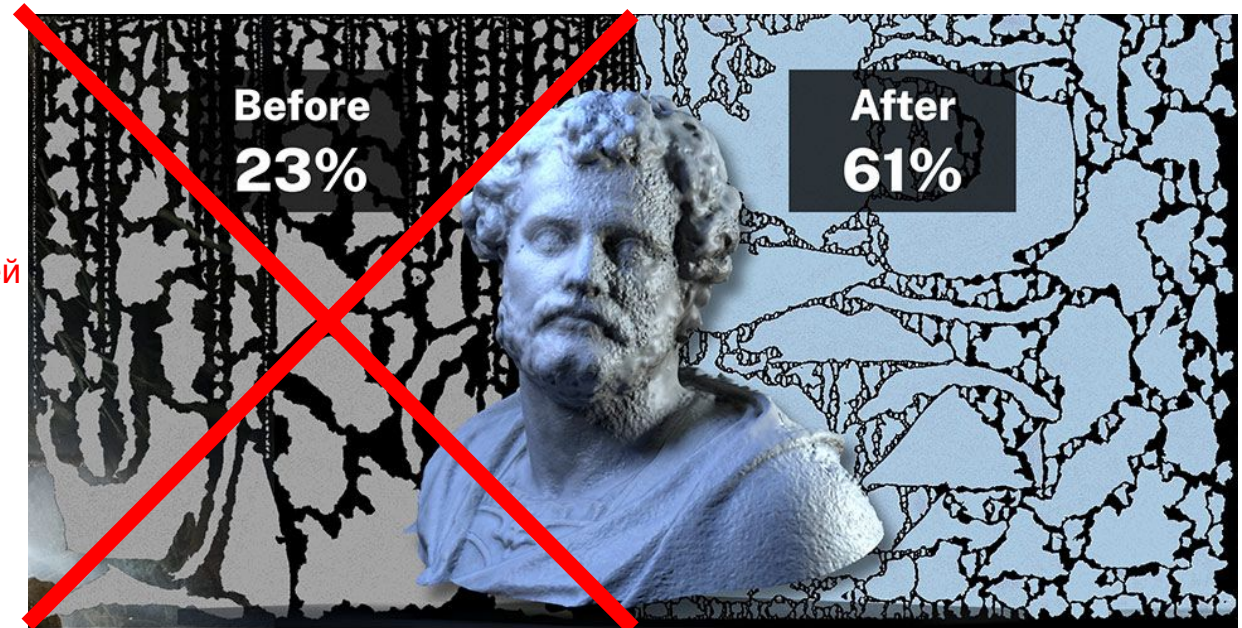
На вход: множество чартов (2D полигонов)

4) Упаковка текстурного атласа (математика/геометрия/олимп. прогр.)

На вход: множество чартов (2D полигонов)

На выход: как можно более плотная упаковка их в текстурном атласе (чтобы было достаточно как можно меньше текстурных страниц)

Много пустых щелей

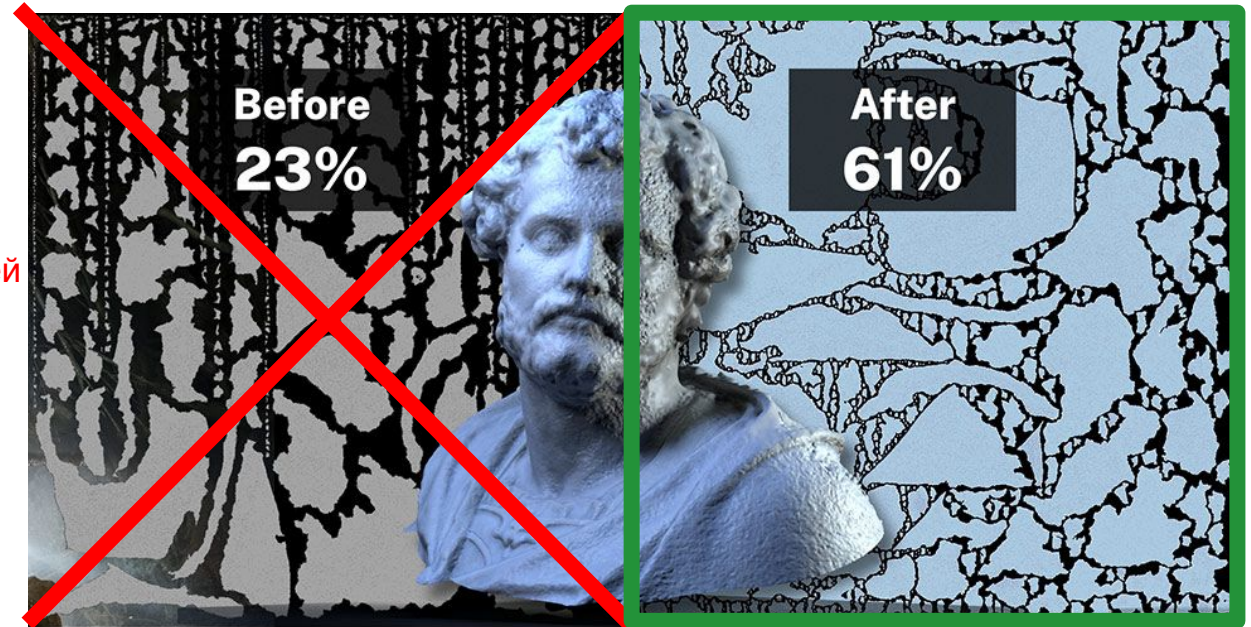


4) Упаковка текстурного атласа (математика/геометрия/олимп. прогр.)

На вход: множество чартов (2D полигонов)

На выход: как можно более плотная упаковка их в текстурном атласе (чтобы было достаточно как можно меньше текстурных страниц)

Много пустых щелей



Хорошая плотная упаковка

4) Упаковка текстурного атласа (математика/геометрия/олимп. прогр.)

На вход: множество чартов (2D полигонов)

На выход: как можно более плотная упаковка их в текстурном атласе (чтобы было достаточно как можно меньше текстурных страниц)

План:

1) Запихивать сначала большие, затем маленькие (жадина)

4) Упаковка текстурного атласа (математика/геометрия/олимп. прогр.)

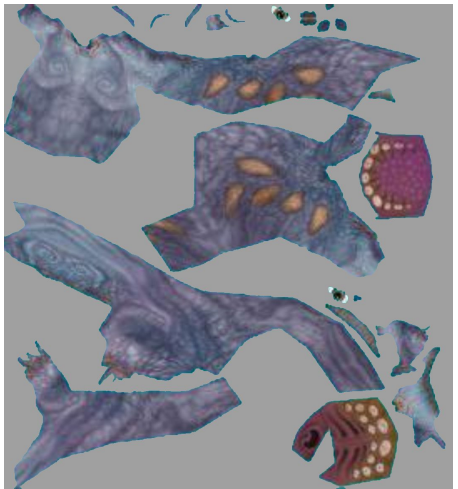
На вход: множество чартов (2D полигонов)

На выход: как можно более плотная упаковка их в текстурном атласе (чтобы было достаточно как можно меньше текстурных страниц)

План:

1) Запихивать сначала большие, затем маленькие (жадина)

2) Обнаружать “пустые зоны”:



4) Упаковка текстурного атласа (математика/геометрия/олимп. прогр.)

На вход: множество чартов (2D полигонов)

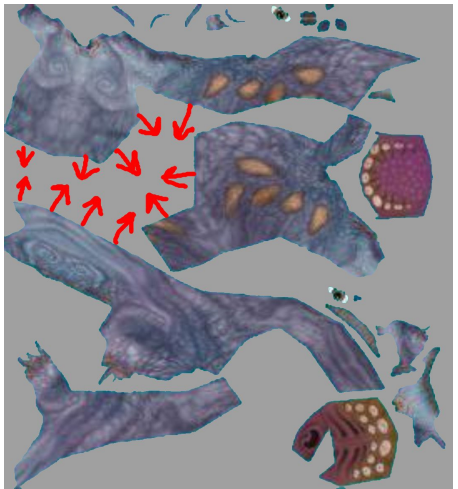
На выход: как можно более плотная упаковка их в текстурном атласе (чтобы было достаточно как можно меньше текстурных страниц)

План:

1) Запихивать сначала большие, затем маленькие (жадина)

2) Обнаружать “пустые зоны”:

2.1) Нашли расстояние до края чарта



4) Упаковка текстурного атласа (математика/геометрия/олимп. прогр.)

На вход: множество чартов (2D полигонов)

На выход: как можно более плотная упаковка их в текстурном атласе (чтобы было достаточно как можно меньше текстурных страниц)

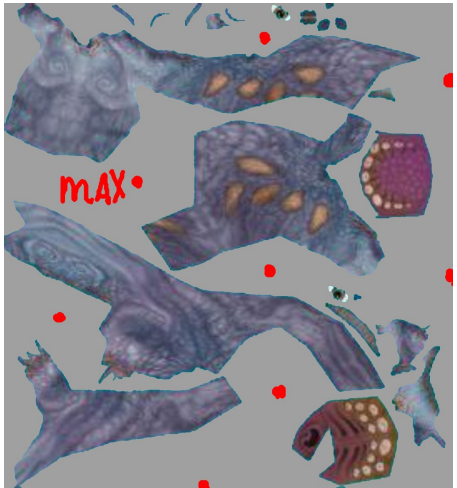
План:

1) Запихивать сначала большие, затем маленькие (жадина)

2) Обнаружать “пустые зоны”:

2.1) Нашли расстояние до края чарта

2.2) Нашли локальные максимумы



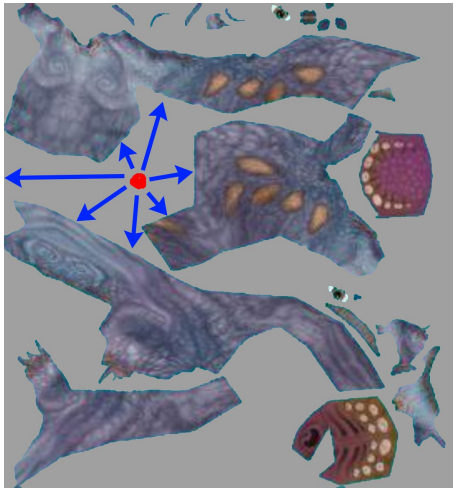
4) Упаковка текстурного атласа (математика/геометрия/олимп. прогр.)

На вход: множество чартов (2D полигонов)

На выход: как можно более плотная упаковка их в текстурном атласе (чтобы было достаточно как можно меньше текстурных страниц)

План:

1) Запихивать сначала большие, затем маленькие (жадина)
2) Обнаружать “пустые зоны”:
2.1) Нашли расстояние до края чарта
2.2) Нашли локальные максимумы
2.3) Пуляем лучи-видимости чтобы найти границы зоны



4) Упаковка текстурного атласа (математика/геометрия/олимп. прогр.)

На вход: множество чартов (2D полигонов)

На выход: как можно более плотная упаковка их в текстурном атласе (чтобы было достаточно как можно меньше текстурных страниц)

План:

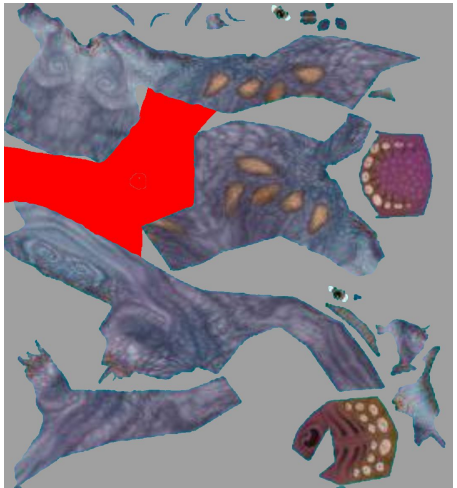
1) Запихивать сначала большие, затем маленькие (жадина)

2) Обнаружать “пустые зоны”:

2.1) Нашли расстояние до края чарта

2.2) Нашли локальные максимумы

2.3) Пуляем лучи-видимости чтобы найти границы зоны



4) Упаковка текстурного атласа (математика/геометрия/олимп. прогр.)

На вход: множество чартов (2D полигонов)

На выход: как можно более плотная упаковка их в текстурном атласе (чтобы было достаточно как можно меньше текстурных страниц)

План:

1) Запихивать сначала большие, затем маленькие (жадина)
2) Обнаружать “пустые зоны”
3) Запихивать в пустые зоны все так же по убыванию размера

4) Упаковка текстурного атласа (математика/геометрия/олимп. прогр.)

На вход: множество чартов (2D полигонов)

На выход: как можно более плотная упаковка их в текстурном атласе (чтобы было достаточно как можно меньше текстурных страниц)

План:

1) Запихивать сначала большие, затем маленькие (жадина)
2) Обнаружать “пустые зоны”
3) Запихивать в пустые зоны все так же по убыванию размера
4) Пробовать вращать каждый чарт (4 варианта: 0, 90, 180, 360 градусов)

4) Упаковка текстурного атласа (математика/геометрия/олимп. прогр.)

На вход: множество чартов (2D полигонов)

На выход: как можно более плотная упаковка их в текстурном атласе (чтобы было достаточно как можно меньше текстурных страниц)

План:

1) Запихивать сначала большие, затем маленькие (жадина)
2) Обнаружать “пустые зоны”
3) Запихивать в пустые зоны все так же по убыванию размера
4) Пробовать вращать каждый чарт (4 варианта: 0, 90, 180, 360 градусов)
5) Распараллелить и ускорить (coarse-to-fine)

4) Упаковка текстурного атласа (математика/геометрия/олимп. прогр.)

На вход: множество чартов (2D полигонов)

На выход: как можно более плотная упаковка их в текстурном атласе (чтобы было достаточно как можно меньше текстурных страниц)

План:

1) Запихивать сначала большие, затем маленькие (жадина)
2) Обнаружать “пустые зоны”
3) Запихивать в пустые зоны все так же по убыванию размера
4) Пробовать вращать каждый чарт (4 варианта: 0, 90, 180, 360 градусов)
5) Распараллелить и ускорить (coarse-to-fine)

Ссылки: статья упаковки как в тетрисе + https://github.com/jpcy/xatlas

Вопросы?

4) Упаковка текстурного атласа (математика/геометрия/олимп. прогр.)

На вход: множество чартов (2D полигонов)

На выход: как можно более плотная упаковка их в текстурном атласе (чтобы было достаточно как можно меньше текстурных страниц)

План:

1) Запихивать сначала большие, затем маленькие (жадина)
2) Обнаружать “пустые зоны”
3) Запихивать в пустые зоны все так же по убыванию размера
4) Пробовать вращать каждый чарт (4 варианта: 0, 90, 180, 360 градусов)
5) Распараллелить и ускорить (coarse-to-fine)

Ссылки: статья упаковки как в тетрисе + https://github.com/jpcy/xatlas

5) Воспроизвести, изучить, адаптировать для больших случаев: PermutoSDF

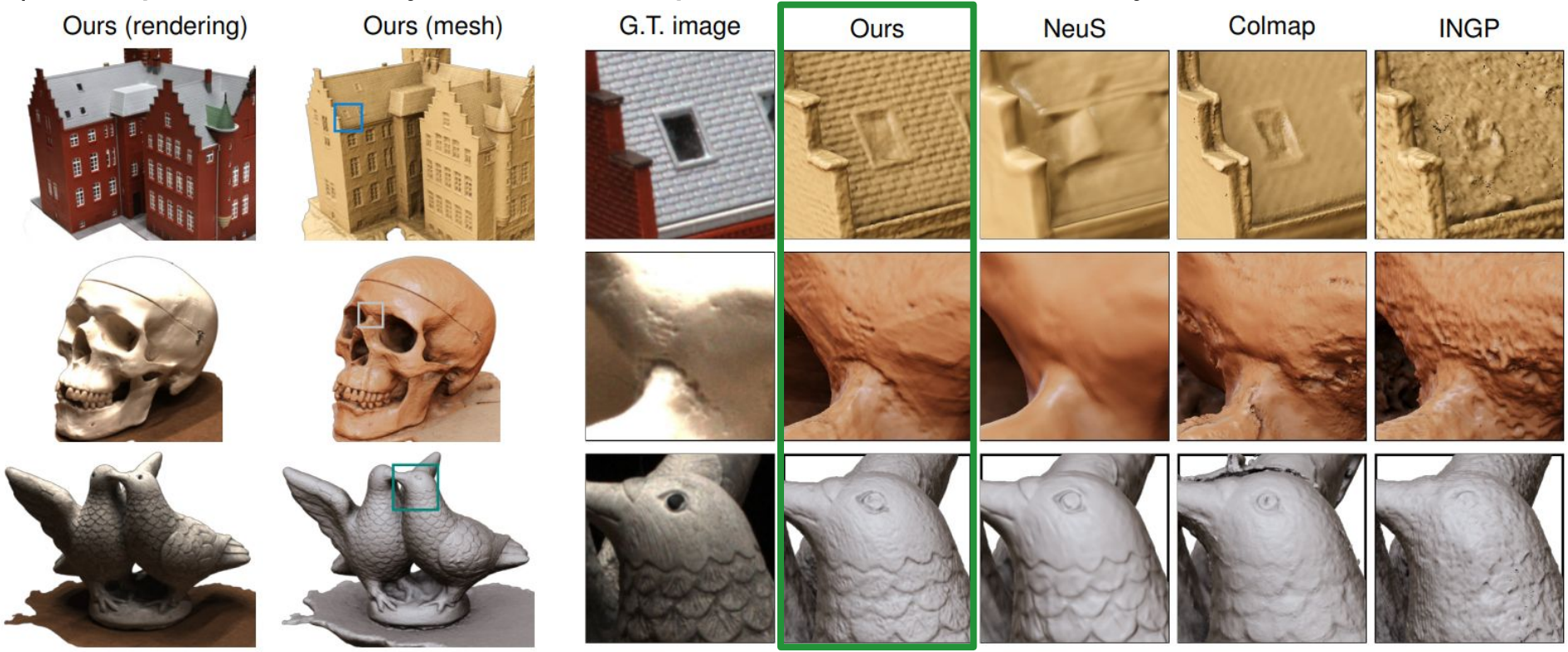


Figure 5. Qualitative comparison of the geometry reconstructed by our method compared to the baselines. Note that our method recovers significantly higher geometrical detail.

5) Воспроизвести, изучить, адаптировать для больших случаев: PermutoSDF

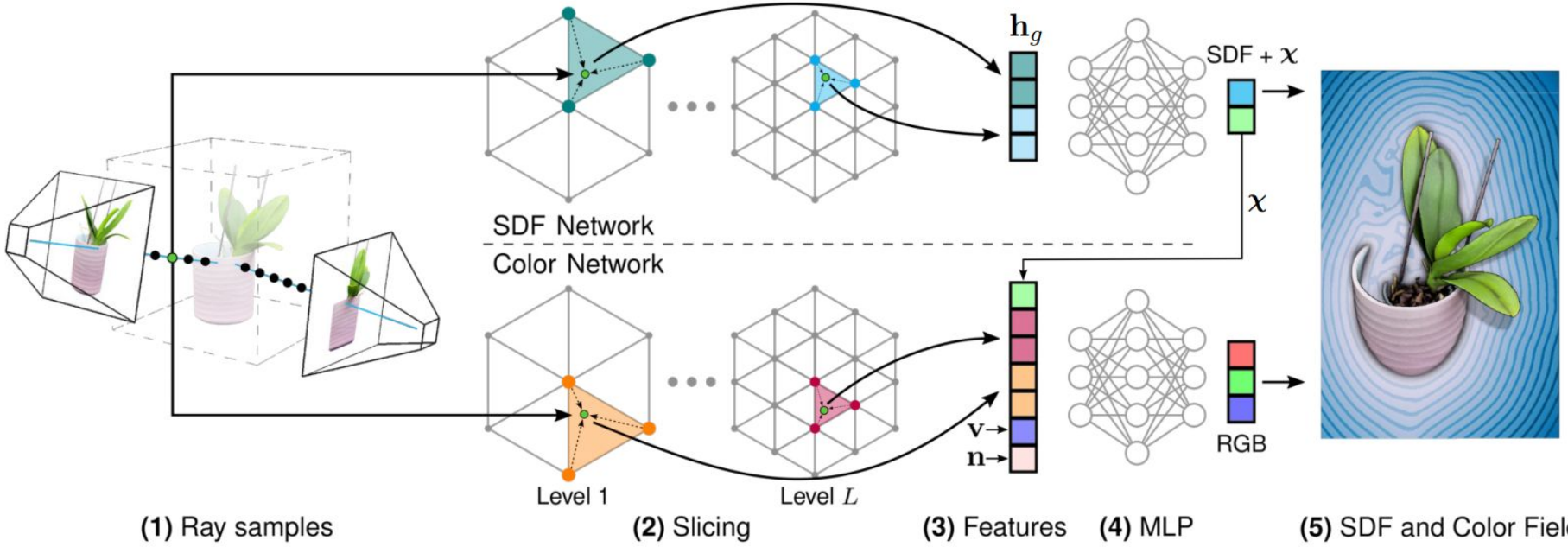


Figure 2. Overview of our PermutoSDF pipeline. **(1)** For a batch of pixels from the posed images, we sample rays inside the volume of interest. **(2)** For each sample, we slice features from a multi-resolution permutohedral lattice. **(3)** The features from all lattice levels are concatenated. For the color network, we also concatenate additional features regarding normal $\mathbf{n}$ of the SDF, view direction $\mathbf{v}$, and learnable features χ from the SDF network. **(4)** Small MLPs decode the SDF and a view-dependent RGB color. **(5)** The output is rendered volumetrically and supervised only with RGB images. We visualize surface color and a 2D slice of the SDF.

5) Воспроизвести, изучить, адаптировать для больших случаев: PermutoSDF

1) Изучить статью и код (есть github)

5) Воспроизвести, изучить, адаптировать для больших случаев: PermutoSDF

1) Изучить статью и код (есть github)
2) Подумать как можно сильно ускорить ценой малой потери качества

5) Воспроизвести, изучить, адаптировать для больших случаев: PermutoSDF

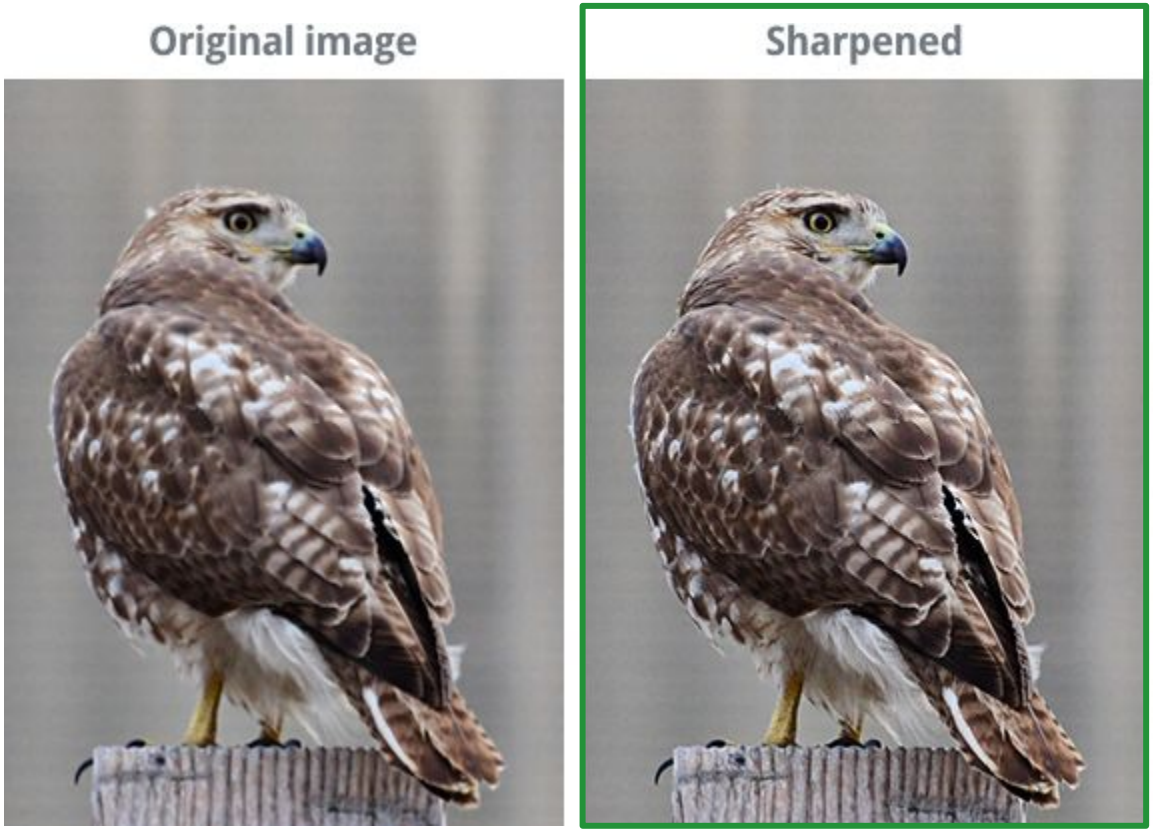
1) Изучить статью и код (есть github)
2) Подумать как можно сильно ускорить ценой малой потери качества
3) Сделать **out-of-core** адаптацию (обработка огромных сцен по кускам)

Вопросы?

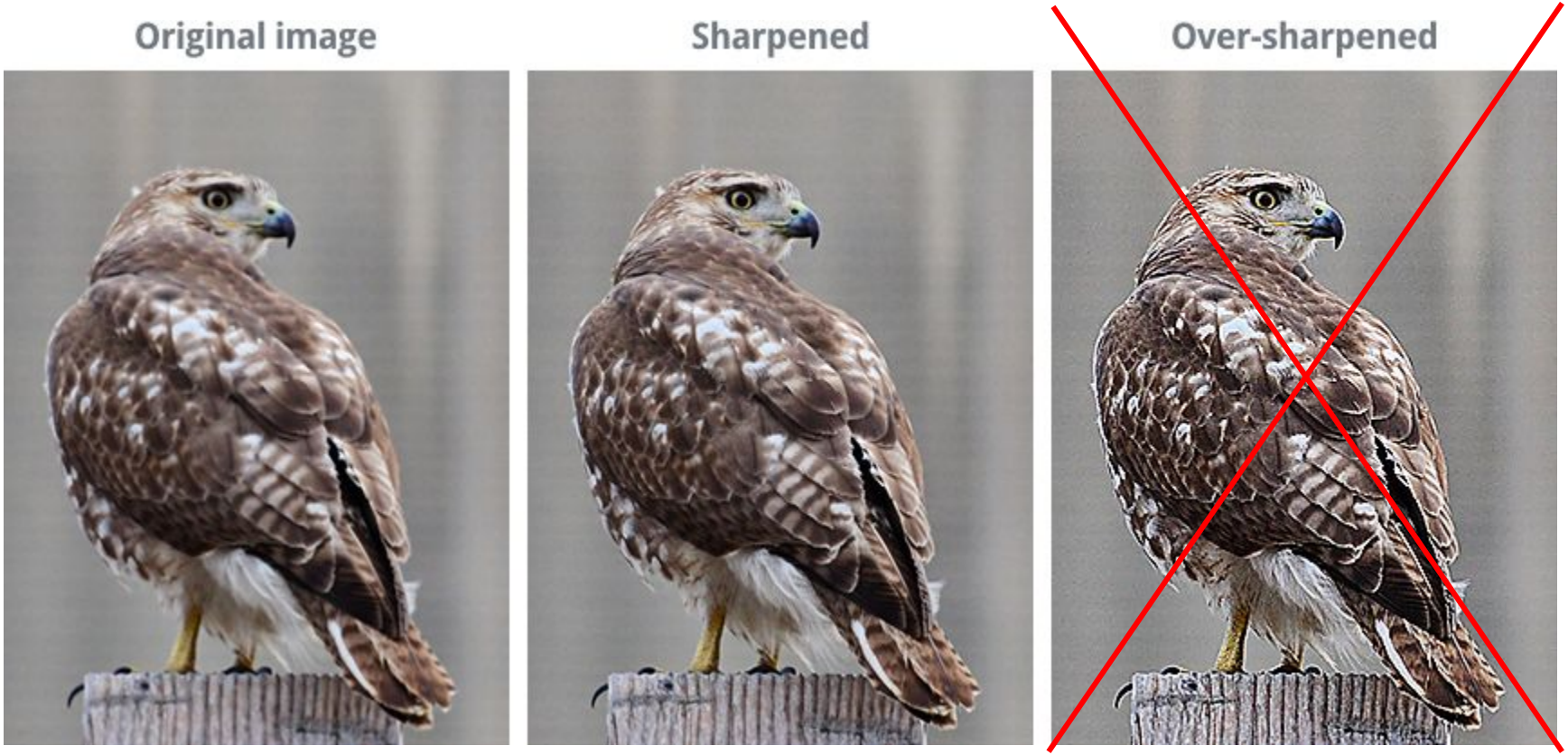
5) Воспроизвести, изучить, адаптировать для больших случаев: PermutoSDF

1) Изучить статью и код (есть github)
2) Подумать как можно сильно ускорить ценой малой потери качества
3) Сделать **out-of-core** адаптацию (обработка огромных сцен по кускам)

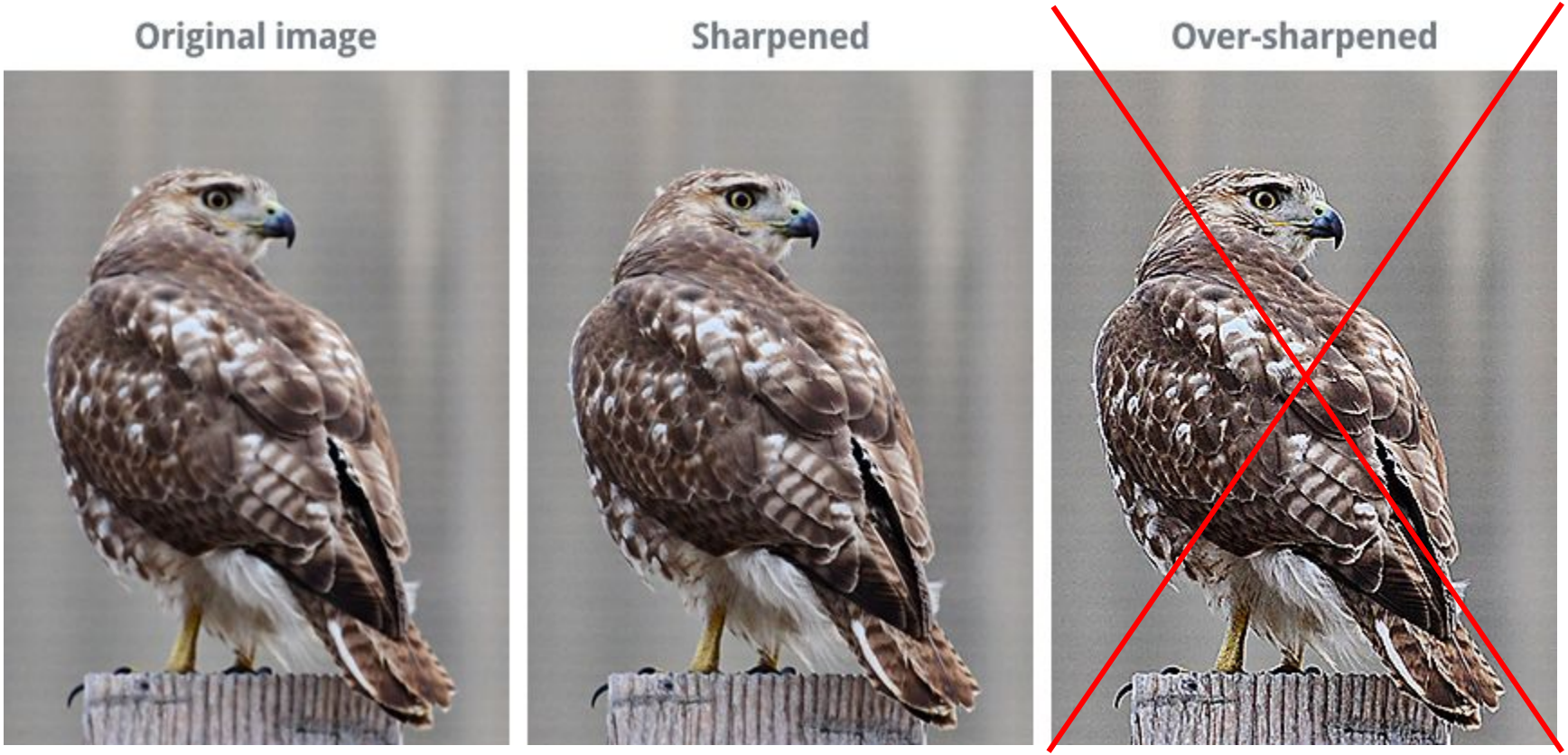
6) Улучшение текстуры - насыщенность/**resharpening**



6) Улучшение текстуры - насыщенность/**resharpening**



6) Улучшение текстуры - насыщенность/**resharpening**



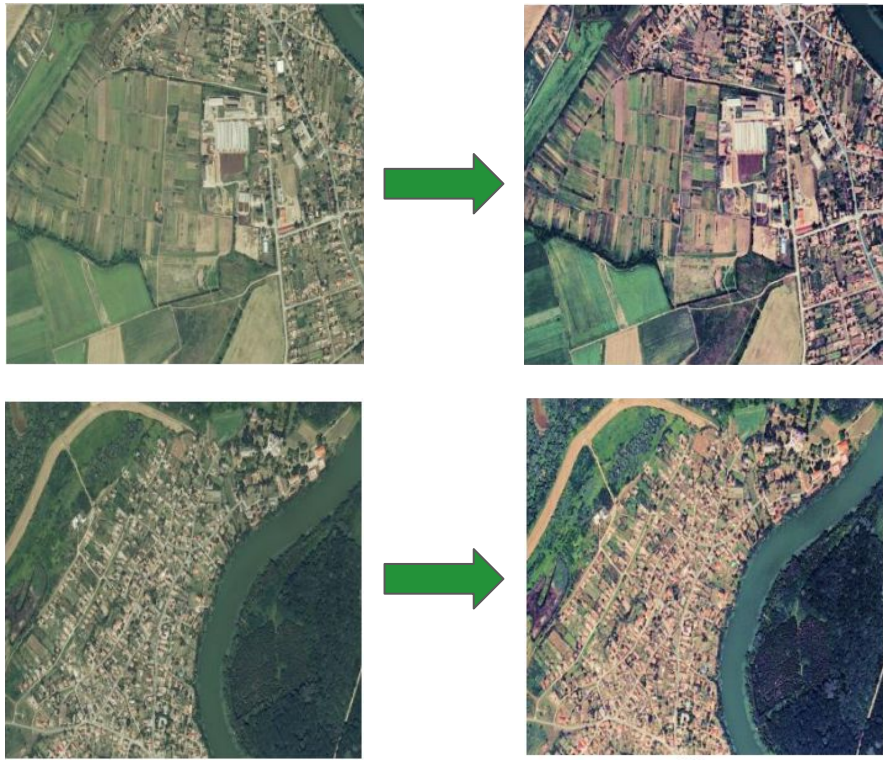
6) Улучшение текстуры - насыщенность/**resharpening**

Radeon Image Sharpening

6) Улучшение текстуры - насыщенность/**resharpening**

Radeon Image Sharpening

6) Улучшение текстуры - насыщенность/resharpening



6) Улучшение текстуры - насыщенность/resharpening

1) Изучить какие есть стандартные практики (в т.ч. у фотографов)

6) Улучшение текстуры - насыщенность/resharpening

1) Изучить какие есть стандартные практики (в т.ч. у фотографов)
2) Найти безопасный метод добавления резкости (чтобы не вредило)

6) Улучшение текстуры - насыщенность/resharpening

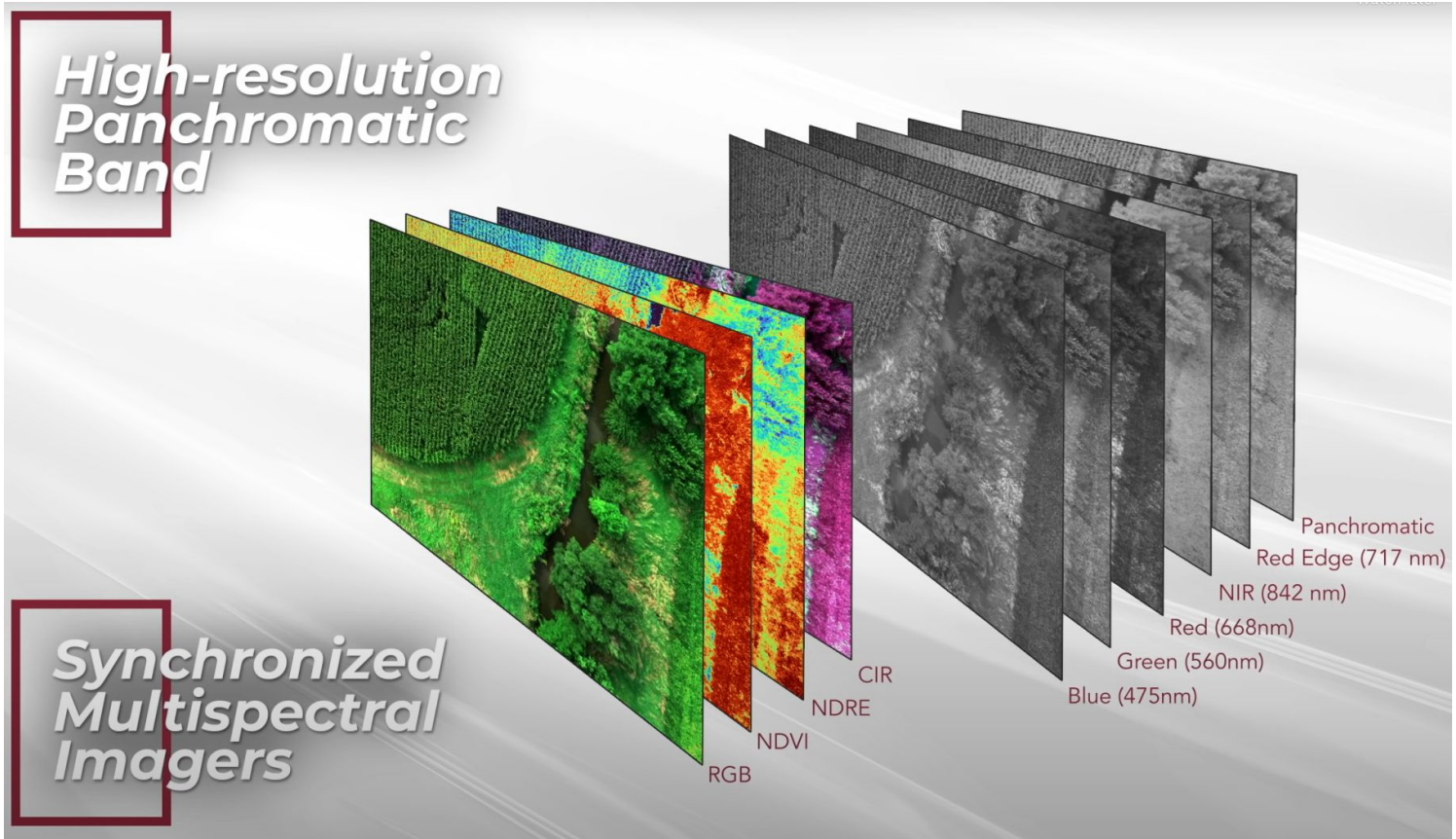
1) Изучить какие есть стандартные практики (в т.ч. у фотографов)
2) Найти безопасный метод добавления резкости (чтобы не вредило)
3) Попробовать улучшить результаты на базе множественных наблюдений

Вопросы?

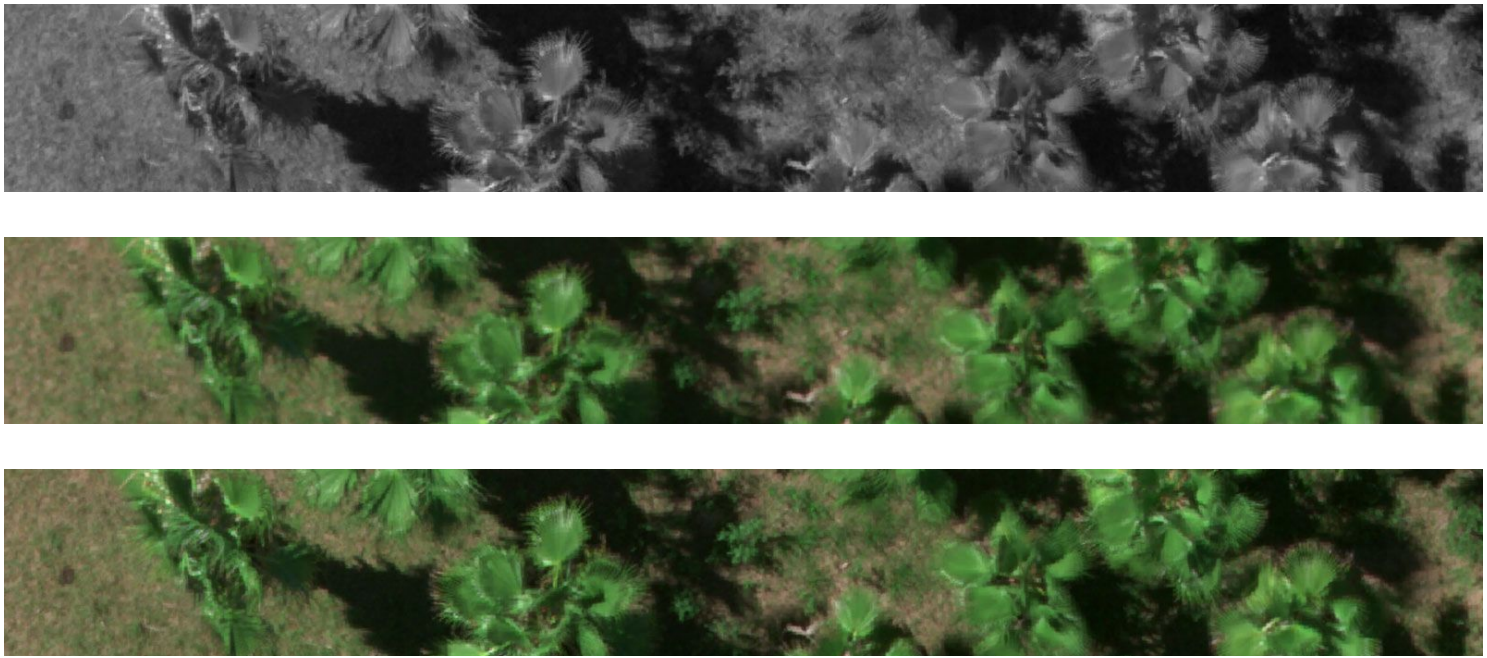
6) Улучшение текстуры - насыщенность/resharpening

1) Изучить какие есть стандартные практики (в т.ч. у фотографов)
2) Найти безопасный метод добавления резкости (чтобы не вредило)
3) Попробовать улучшить результаты на базе множественных наблюдений

7) Super-resolution тепловых снимков и мультиспектральных спутников



7) Super-resolution тепловых снимков и мультиспектральных спутников



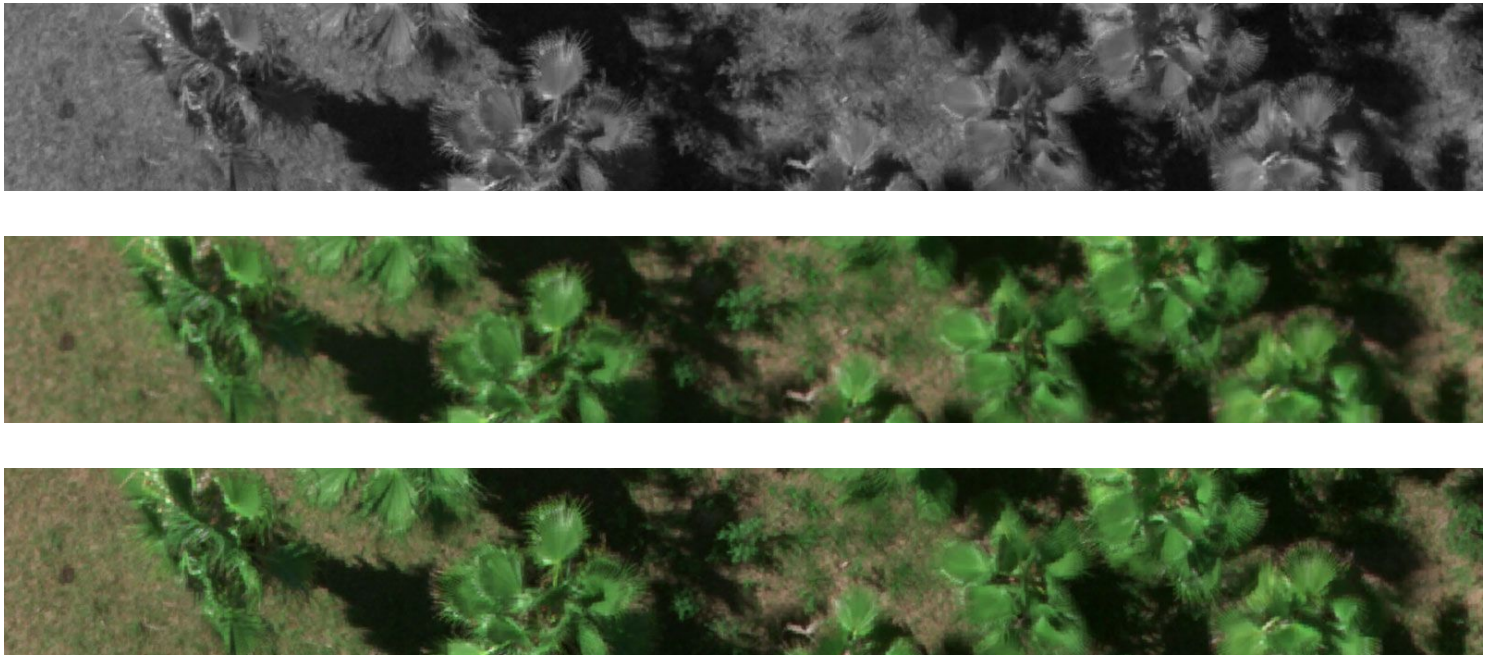
High-res grayscale

Low-res RGB

High-res RGB

Вопросы?

7) Super-resolution тепловых снимков и мультиспектральных спутников



High-res grayscale

+

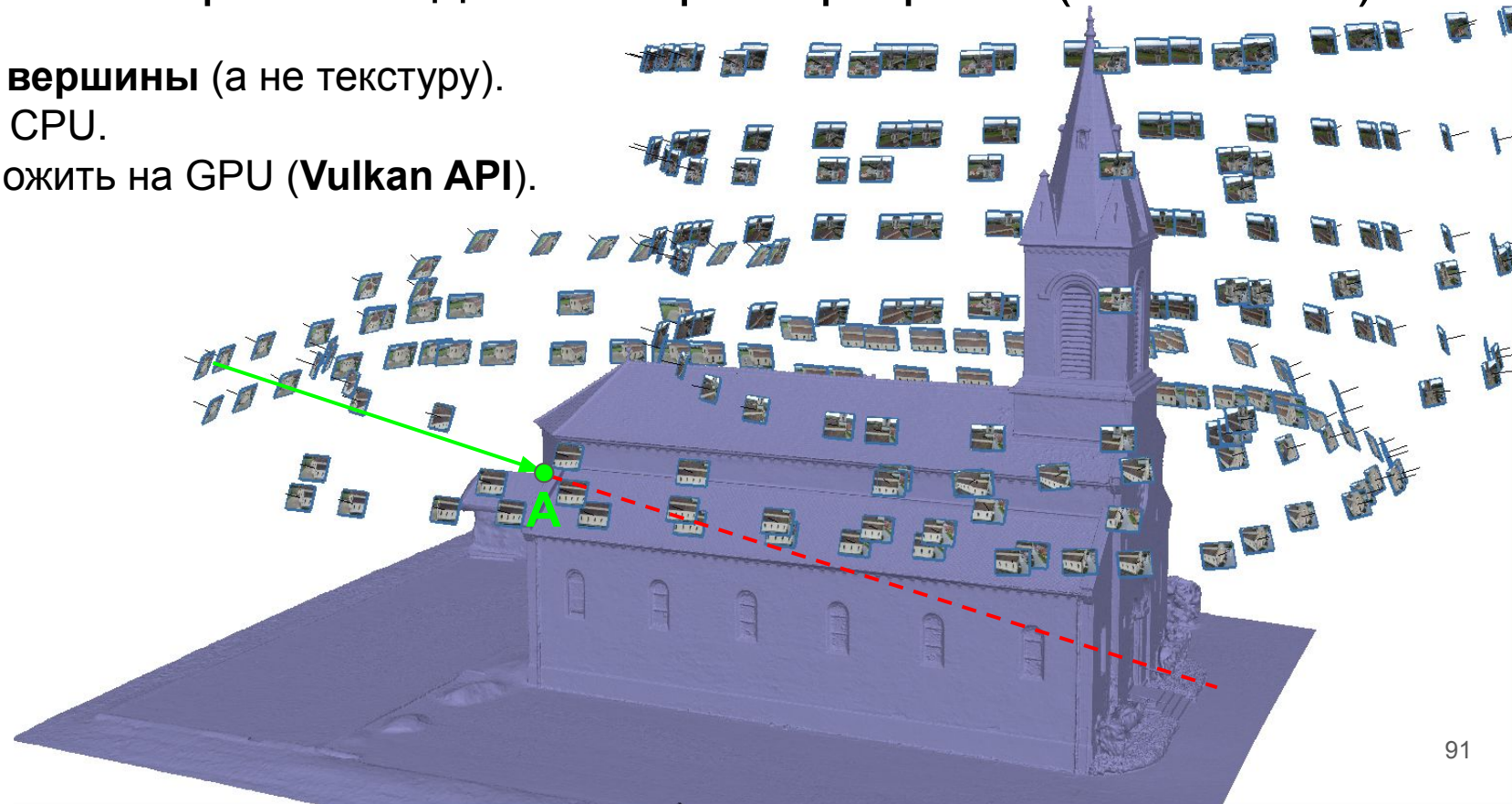
Low-res RGB

=

High-res RGB

8) Раскраска вершин модели по фотографиям (Vulkan API)

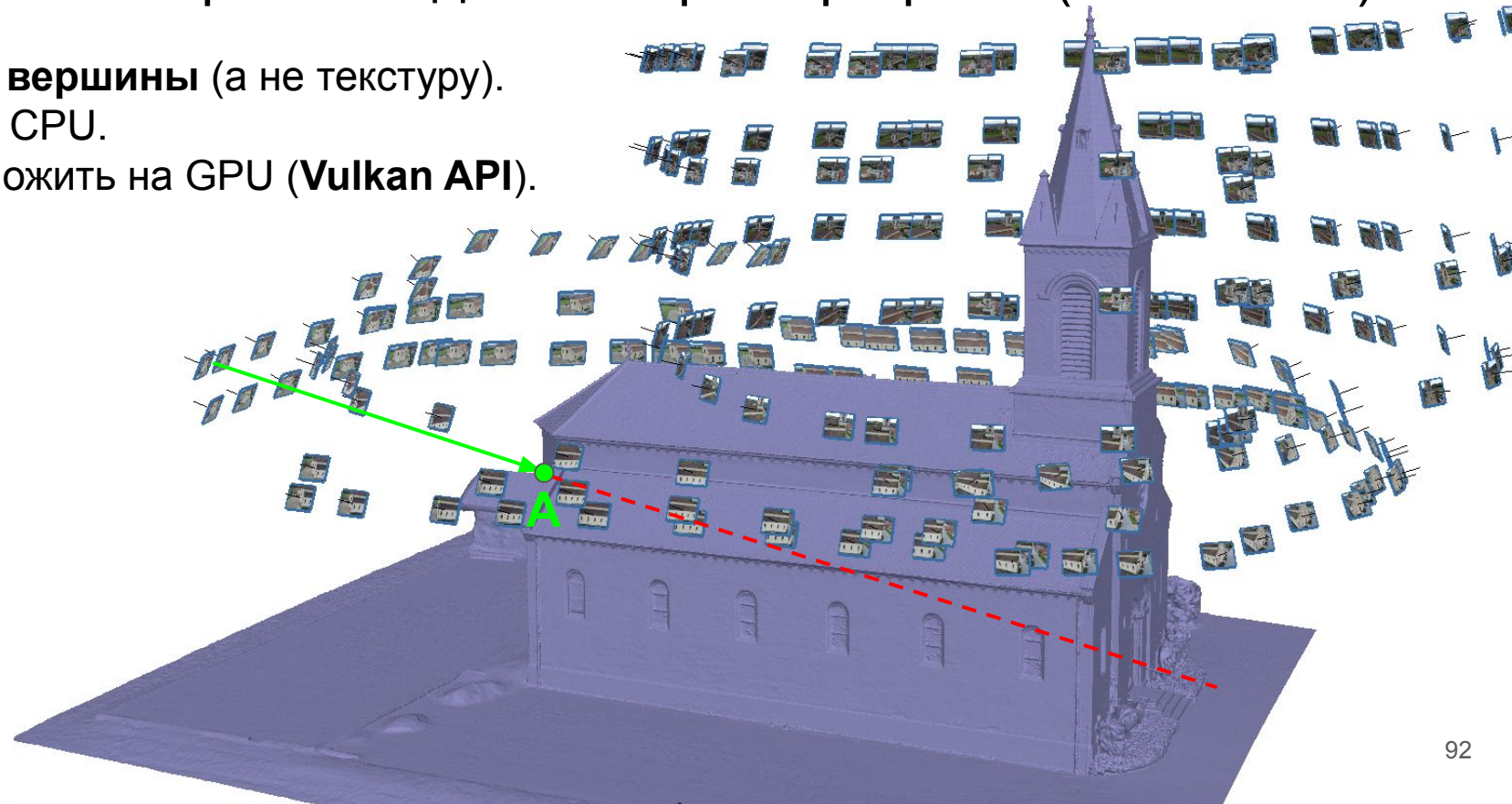
Раскрасить **вершины** (а не текстуру).
Есть код на CPU.
Надо переложить на GPU (**Vulkan API**).



Вопросы?

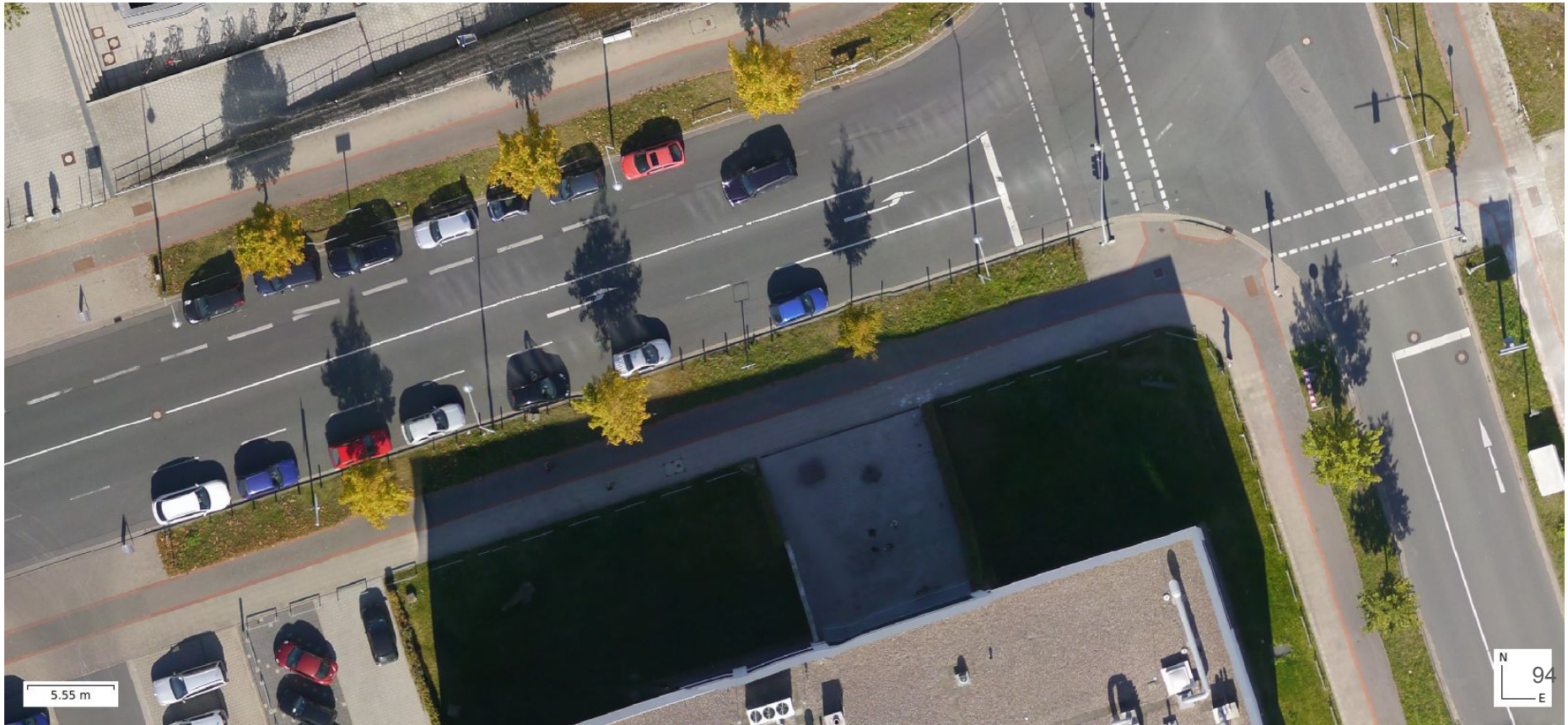
8) Раскраска вершин модели по фотографиям (Vulkan API)

Раскрасить **вершины** (а не текстуру).
Есть код на CPU.
Надо переложить на GPU (**Vulkan API**).

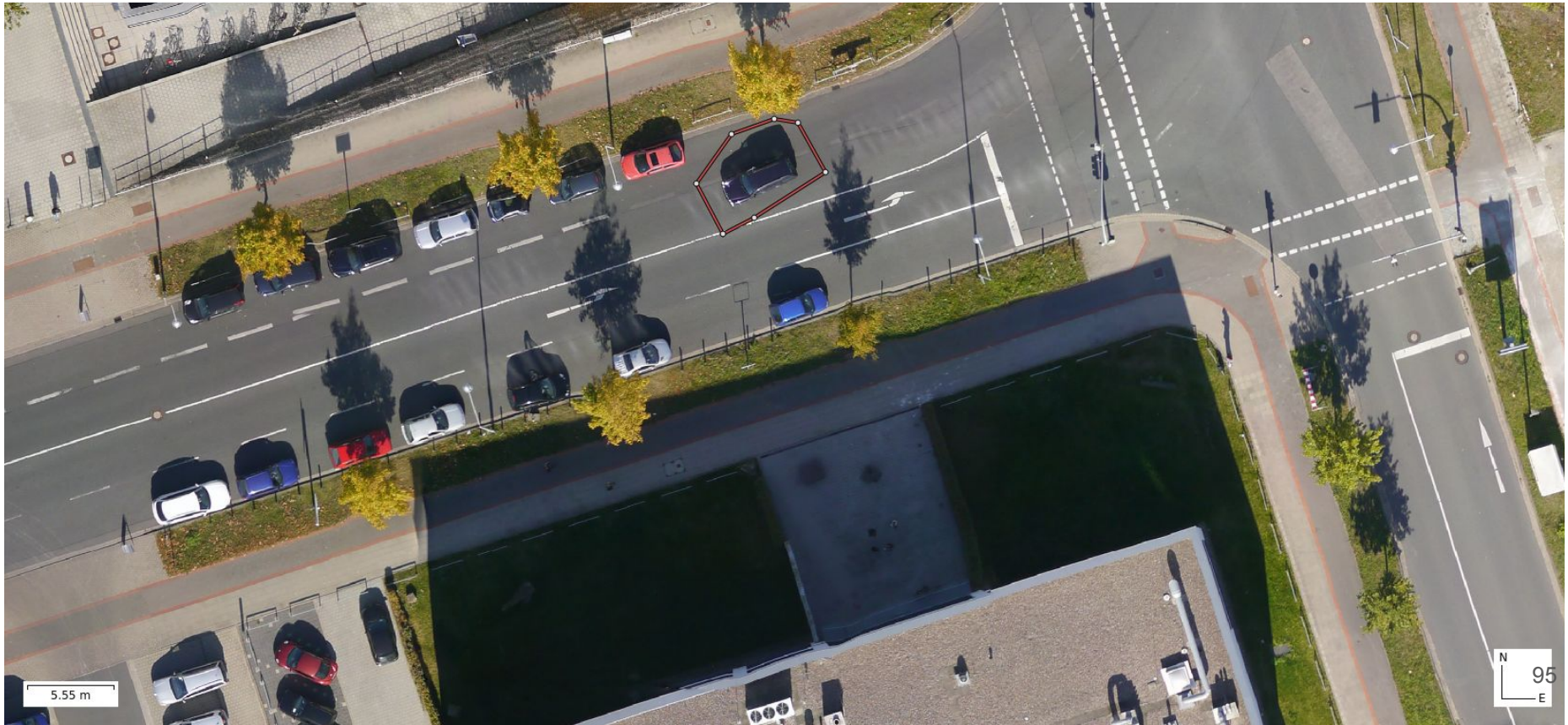


9) Ретуширование ортофотоплана (healing brush)

9) Ретуширование ортофотоплана (healing brush)

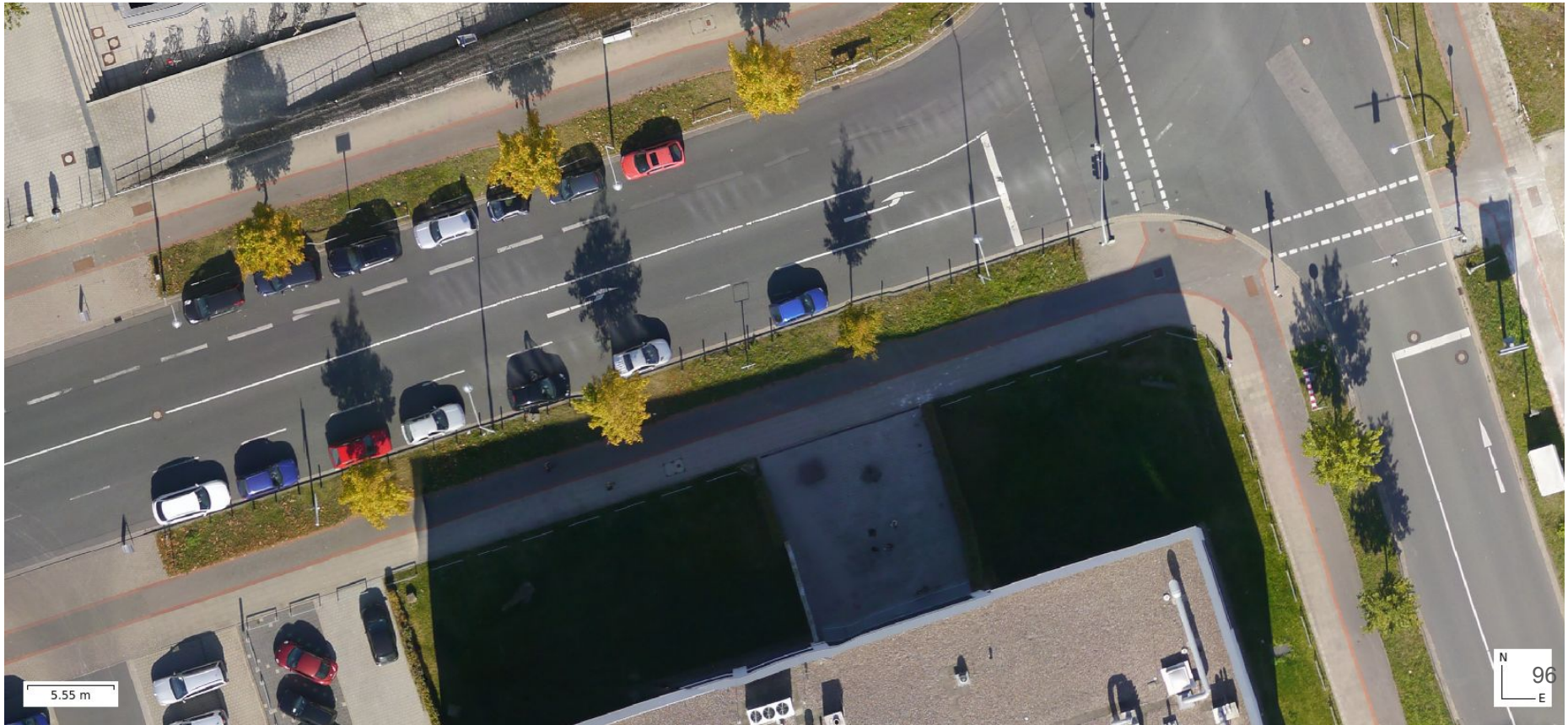


9) Ретуширование ортофотоплана (healing brush)



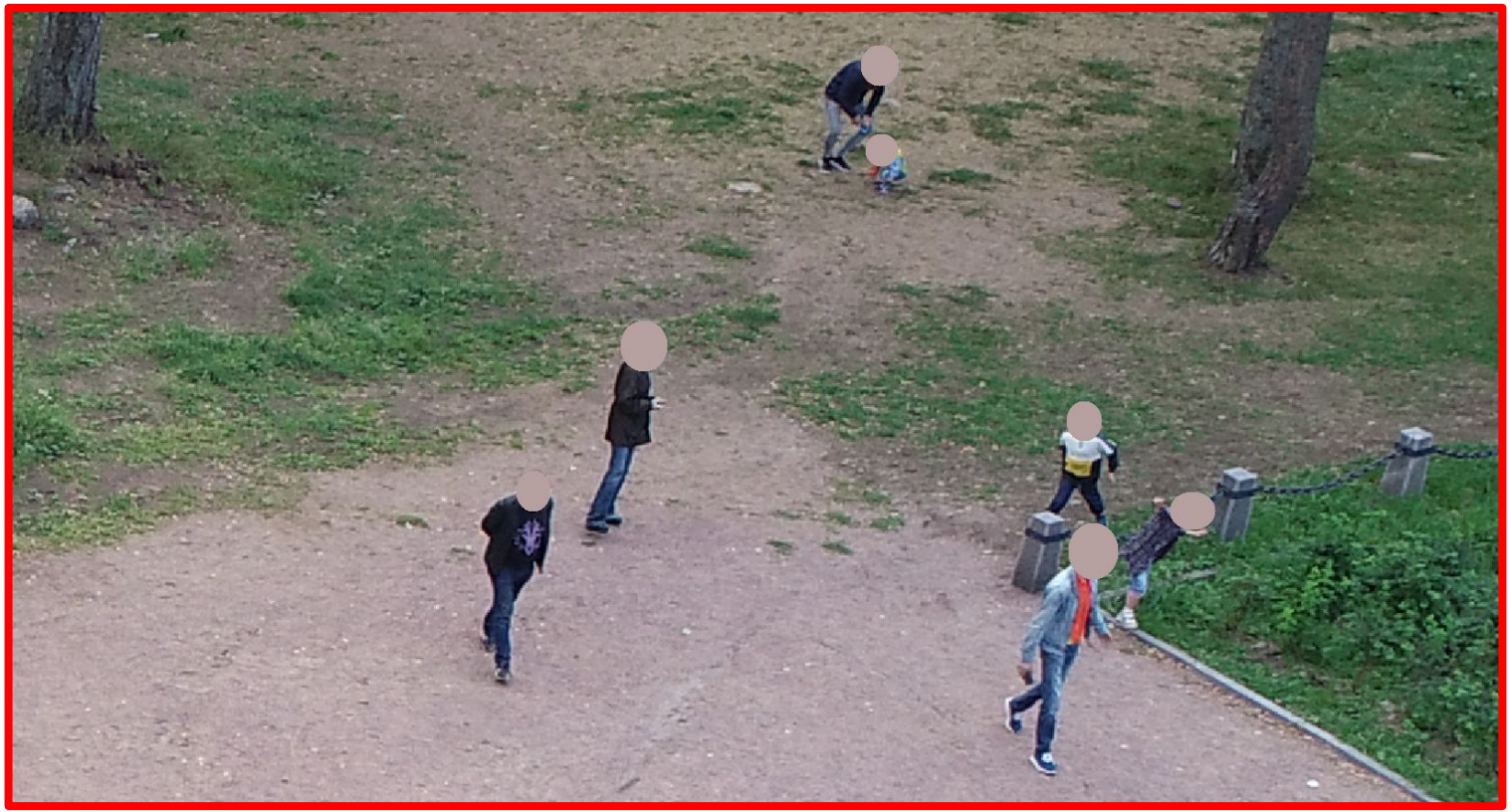
Вопросы?

9) Ретуширование ортофотоплана (healing brush)



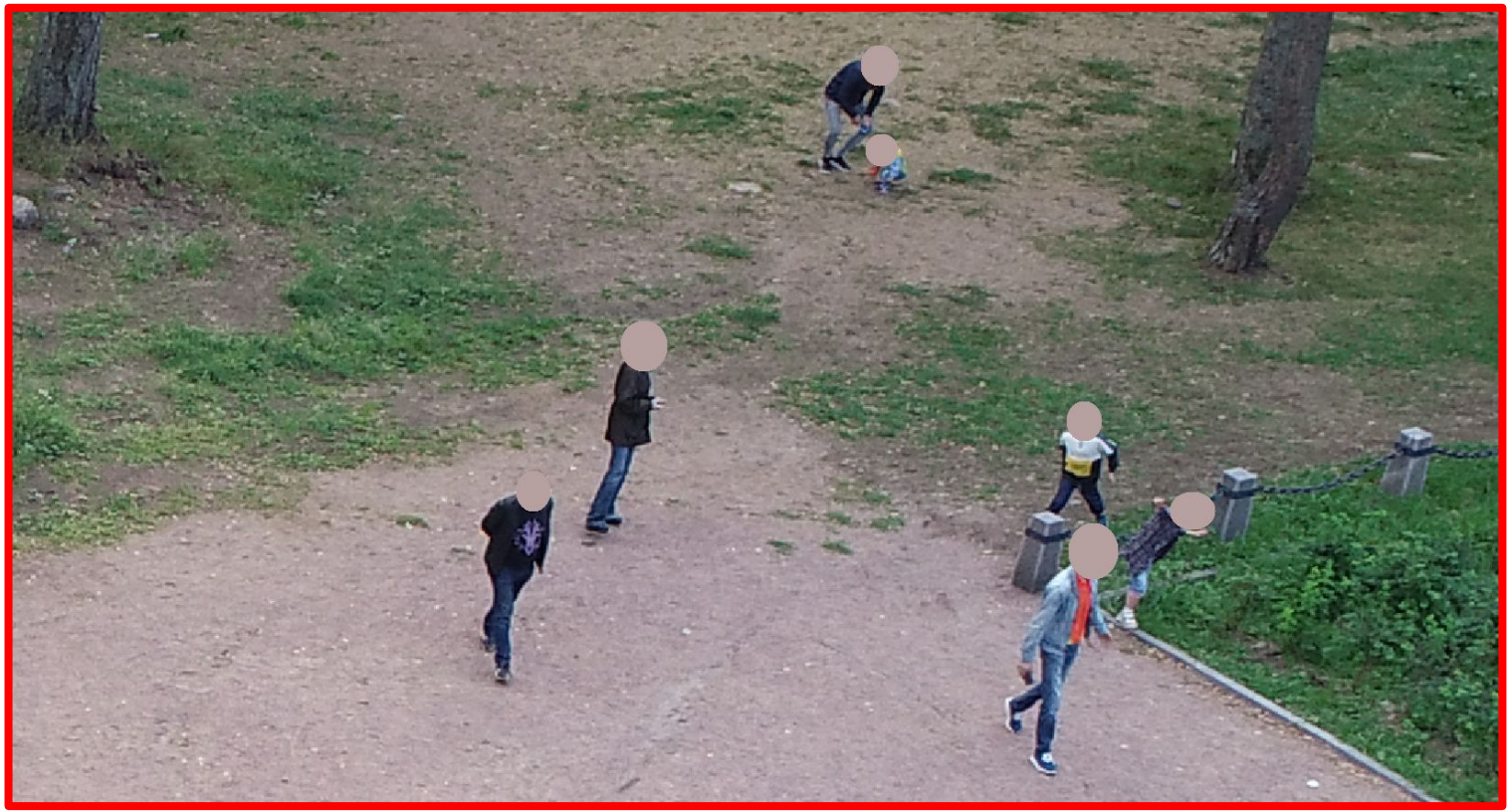
10) Сделать **Python**-скрипт для удаления людей с фотографий

10) Сделать **Python**-скрипт для удаления людей с фотографий



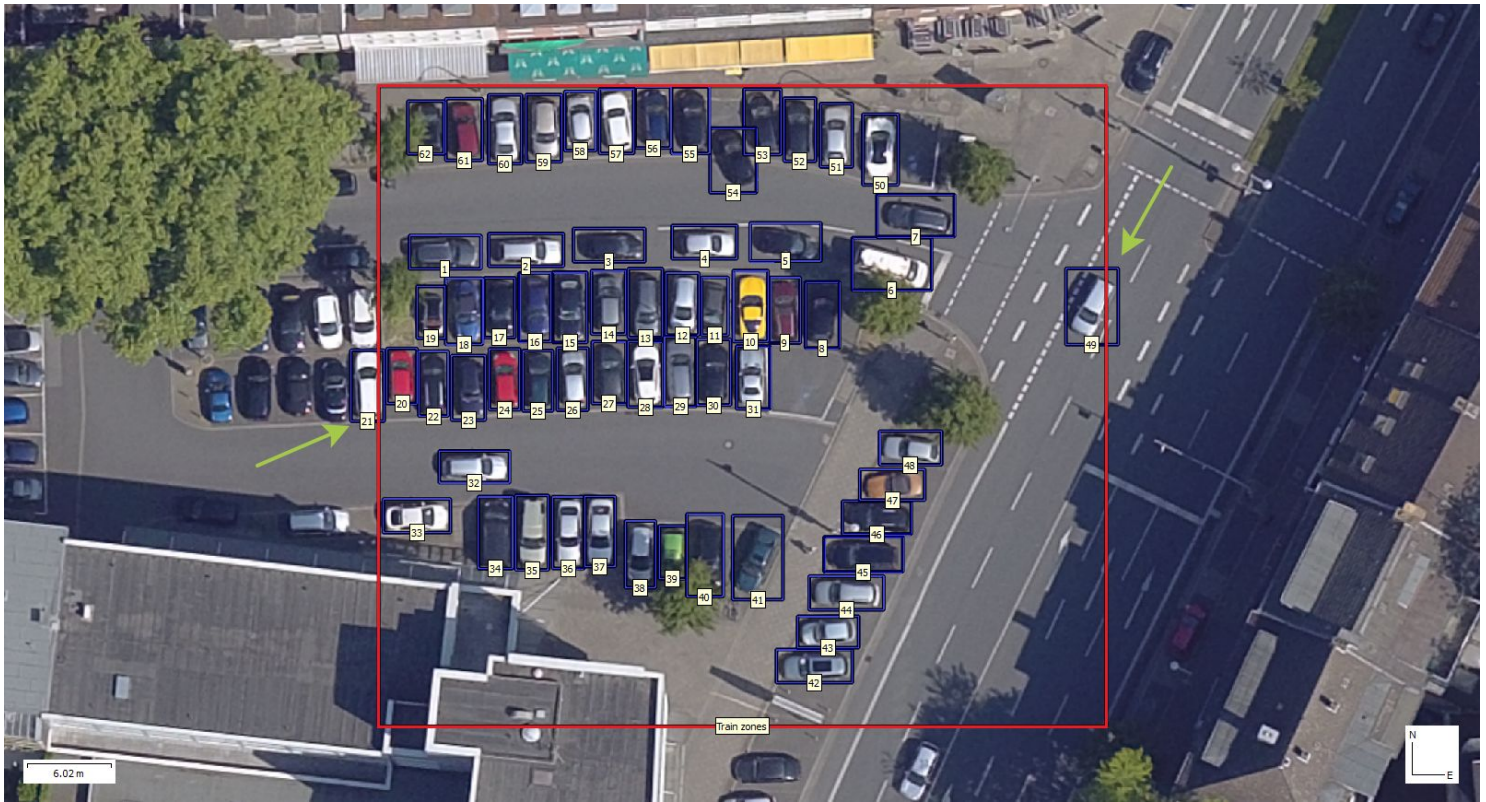
Вопросы?

10) Сделать **Python**-скрипт для удаления людей с фотографий



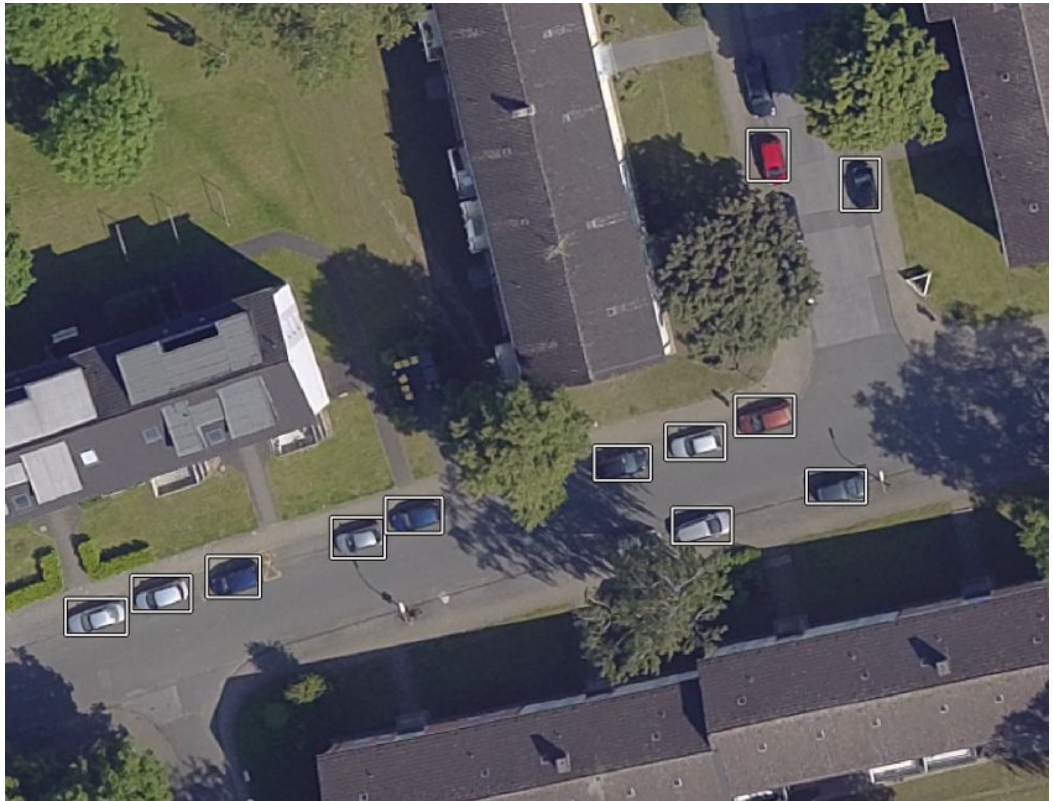
11) Внедрение Python-прототипа нейронки (с дообучением) в C++ приложение

Пользователь указывает зону обучения и размечает там все объекты интереса:



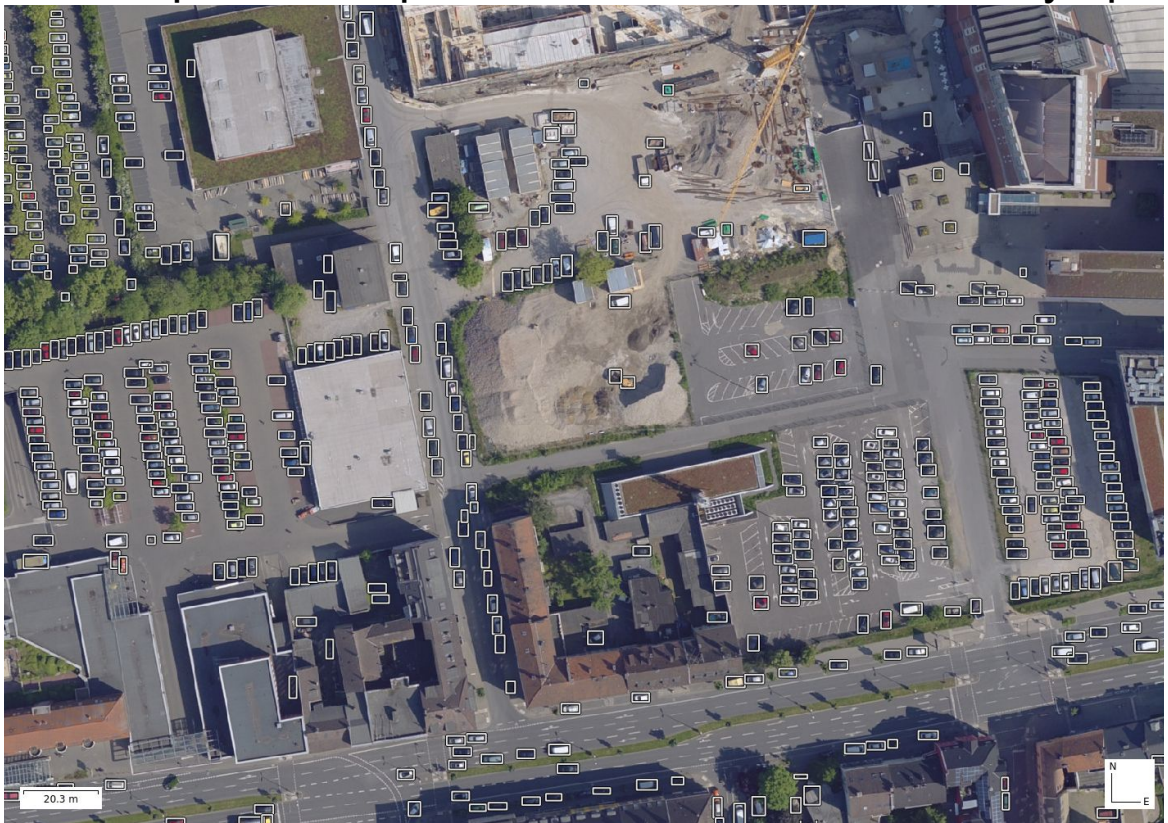
11) Внедрение Python-прототипа нейронки (с дообучением) в C++ приложение

Дообученная таким образом нейронка ищет объекты по всему ортофотоплану:



11) Внедрение Python-прототипа нейронки (с дообучением) в C++ приложение

Дообученная таким образом нейронка ищет объекты по всему ортофотоплану:



11) Внедрение Python-прототипа нейронки (с дообучением) в C++ приложение

Работает с чем угодно: машинами, деревьями, морскими котиками, единорогами.

11) Внедрение Python-прототипа нейронки (с дообучением) в C++ приложение

Скрипт работает, но:

1) Сложно установить т.к. надо скачать много гигабайт питоновских пакетов

11) Внедрение Python-прототипа нейронки (с дообучением) в C++ приложение

Скрипт работает, но:

1) Сложно установить т.к. надо скачать много гигабайт питоновских пакетов
2) Требует наличие NVIDIA GPU (все остальные алгоритмы работают везде)

11) Внедрение Python-прототипа нейронки (с дообучением) в C++ приложение

Скрипт работает, но:
1) Сложно установить т.к. надо скачать много гигабайт питоновских пакетов
2) Требует наличие NVIDIA GPU (все остальные алгоритмы работают везде)

Хочется изучит вопрос - как лучше всего внедрять нейронки в C++ приложение.
В идеале на базе открытых API (OpenCL / Vulkan и т.п.)

Вопросы?

11) Внедрение Python-прототипа нейронки (с дообучением) в C++ приложение

Скрипт работает, но:
1) Сложно установить т.к. надо скачать много гигабайт питоновских пакетов
2) Требует наличие NVIDIA GPU (все остальные алгоритмы работают везде)

Хочется изучит вопрос - как лучше всего внедрять нейронки в C++ приложение.
В идеале на базе открытых API (OpenCL / Vulkan и т.п.)

12) Сделать Python-скрипт для возможности использовать нейронные методы реконструкции карт глубины, сравнить SGM и PatchMatch с нейронным SOTA

12) Сделать Python-скрипт для возможности использовать нейронные методы реконструкции карт глубины, сравнить SGM и PatchMatch с нейронным SOTA

У нас есть Python API.

1) Хочется сделать скрипт который будет делегировать задачу построения карт глубины внешнему алгоритму

12) Сделать Python-скрипт для возможности использовать нейронные методы реконструкции карт глубины, сравнить SGM и PatchMatch с нейронным SOTA

У нас есть Python API.

1) Хочется сделать скрипт который будет делегировать задачу построения карт глубины внешнему алгоритму
2) Сравнить наши алгоритмы с нейронными методами

Вопросы?

12) Сделать Python-скрипт для возможности использовать нейронные методы реконструкции карт глубины, сравнить SGM и PatchMatch с нейронным SOTA

У нас есть Python API.

1) Хочется сделать скрипт который будет делегировать задачу построения карт глубины внешнему алгоритму
2) Сравнить наши алгоритмы с нейронными методами

Административные детали

- Офис в нескольких минутах пешком от ст. м. Маяковская
- Если есть доп. вопросы или какие-то сомнения - смело пишите (telegram/email)
- Стажировка оплачиваемая
- Потенциальное трудоустройство
- Потенциальное развитие до диплома/статьи

Agisoft Metashape
Полярный Николай
polarnick@agisoft.com

Вопросы?

Agisoft
Metashape

Полярный Николай
polarnick@agisoft.com