

Введение в фотограмметрию

Объединение карт глубины в облако точек

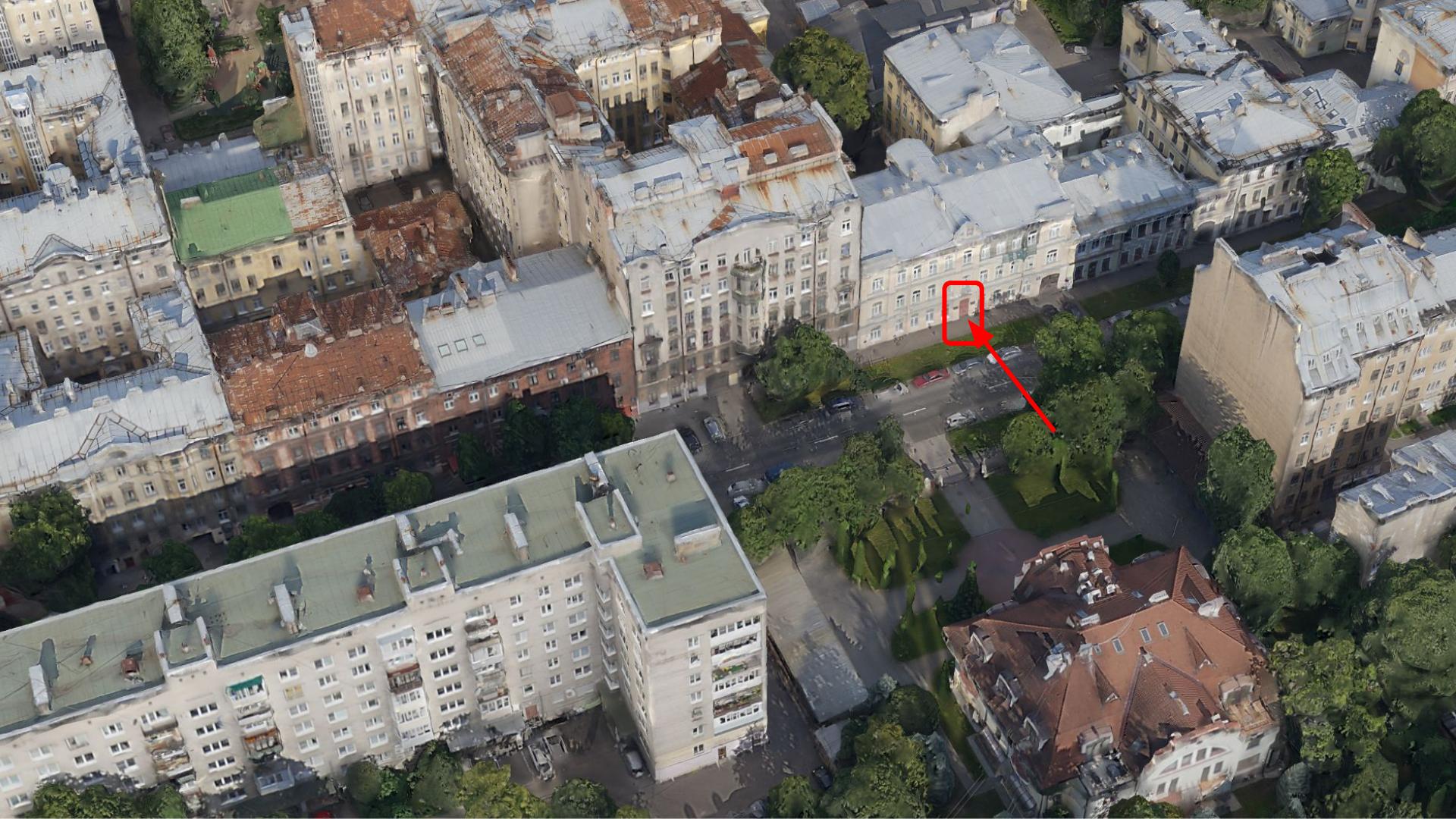
Фотограмметрия. Лекция 14

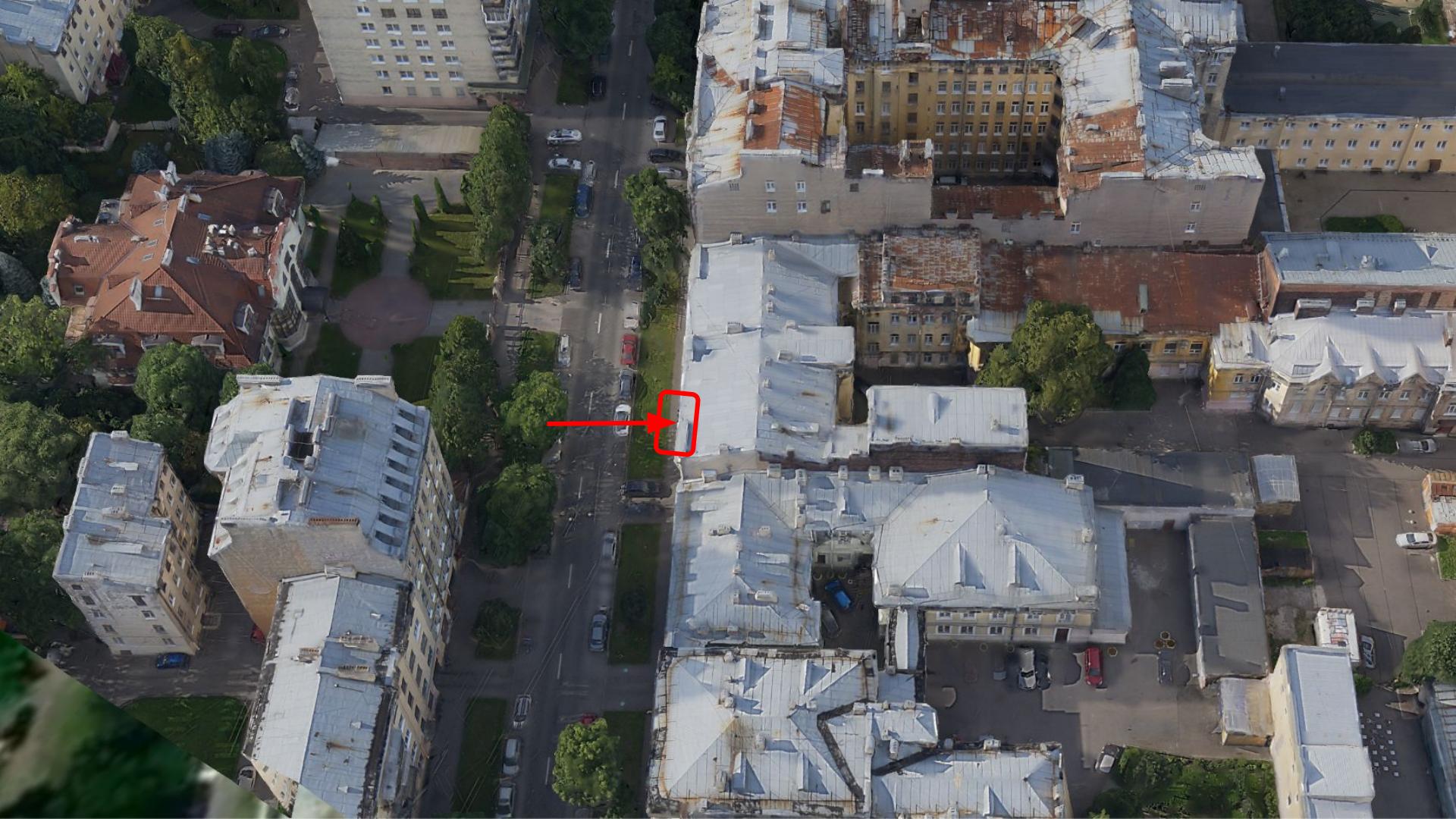


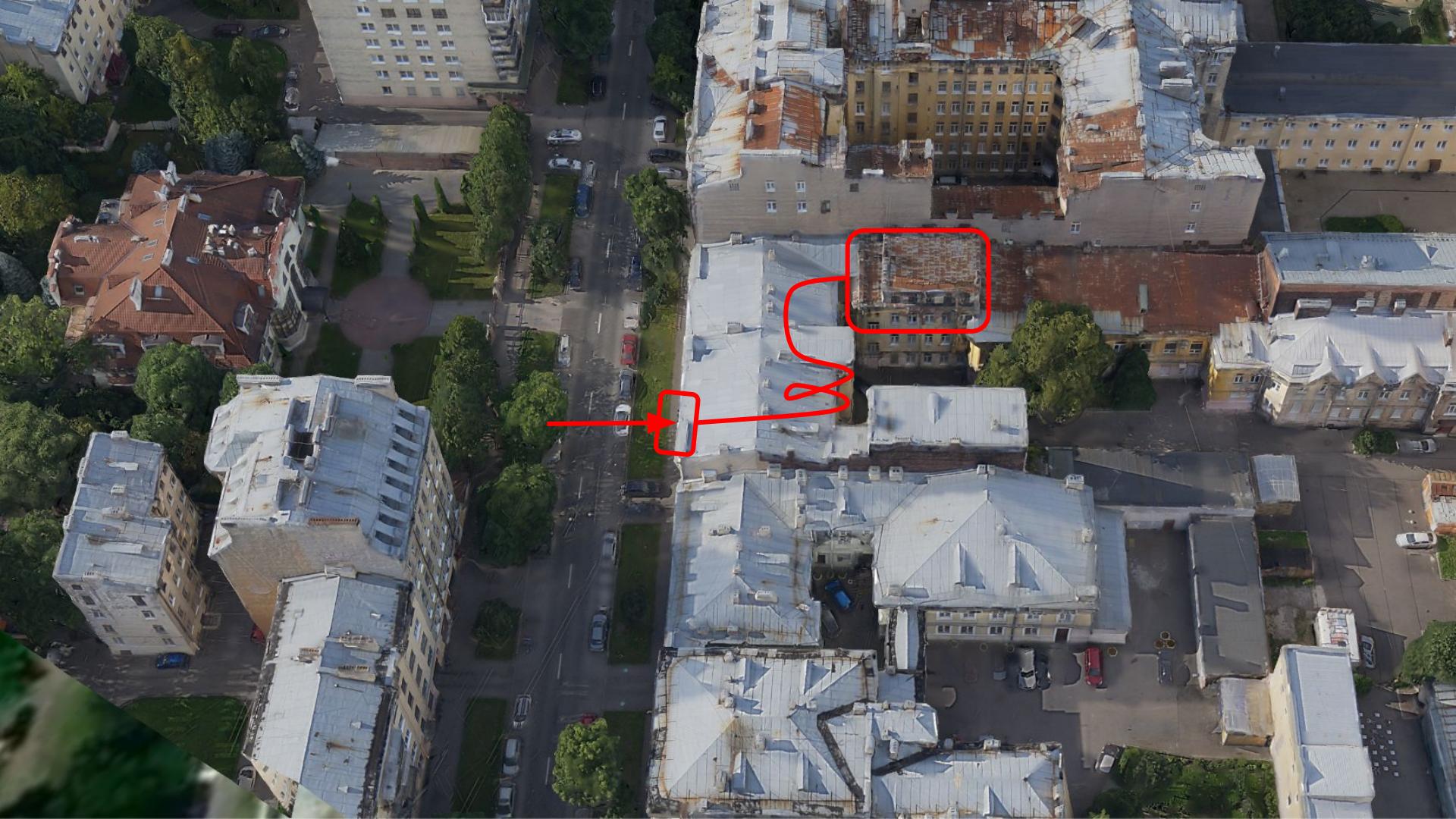
- **PatchMatch для реконструкции волос**
- **Фильтрация отдельной карты глубины**
- **Совместная фильтрация**
- **Poisson реконструкция поверхности**

Полярный Николай
polarnick239@gmail.com







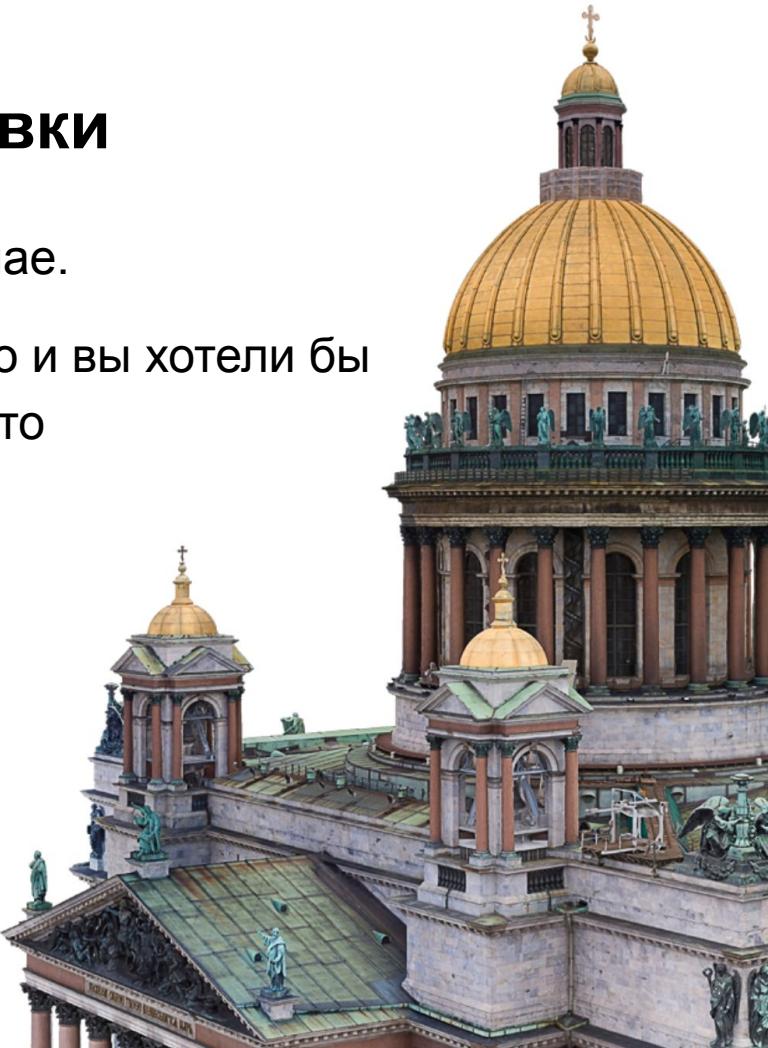


Летние стажировки

Темы летних стажировок будут предложены в мае.

Если же вам важно заранее запланировать лето и вы хотели бы пройти у нас стажировку/устроиться на работу, то напишите мне в телеграмме/на почту:

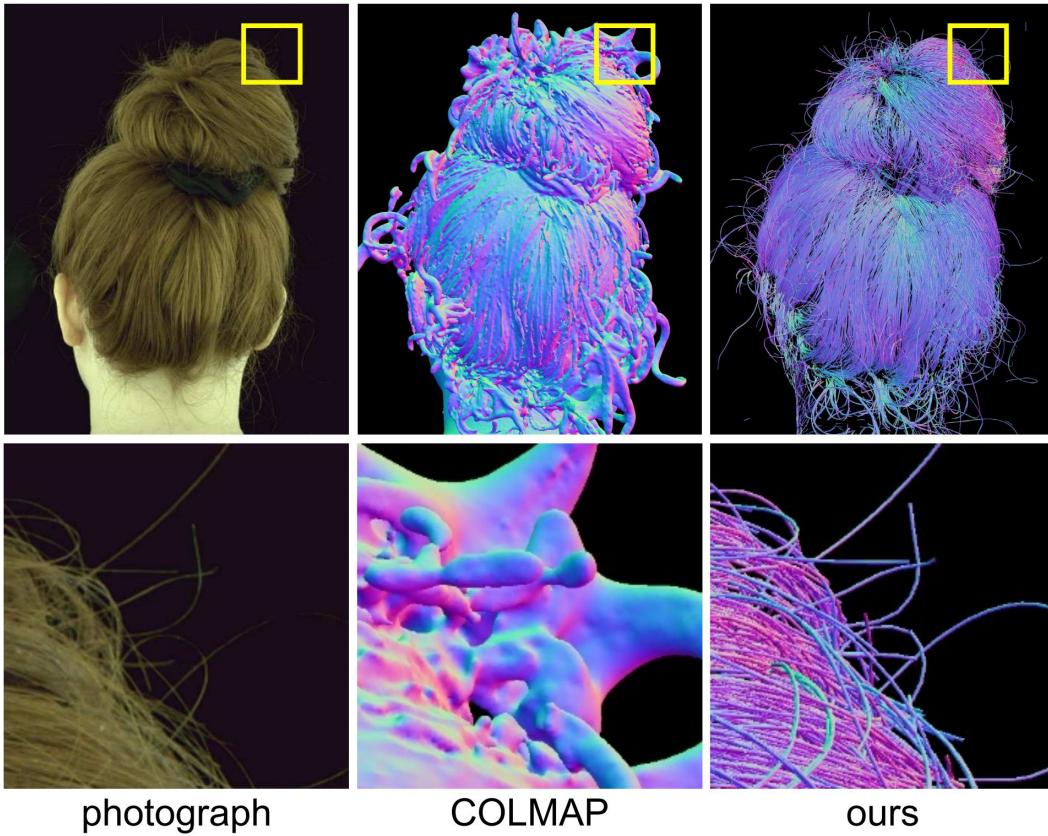
polarnick@agisoft.com



Strand-accurate Multi-view Hair Capture, Nam et. al., 2019

Abstract

Hair is one of the most challenging objects to reconstruct due to its micro-scale structure and a large number of repeated strands with heavy occlusions. In this paper, we present the first method to capture high-fidelity hair geometry with strand-level accuracy. Our method takes three stages to achieve this. In the first stage, a new multi-view stereo method with a slanted support line is proposed to solve the hair correspondences between different views. In detail, we contribute a novel cost function consisting of both photo-consistency term and geometric term that reconstructs each hair pixel as a 3D line. By merging all the depth maps, a point cloud, as well as local line directions for each point, is obtained. Thus, in the second stage, we feature a novel strand reconstruction method with the mean-shift to convert the noisy point data to a set of strands. Lastly, we grow the hair strands with multi-view geometric constraints to elongate the short strands and recover the missing strands, thus significantly increasing the reconstruction completeness. We evaluate our method on both synthetic data and real captured data, showing that our method can reconstruct hair strands with sub-millimeter accuracy.



[youtube: \[CVPR19\] Strand-accurate Multi-view Hair Capture](#)

Strand-accurate Multi-view Hair Capture, Nam et. al., 2019

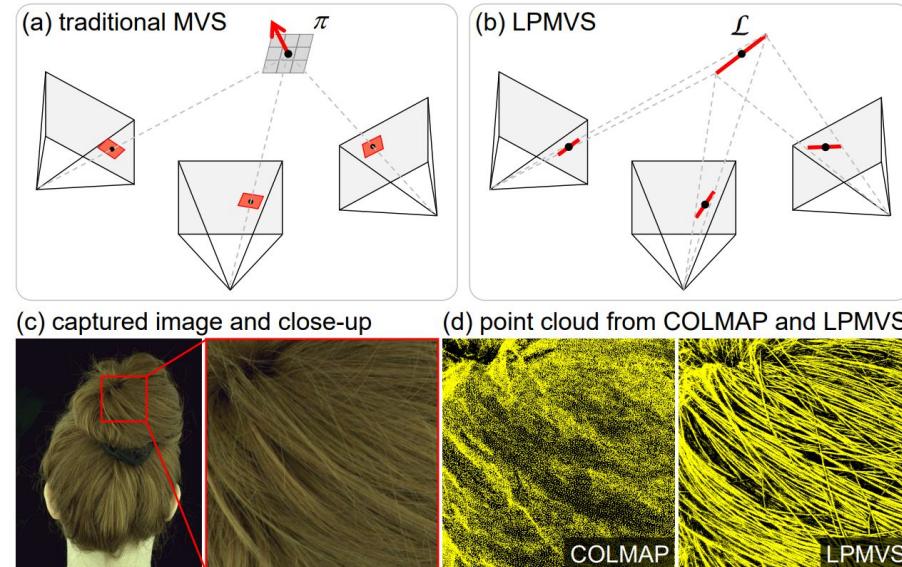


Figure 3. (a) Traditional MVS with a plane assumption. (b) LP-MVS with a line assumption. (c) One of the captured images. (d) Point cloud from COLMAP [33] and our LPMVS.

[youtube: \[CVPR19\] Strand-accurate Multi-view Hair Capture](#)

Strand-accurate Multi-view Hair Capture, Nam et. al., 2019

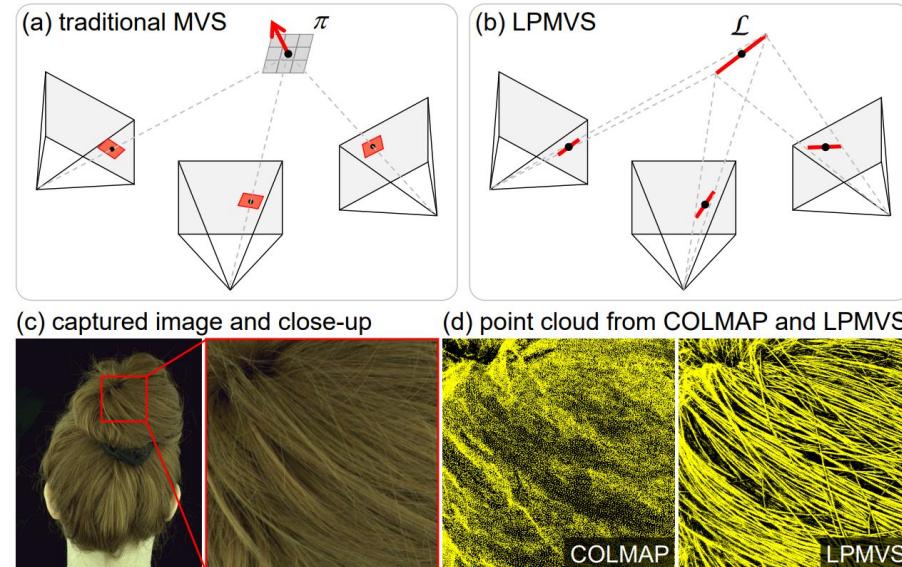


Figure 3. (a) Traditional MVS with a plane assumption. (b) LPMVS with a line assumption. (c) One of the captured images. (d) Point cloud from COLMAP [33] and our LPMVS.

Algorithm 1 Line-based PatchMatch MVS

Input: multi-view images and 2D orientation fields

Output: 3D line maps

```
1: for each view do
2:   set reference and neighboring views
3:   randomly initialize a 3D line map
4:   for iteration  $i = 1$  to  $N_{iter}$  do
5:     update the 3D line map via spatial propagation
6:     refine the 3D line map via random perturbation
7:   end for
8: end for
```

[youtube: \[CVPR19\] Strand-accurate Multi-view Hair Capture](#)

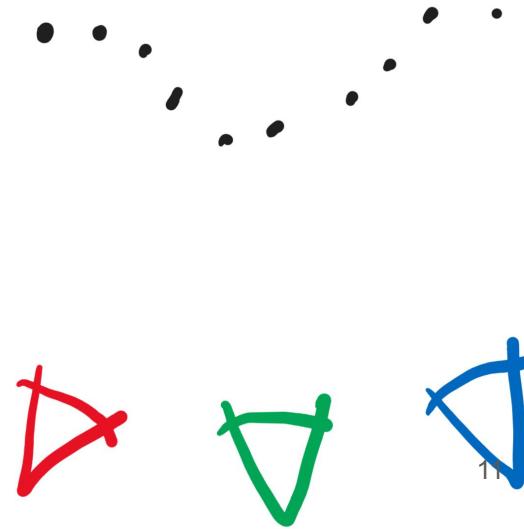
Где мы сейчас?

- Есть точная калибровка камеры и ракурсов фотографирования (intrinsics & extrinsics parameters)



Где мы сейчас?

- Есть точная калибровка камеры и ракурсов фотографирования (intrinsics & extrinsics parameters)
- Есть точное но разреженное (недетальное) облако ключевых точек



Где мы сейчас?

- Есть точная калибровка камеры и ракурсов фотографирования (intrinsics & extrinsics parameters)
- Есть точное но разреженное (недетальное) облако ключевых точек
- Есть детальная но с ошибками/выбросами карта глубины каждой камеры



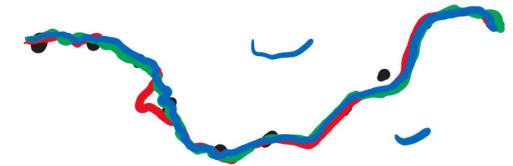
Где мы сейчас?

- Есть точная калибровка камеры и ракурсов фотографирования (intrinsics & extrinsics parameters)
- Есть точное но разреженное (недетальное) облако ключевых точек
- Есть детальная но с ошибками/выбросами карта глубины каждой камеры



Где мы сейчас?

- Есть точная калибровка камеры и ракурсов фотографирования (intrinsics & extrinsics parameters)
- Есть точное но разреженное (недетальное) облако ключевых точек
- Есть детальная но с ошибками/выбросами карта глубины каждой камеры



Где мы сейчас?

- Есть точная калибровка камеры и ракурсов фотографирования (intrinsics & extrinsics parameters)
- Есть точное но разреженное (недетальное) облако ключевых точек
- Есть детальная но с ошибками/выбросами карта глубины каждой камеры

Хотим:

- Очистить каждую карту глубины от выбросов



Где мы сейчас?

- Есть точная калибровка камеры и ракурсов фотографирования (intrinsics & extrinsics parameters)
- Есть точное но разреженное (недетальное) облако ключевых точек
- Есть детальная но с ошибками/выбросами карта глубины каждой камеры

Хотим:

- Очистить каждую карту глубины от выбросов
- Совместно отфильтровать карты глубины от выбросов



Где мы сейчас?

- Есть точная калибровка камеры и ракурсов фотографирования (intrinsics & extrinsics parameters)
- Есть точное но разреженное (недетальное) облако ключевых точек
- Есть детальная но с ошибками/выбросами карта глубины каждой камеры

Хотим:

- Очистить каждую карту глубины от выбросов
- Совместно отфильтровать карты глубины от выбросов
- Объединить отфильтрованные карты глубины в облако точек



Где мы сейчас?

- Есть точная калибровка камеры и ракурсов фотографирования (intrinsics & extrinsics parameters)
- Есть точное но разреженное (недетальное) облако ключевых точек
- Есть детальная но с ошибками/выбросами карта глубины каждой камеры

Хотим:

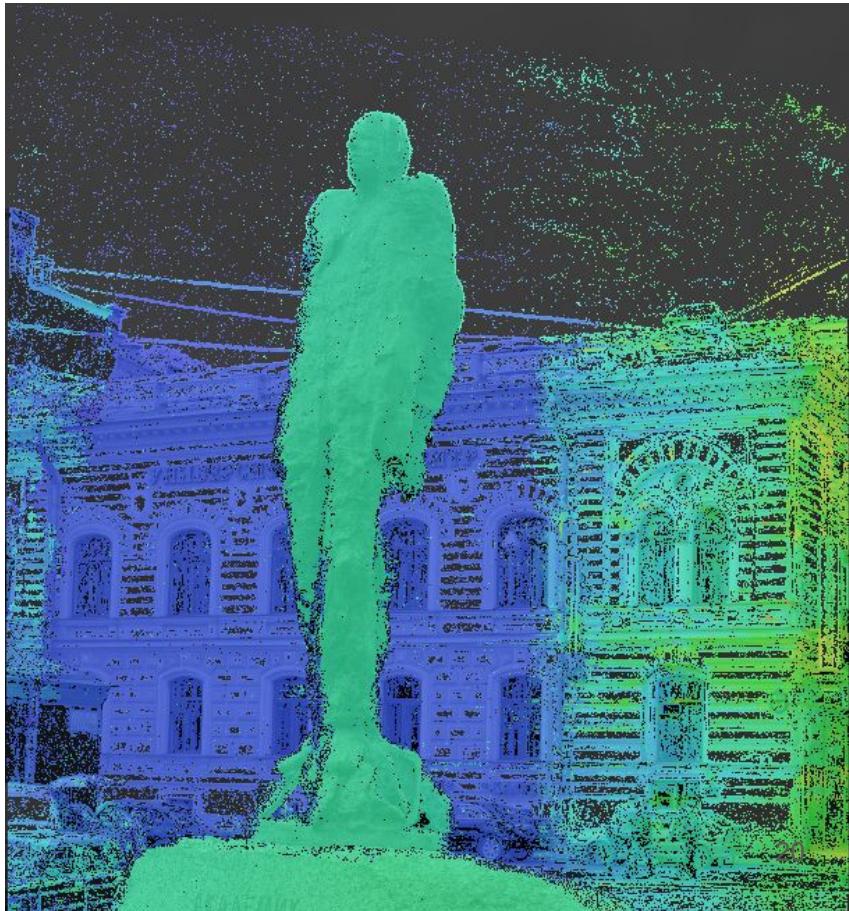
- Очистить каждую карту глубины от выбросов
- Совместно отфильтровать карты глубины от выбросов
- Объединить отфильтрованные карты глубины в облако точек
- По облаку точек построить полигональную модель



Фильтрация шума в построенной по стереопаре карте глубины

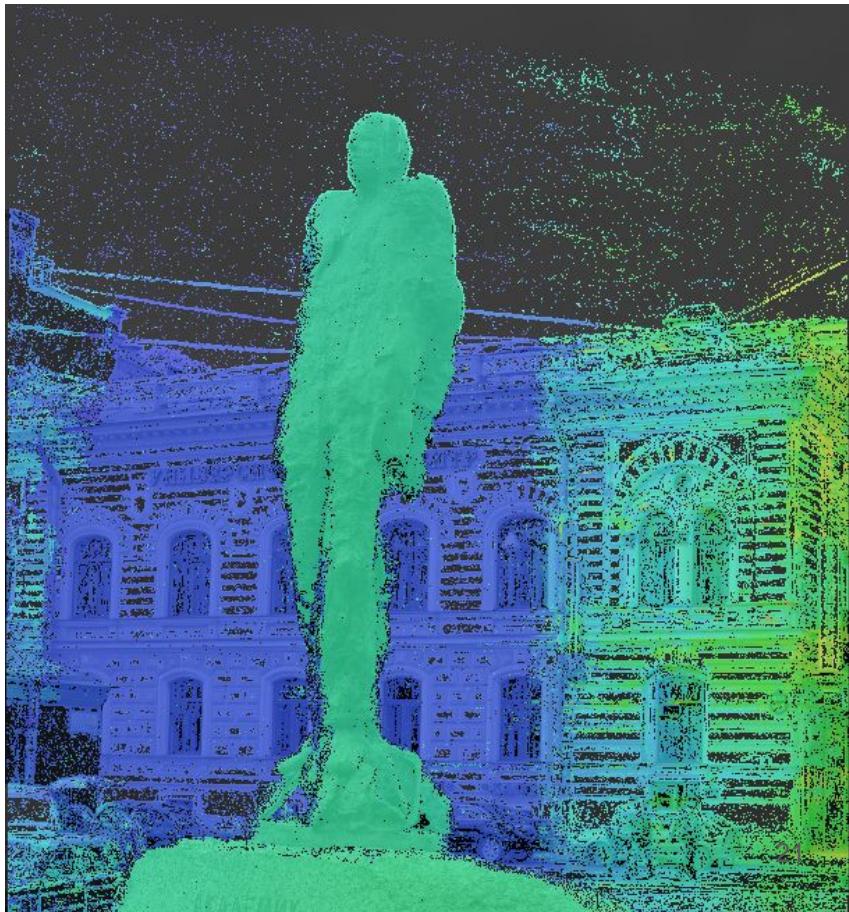


1. Удалим всех с **cost > threshold**:

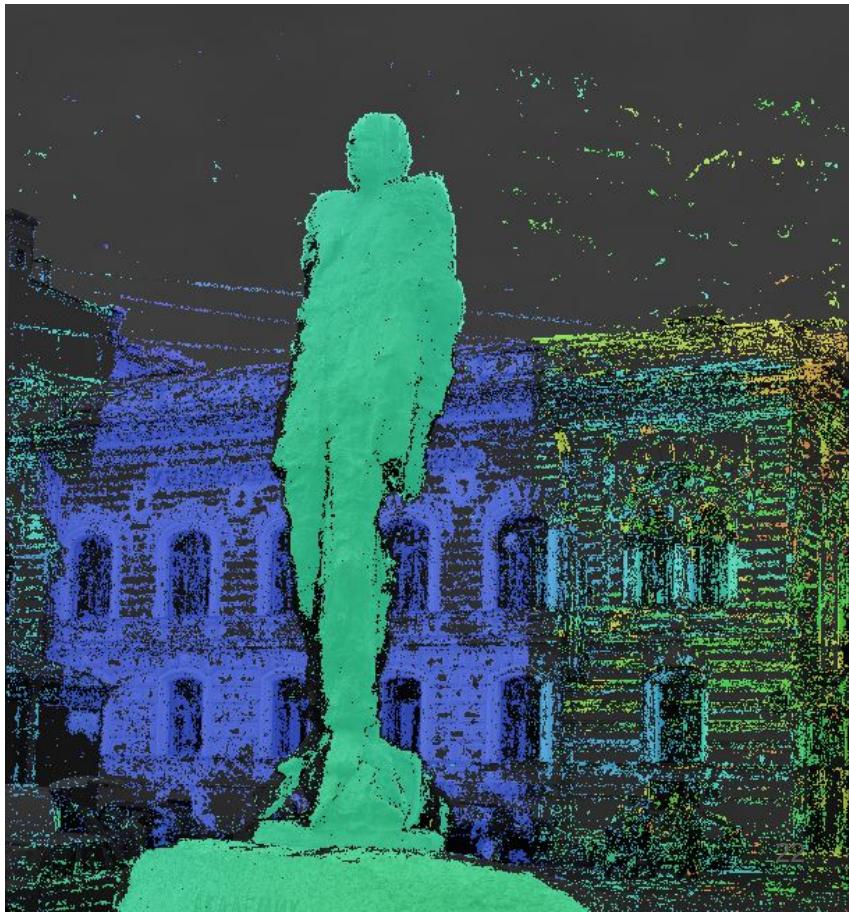


1. Удалим всех с **cost > threshold**:

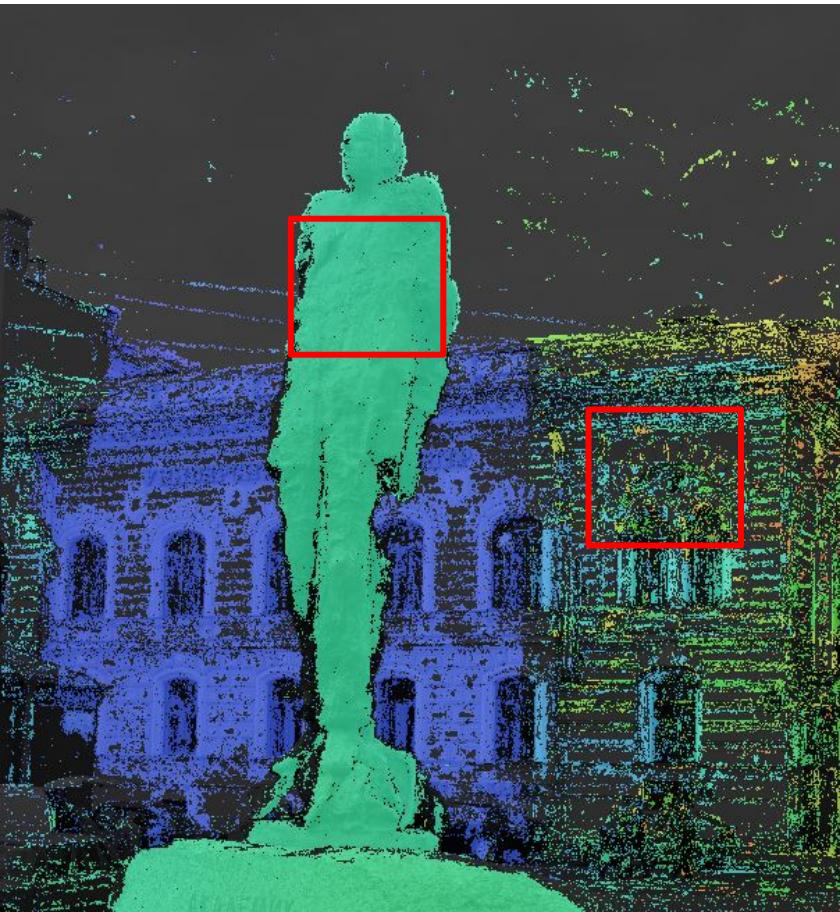
$$C_{NCC}(\mathbf{p}, \mathbf{d}) = \frac{\sum_{\mathbf{q} \in N_p} I_L(\mathbf{q})I_R(\mathbf{q} - \mathbf{d})}{\sqrt{\sum_{\mathbf{q} \in N_p} I_L(\mathbf{q})^2 \sum_{\mathbf{q} \in N_p} I_R(\mathbf{q} - \mathbf{d})^2}}$$



2. Удалим всех чья нормаль не согласована с глубинами вокруг:



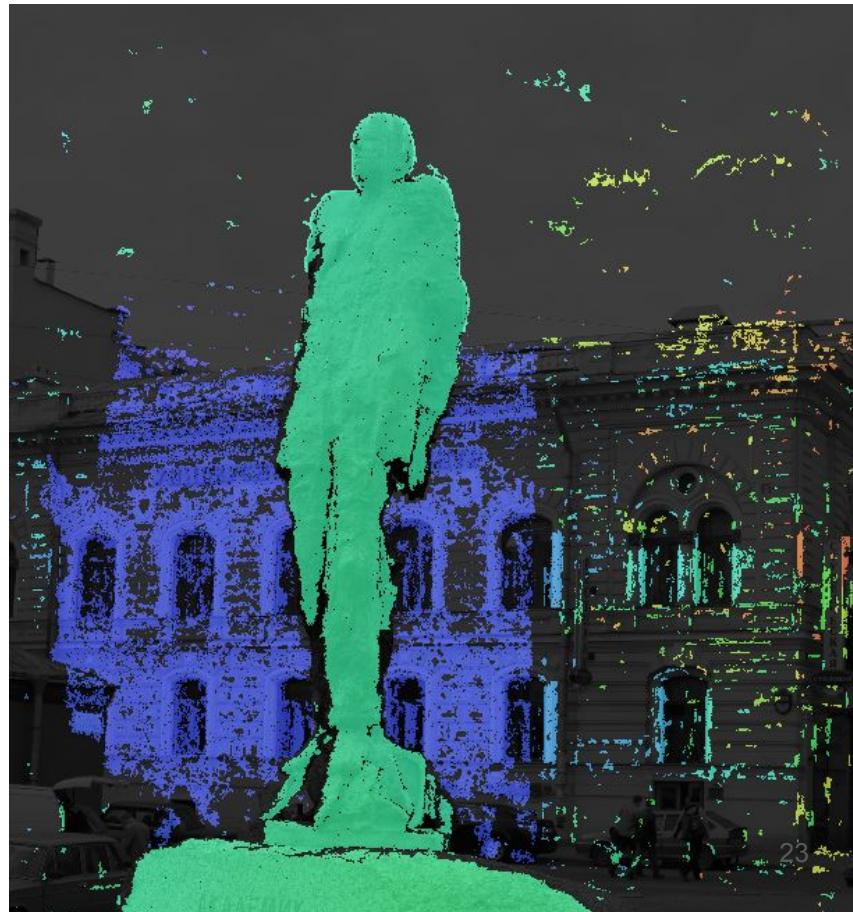
3. Удалим всех чьи компонента связности **маленького** размера:



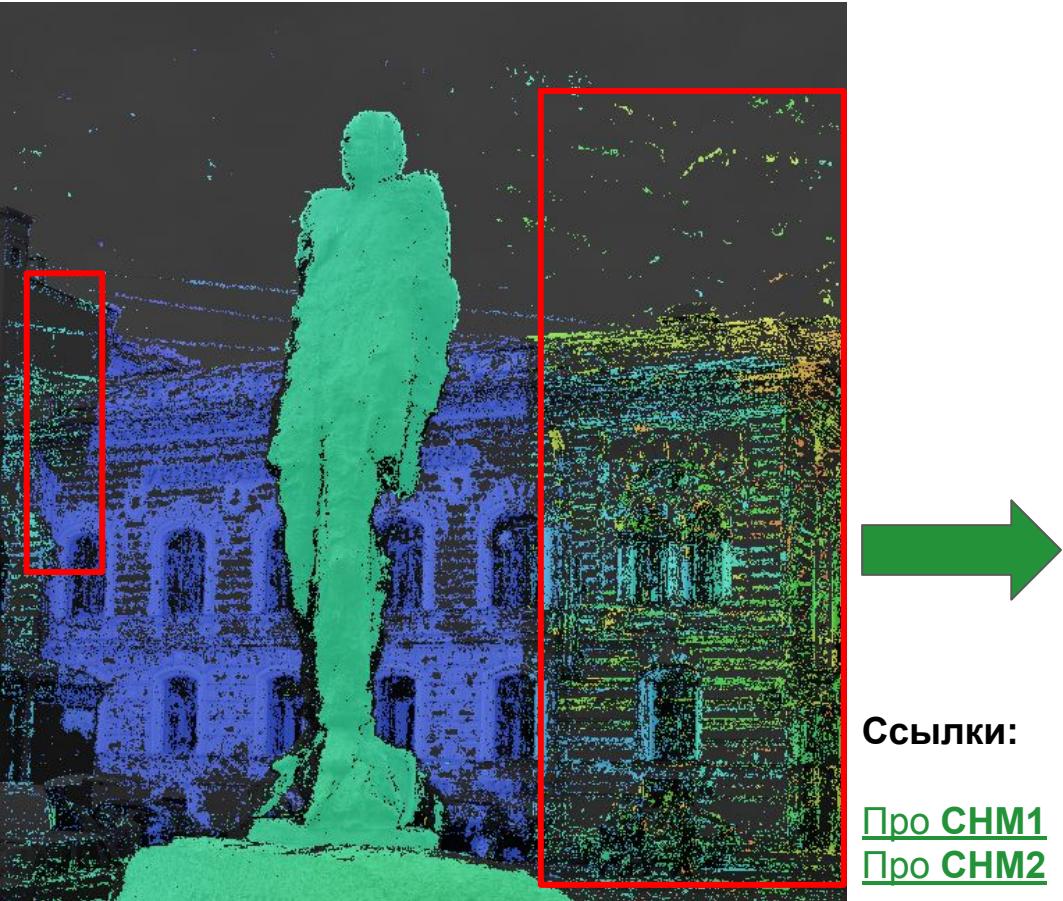
Ссылки:

[Про CHM1](#)

[Про CHM2](#)



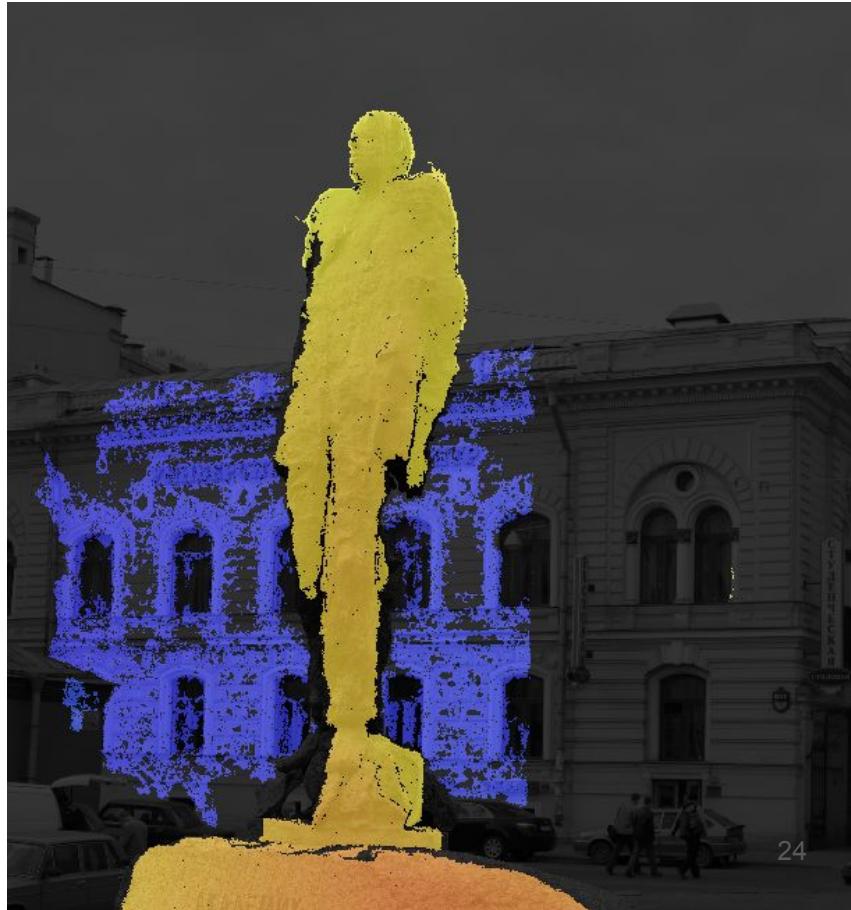
3. Удалим всех чья компонента связности **среднего** размера:



Ссылки:

[Про CHM1](#)

[Про CHM2](#)



Наблюдения этапа фильтрации отдельной карты глубины

- 1) Внутренние части кирпичей стены - отфильтровались (бестекстурные)

Наблюдения этапа фильтрации отдельной карты глубины

- 1) Внутренние части кирпичей стены - отфильтровались (бестекстурные)
- 2) Целевой объект - статуя на камне - практически не тронута

Наблюдения этапа фильтрации отдельной карты глубины

- 1) Внутренние части кирпичей стены - отфильтровались (бестекстурные)
- 2) Целевой объект - статуя на камне - практически не тронута
- 3) Зоны видные только с одной камеры - отфильтровались

Наблюдения этапа фильтрации отдельной карты глубины

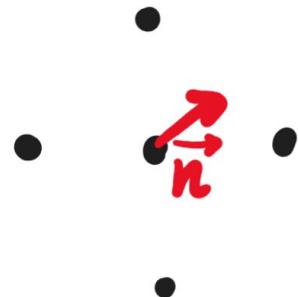
- 1) Внутренние части кирпичей стены - отфильтровались (бестекстурные)
- 2) Целевой объект - статуя на камне - практически не тронута
- 3) Зоны видные только с одной камеры - отфильтровались
- 4) Провода отфильтровались на этапе проверки согласованности нормалей

Наблюдения этапа фильтрации отдельной карты глубины

- 1) Внутренние части кирпичей стены - отфильтровались (бестекстурные)
- 2) Целевой объект - статуя на камне - практически не тронута
- 3) Зоны видные только с одной камеры - отфильтровались
- 4) Провода отфильтровались на этапе проверки согласованности нормалей
- 5) **Как фильтровать по нормалям?**

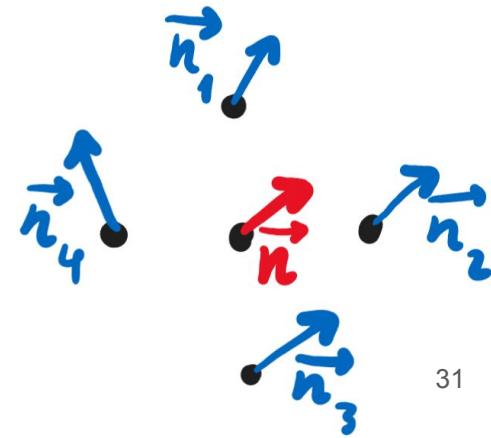
Наблюдения этапа фильтрации отдельной карты глубины

- 1) Внутренние части кирпичей стены - отфильтровались (бестекстурные)
- 2) Целевой объект - статуя на камне - практически не тронута
- 3) Зоны видные только с одной камеры - отфильтровались
- 4) Провода отфильтровались на этапе проверки согласованности нормалей
- 5) **Как фильтровать по нормалям?**



Наблюдения этапа фильтрации отдельной карты глубины

- 1) Внутренние части кирпичей стены - отфильтровались (бестекстурные)
- 2) Целевой объект - статуя на камне - практически не тронута
- 3) Зоны видные только с одной камеры - отфильтровались
- 4) Провода отфильтровались на этапе проверки согласованности нормалей
- 5) **Как фильтровать по нормалям?**

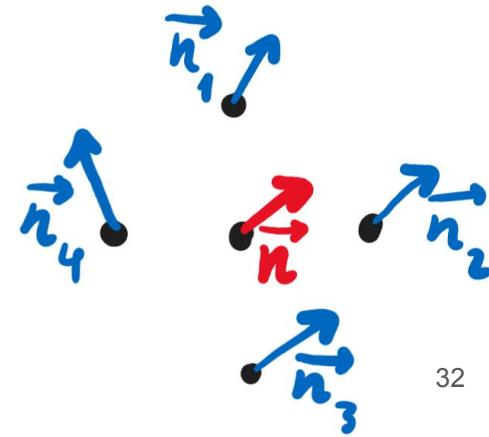


Наблюдения этапа фильтрации отдельной карты глубины

- 1) Внутренние части кирпичей стены - отфильтровались (бестекстурные)
- 2) Целевой объект - статуя на камне - практически не тронута
- 3) Зоны видные только с одной камеры - отфильтровались
- 4) Провода отфильтровались на этапе проверки согласованности нормалей
- 5) **Как фильтровать по нормалям?**

5.1) По согласованности с нормалями соседей

$$\vec{n} \approx \text{average}/\text{median}(\vec{n}_1, \vec{n}_2, \vec{n}_3, \vec{n}_4)$$



Наблюдения этапа фильтрации отдельной карты глубины

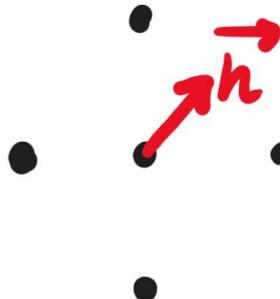
- 1) Внутренние части кирпичей стены - отфильтровались (бестекстурные)
- 2) Целевой объект - статуя на камне - практически не тронута
- 3) Зоны видные только с одной камеры - отфильтровались
- 4) Провода отфильтровались на этапе проверки согласованности нормалей
- 5) Как фильтровать по нормалям?**
 - 5.1) По согласованности с нормалями соседей**
 - 5.2) ???**

Наблюдения этапа фильтрации отдельной карты глубины

- 1) Внутренние части кирпичей стены - отфильтровались (бестекстурные)
- 2) Целевой объект - статуя на камне - практически не тронута
- 3) Зоны видные только с одной камеры - отфильтровались
- 4) Провода отфильтровались на этапе проверки согласованности нормалей
- 5) Как фильтровать по нормалям?**
 - 5.1) По согласованности с нормалями соседей**
 - 5.2) ???**

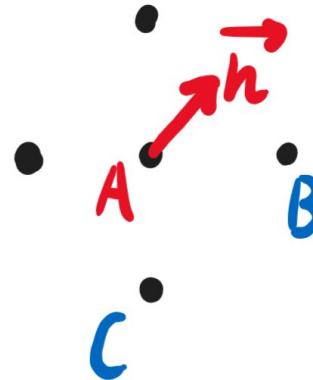
Наблюдения этапа фильтрации отдельной карты глубины

- 1) Внутренние части кирпичей стены - отфильтровались (бестекстурные)
- 2) Целевой объект - статуя на камне - практически не тронута
- 3) Зоны видные только с одной камеры - отфильтровались
- 4) Провода отфильтровались на этапе проверки согласованности нормалей
- 5) **Как фильтровать по нормалям?**
 - 5.1) По согласованности с нормалями соседей
 - 5.2) ???



Наблюдения этапа фильтрации отдельной карты глубины

- 1) Внутренние части кирпичей стены - отфильтровались (бестекстурные)
- 2) Целевой объект - статуя на камне - практически не тронута
- 3) Зоны видные только с одной камеры - отфильтровались
- 4) Провода отфильтровались на этапе проверки согласованности нормалей
- 5) **Как фильтровать по нормалям?**
 - 5.1) По согласованности с нормалями соседей
 - 5.2) ???

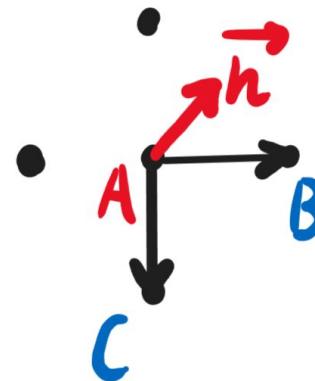


Наблюдения этапа фильтрации отдельной карты глубины

- 1) Внутренние части кирпичей стены - отфильтровались (бестекстурные)
- 2) Целевой объект - статуя на камне - практически не тронута
- 3) Зоны видные только с одной камеры - отфильтровались
- 4) Провода отфильтровались на этапе проверки согласованности нормалей
- 5) **Как фильтровать по нормалям?**

5.1) По согласованности с нормалями соседей

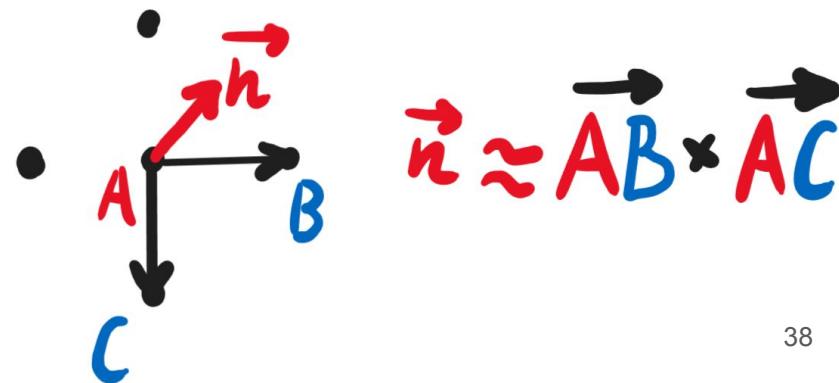
5.2) ???



$$\vec{AB} \times \vec{AC}$$

Наблюдения этапа фильтрации отдельной карты глубины

- 1) Внутренние части кирпичей стены - отфильтровались (бестекстурные)
- 2) Целевой объект - статуя на камне - практически не тронута
- 3) Зоны видные только с одной камеры - отфильтровались
- 4) Провода отфильтровались на этапе проверки согласованности нормалей
- 5) **Как фильтровать по нормалям?**
 - 5.1) По согласованности с нормалями соседей
 - 5.2) По согласованности с глубиной



Наблюдения этапа фильтрации отдельной карты глубины

- 1) Внутренние части кирпичей стены - отфильтровались (бестекстурные)
- 2) Целевой объект - статуя на камне - практически не тронута
- 3) Зоны видные только с одной камеры - отфильтровались
- 4) Провода отфильтровались на этапе проверки согласованности нормалей
- 5) Как фильтровать по нормалям?
 - 5.1) По согласованности с нормалями соседей
 - 5.2) По согласованности с глубиной
- 6) **Всегда ли соседние пиксели объединяются в компоненту связности?**

Наблюдения этапа фильтрации отдельной карты глубины

- 1) Внутренние части кирпичей стены - отфильтровались (бестекстурные)
- 2) Целевой объект - статуя на камне - практически не тронута
- 3) Зоны видные только с одной камеры - отфильтровались
- 4) Провода отфильтровались на этапе проверки согласованности нормалей
- 5) Как фильтровать по нормалям?
 - 5.1) По согласованности с нормалями соседей
 - 5.2) По согласованности с глубиной
- 6) **Объединяем только близкие по глубине соседние пиксели**

0. Везде что-то да нашли:



1. Удалим всех с **cost > threshold**:



2. Удалим всех чья нормаль не согласована с глубинами вокруг:



АКАДЕМИК
А.САХОРОВ

3. Удалим всех чья компонента связности **маленького** размера:



3. Удалим всех чья компонента связности **среднего** размера:

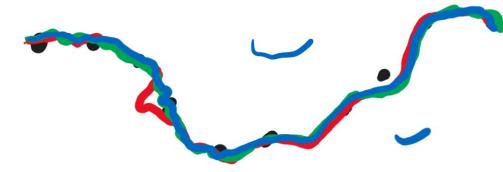


Где мы сейчас?

- Есть точная калибровка камеры и ракурсов фотографирования (intrinsics & extrinsics parameters)
- Есть точное но разреженное (недетальное) облако ключевых точек
- Есть детальная но с ошибками/выбросами карта глубины каждой камеры

Хотим:

- Очистить каждую карту глубины от выбросов
- **Совместно отфильтровать карты глубины от выбросов**
- Объединить отфильтрованные карты глубины в облако точек
- По облаку точек построить полигональную модель



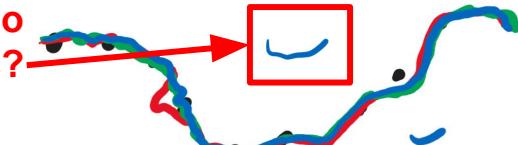
Где мы сейчас?

- Есть точная калибровка камеры и ракурсов фотографирования (intrinsics & extrinsics parameters)
- Есть точное но разреженное (недетальное) облако ключевых точек
- Есть детальная но с ошибками/выбросами карта глубины каждой камеры

Хотим:

- Очистить каждую карту глубины от выбросов
- **Совместно отфильтровать карты глубины от выбросов**
- Объединить отфильтрованные карты глубины в облако точек
- По облаку точек построить полигональную модель

На базе какой улики - можно
удалить этот выброс?



Где мы сейчас?

- Есть точная калибровка камеры и ракурсов фотографирования (intrinsics & extrinsics parameters)
- Есть точное но разреженное (недетальное) облако ключевых точек
- Есть детальная но с ошибками/выбросами карта глубины каждой камеры

Хотим:

- Очистить каждую карту глубины от выбросов
- **Совместно отфильтровать карты глубины от выбросов**
- Объединить отфильтрованные карты глубины в облако точек
- По облаку точек построить полигональную модель

На пути нашего синего луча
лежат две другие поверхности!



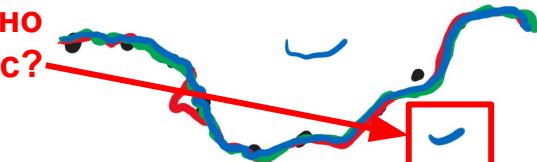
Где мы сейчас?

- Есть точная калибровка камеры и ракурсов фотографирования (intrinsics & extrinsics parameters)
- Есть точное но разреженное (недетальное) облако ключевых точек
- Есть детальная но с ошибками/выбросами карта глубины каждой камеры

Хотим:

- Очистить каждую карту глубины от выбросов
- **Совместно отфильтровать карты глубины от выбросов**
- Объединить отфильтрованные карты глубины в облако точек
- По облаку точек построить полигональную модель

На базе какой улики - можно
удалить этот выброс?



Где мы сейчас?

- Есть точная калибровка камеры и ракурсов фотографирования (intrinsics & extrinsics parameters)
- Есть точное но разреженное (недетальное) облако ключевых точек
- Есть детальная но с ошибками/выбросами карта глубины каждой камеры

Хотим:

- Очистить каждую карту глубины от выбросов
- **Совместно отфильтровать карты глубины от выбросов**
- Объединить отфильтрованные карты глубины в облако точек
- По облаку точек построить полигональную модель

Эта синяя поверхность лежит на
пути у двух других лучей!



Где мы сейчас?

- Есть точная калибровка камеры и ракурсов фотографирования (intrinsics & extrinsics parameters)
- Есть точное но разреженное (недетальное) облако ключевых точек
- Есть детальная но с ошибками/выбросами карта глубины каждой камеры

Хотим:

- Очистить каждую карту глубины от выбросов
- **Совместно отфильтровать карты глубины от выбросов**
- Объединить отфильтрованные карты глубины в облако точек
- По облаку точек построить полигональную модель

Как в камере A по пикслю с
глубиной быстро проверить -
заслоняем ли мы что-то в другой
камере B?



Где мы сейчас?

- Есть точная калибровка камеры и ракурсов фотографирования (intrinsics & extrinsics parameters)
- Есть точное но разреженное (недетальное) облако ключевых точек
- Есть детальная но с ошибками/выбросами карта глубины каждой камеры

Хотим:

- Очистить каждую карту глубины от выбросов
- **Совместно отфильтровать карты глубины от выбросов**
- Объединить отфильтрованные карты глубины в облако точек
- По облаку точек построить полигональную модель

Как в камере A по пикслю с
глубиной быстро проверить -
заслоняем ли мы что-то в другой
камере B?



Где мы сейчас?

- Есть точная калибровка камеры и ракурсов фотографирования (intrinsics & extrinsics parameters)
- Есть точное но разреженное (недетальное) облако ключевых точек
- Есть детальная но с ошибками/выбросами карта глубины каждой камеры

Хотим:

- Очистить каждую карту глубины от выбросов
- Совместно отфильтровать карты глубины от выбросов**
- Объединить отфильтрованные карты глубины в облако точек
- По облаку точек построить полигональную модель

Как в камере A по пикслю с
глубиной быстро проверить -
заслоняем ли мы что-то в другой
камере B?



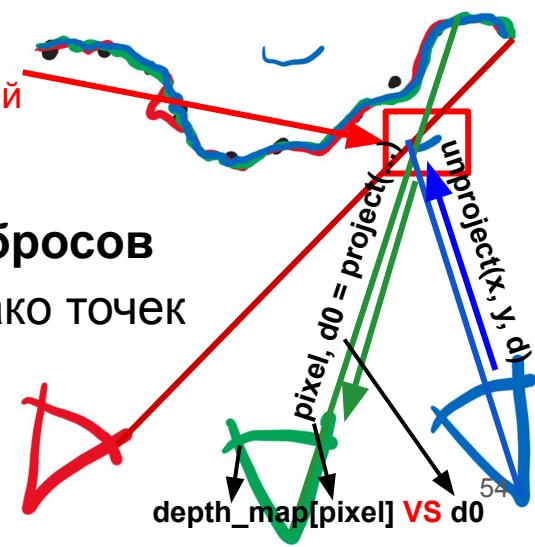
Где мы сейчас?

- Есть точная калибровка камеры и ракурсов фотографирования (intrinsics & extrinsics parameters)
- Есть точное но разреженное (недетальное) облако ключевых точек
- Есть детальная но с ошибками/выбросами карта глубины каждой камеры

Хотим:

- Очистить каждую карту глубины от выбросов
- **Совместно отфильтровать карты глубины от выбросов**
- Объединить отфильтрованные карты глубины в облако точек
- По облаку точек построить полигональную модель

Как в камере A по пикслю с
глубиной быстро проверить -
заслоняем ли мы что-то в другой
камере B?



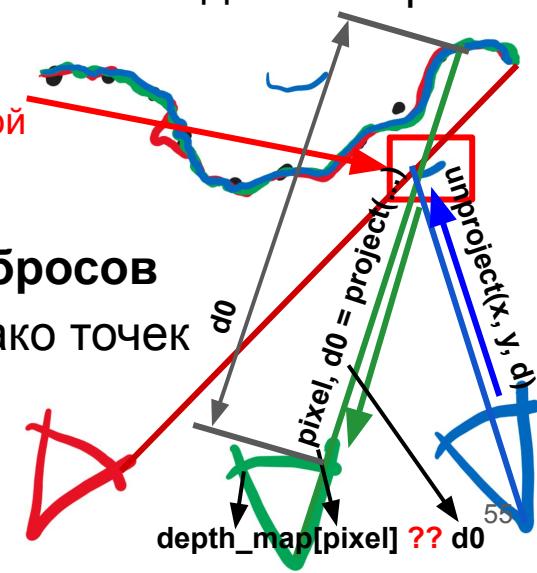
Где мы сейчас?

- Есть точная калибровка камеры и ракурсов фотографирования (intrinsics & extrinsics parameters)
- Есть точное но разреженное (недетальное) облако ключевых точек
- Есть детальная но с ошибками/выбросами карта глубины каждой камеры

Хотим:

- Очистить каждую карту глубины от выбросов
- **Совместно отфильтровать карты глубины от выбросов**
- Объединить отфильтрованные карты глубины в облако точек
- По облаку точек построить полигональную модель

Как в камере A по пикслю с
глубиной быстро проверить -
заслоняем ли мы что-то в другой
камере B?



Где мы сейчас?

- Есть точная калибровка камеры и ракурсов фотографирования (intrinsics & extrinsics parameters)
- Есть точное но разреженное (недетальное) облако ключевых точек
- Есть детальная но с ошибками/выбросами карта глубины каждой камеры

Хотим:

- Очистить каждую карту глубины от выбросов
- **Совместно отфильтровать карты глубины от выбросов**
- Объединить отфильтрованные карты глубины в облако точек
- По облаку точек построить полигональную модель

Как в камере A по пикслю с
глубиной быстро проверить -
заслоняем ли мы что-то в другой
камере B?



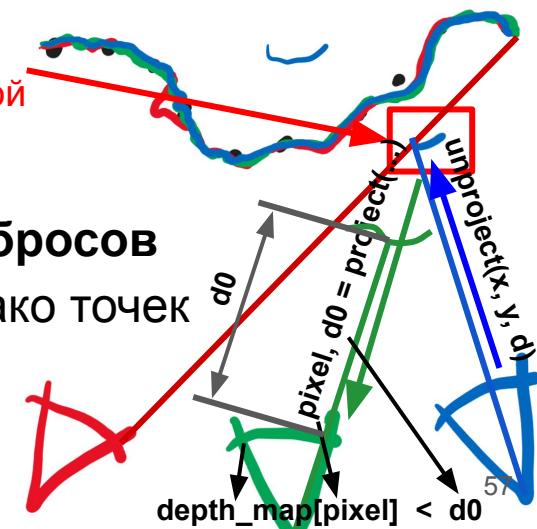
Где мы сейчас?

- Есть точная калибровка камеры и ракурсов фотографирования (intrinsics & extrinsics parameters)
- Есть точное но разреженное (недетальное) облако ключевых точек
- Есть детальная но с ошибками/выбросами карта глубины каждой камеры

Хотим:

- Очистить каждую карту глубины от выбросов
- Совместно отфильтровать карты глубины от выбросов**
- Объединить отфильтрованные карты глубины в облако точек
- По облаку точек построить полигональную модель

Как в камере A по пикслю с
глубиной быстро проверить -
заслоняем ли мы что-то в другой
камере B?



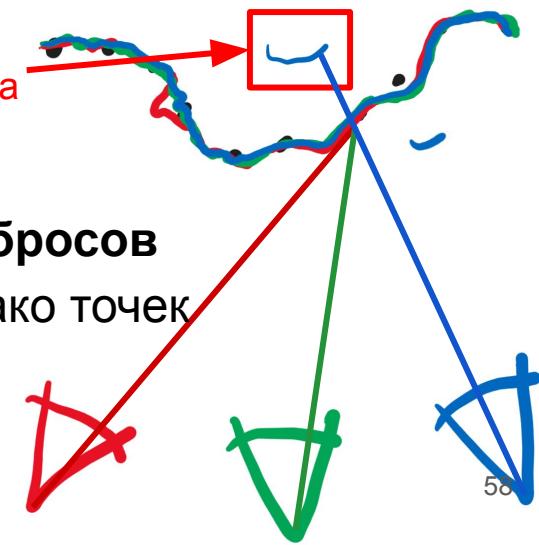
Где мы сейчас?

- Есть точная калибровка камеры и ракурсов фотографирования (intrinsics & extrinsics parameters)
- Есть точное но разреженное (недетальное) облако ключевых точек
- Есть детальная но с ошибками/выбросами карта глубины каждой камеры

Как в камере А по пикслю с
глубиной быстро проверить -
заслоняет ли нас какая то точка
из камеры В?

Хотим:

- Очистить каждую карту глубины от выбросов
- **Совместно отфильтровать карты глубины от выбросов**
- Объединить отфильтрованные карты глубины в облако точек
- По облаку точек построить полигональную модель



Где мы сейчас?

- Есть точная калибровка камеры и ракурсов фотографирования (intrinsics & extrinsics parameters)
- Есть точное но разреженное (недетальное) облако ключевых точек
- Есть детальная но с ошибками/выбросами карта глубины каждой камеры

Как в камере А по пикслю с
глубиной быстро проверить -
заслоняет ли нас какая то точка
из камеры В?

Хотим:

- Очистить каждую карту глубины от выбросов
- **Совместно отфильтровать карты глубины от выбросов**
- Объединить отфильтрованные карты глубины в облако точек
- По облаку точек построить полигональную модель



Где мы сейчас?

- Есть точная калибровка камеры и ракурсов фотографирования (intrinsics & extrinsics parameters)
- Есть точное но разреженное (недетальное) облако ключевых точек
- Есть детальная но с ошибками/выбросами карта глубины каждой камеры

Как в камере А по пикслю с
глубиной быстро проверить -
заслоняет ли нас какая то точка
из камеры В?

Хотим:

- Очистить каждую карту глубины от выбросов
- **Совместно отфильтровать карты глубины от выбросов**
- Объединить отфильтрованные карты глубины в облако точек
- По облаку точек построить полигональную модель



Где мы сейчас?

- Есть точная калибровка камеры и ракурсов фотографирования (intrinsics & extrinsics parameters)
- Есть точное но разреженное (недетальное) облако ключевых точек
- Есть детальная но с ошибками/выбросами карта глубины каждой камеры

Как в камере А по пикслю с
глубиной быстро проверить -
заслоняет ли нас какая то точка
из камеры В?

Хотим:

- Очистить каждую карту глубины от выбросов
- **Совместно отфильтровать карты глубины от выбросов**
- Объединить отфильтрованные карты глубины в облако точек
- По облаку точек построить полигональную модель



Где мы сейчас?

- Есть точная калибровка камеры и ракурсов фотографирования (intrinsics & extrinsics parameters)
- Есть точное но разреженное (недетальное) облако ключевых точек
- Есть детальная но с ошибками/выбросами карта глубины каждой камеры

Хотим:

- Очистить каждую карту глубины от выбросов
- **Совместно отфильтровать карты глубины от выбросов**
- Объединить отфильтрованные карты глубины в облако точек
- По облаку точек построить полигональную модель

Как в камере А по пикслю с
глубиной быстро проверить -
заслоняет ли нас какая то точка
из камеры В?



Real-Time Visibility-Based Fusion of Depth Maps

Paul Merrell¹, Amir Akbarzadeh², Liang Wang², Philippos Mordohai¹, Jan-Michael Frahm¹,
Ruiqiang Yang², David Nistér², and Marc Pollefeys¹

¹Department of Computer Science ² Center for Visualization and Virtual Environments
University of North Carolina, Chapel Hill, USA University of Kentucky, Lexington, USA

Abstract

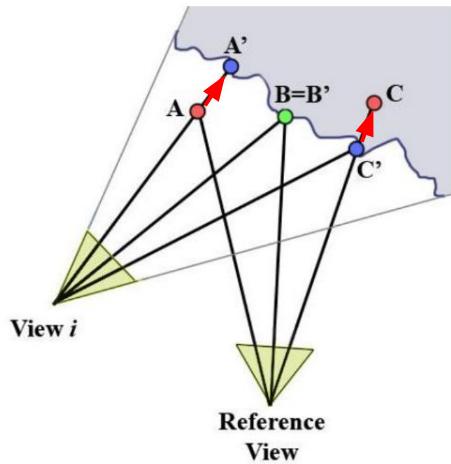
We present a viewpoint-based approach for the quick fusion of multiple stereo depth maps. Our method selects depth estimates for each pixel that minimize violations of visibility constraints and thus remove errors and inconsistencies from the depth maps to produce a consistent surface. We advocate a two-stage process in which the first stage generates potentially noisy, overlapping depth maps from a set of calibrated images and the second stage fuses these depth maps to obtain an integrated surface with higher accuracy, suppressed noise, and reduced redundancy. We show that by dividing the processing into two stages we are able to achieve a very high throughput because we are able to use a computationally cheap stereo algorithm and because this architecture is amenable to hardware-accelerated (GPU) implementations. A rigorous formulation based on the notion of stability of a depth estimate is presented first. It aims to determine the validity of a depth estimate by rendering multiple depth maps into the reference view as well as rendering the reference depth map into the other views in order to detect occlusions and free-space violations. We also present an approximate alterna-



Figure 1. Aerial View and Detailed Views of reconstructed models from a 170,000 frame video

an interest in ground-based models as the next logical step in creating a more effective city visualization tool. Visualizations from ground imagery are possible in the form of panoramic mosaics [21, 16] or simple geometric models [2] which require less data to construct, but limit the user's ability to freely navigate the environment. For unconstrained navigation, accurate and detailed 3D models are needed. Video-based 3D reconstruction can deliver this type of content, but only once it is capable of overcoming the challenge of processing massive amounts of imagery. Typical multiple-view reconstruction algorithms that process all images simultaneously are unable to create models such as the

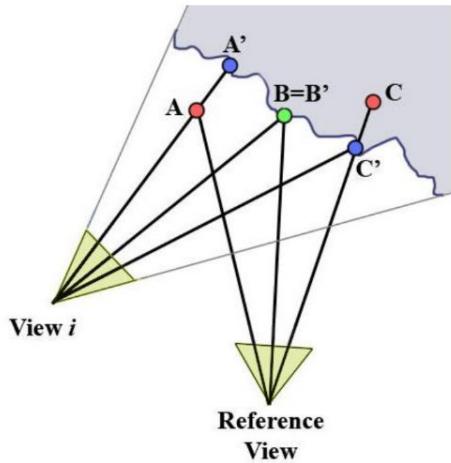
1) Противоречия: мы заслоняем/нас заслоняют



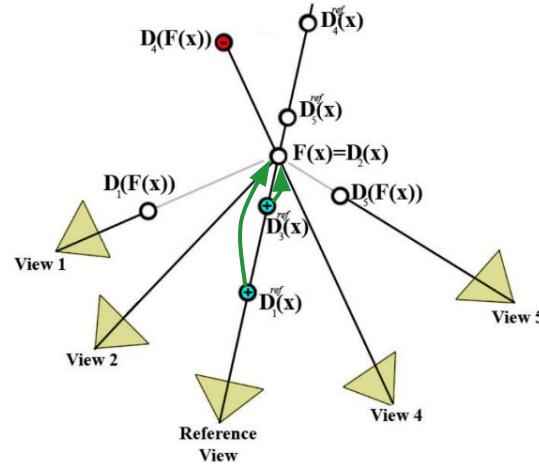
(a) Visibility relations between points

Figure 2. (a) Visibility relations between points. The point A' seen in view i has its free space violated by A seen in the reference view. B' supports B . C seen in the reference view is occluded by C' .

2) Occlusions - сколько заслоняет нас



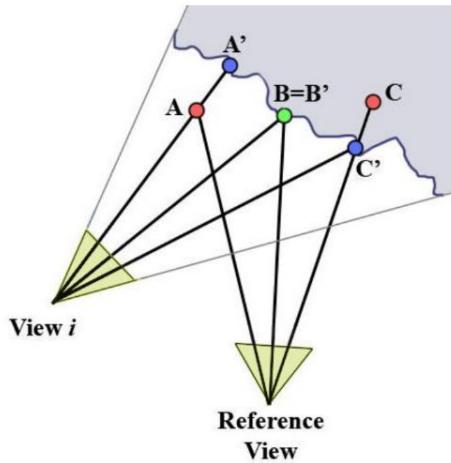
(a) Visibility relations between points



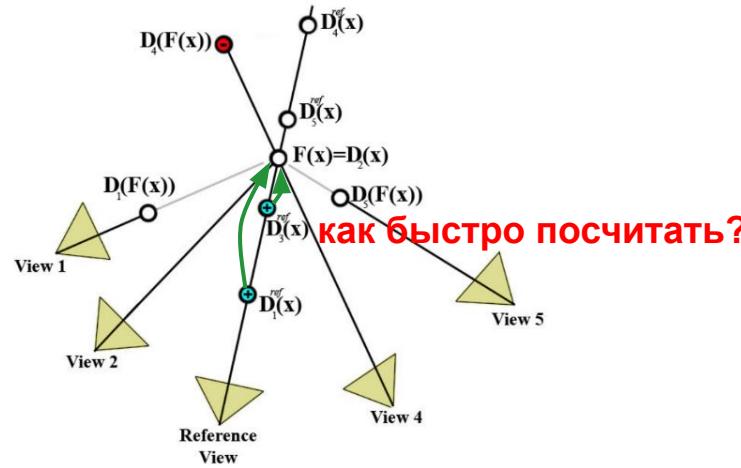
(b) Stability calculation

Figure 2. (a) Visibility relations between points. The point A' seen in view i has its free space violated by A seen in the reference view. B' supports B . C seen in the reference view is occluded by C' . (b) Stability Calculation. In this example, there are two occlusions which raise stability and one free-space violations which lowers it. The stability is +1.

2) Occlusions - сколько заслоняет нас



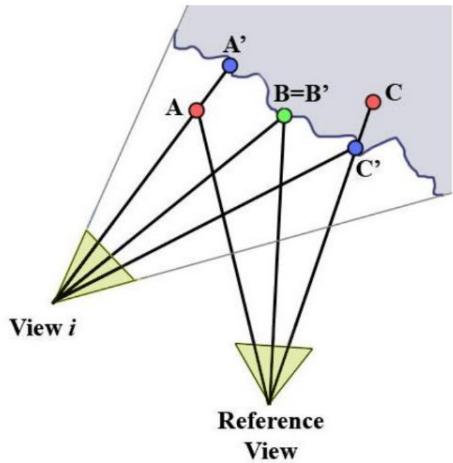
(a) Visibility relations between points



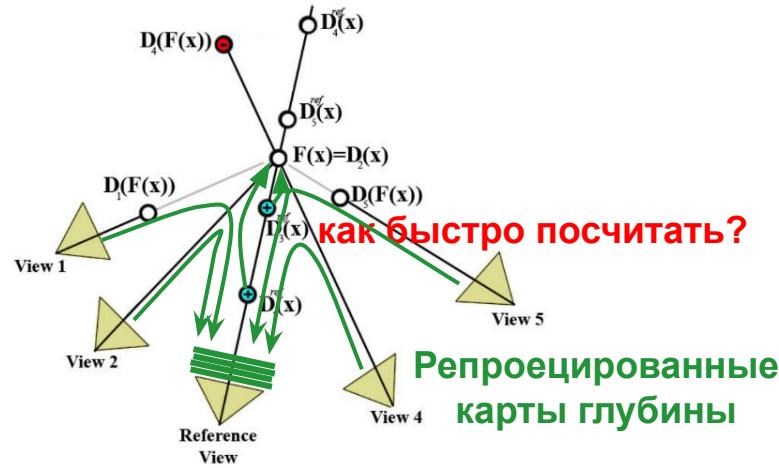
(b) Stability calculation

Figure 2. (a) Visibility relations between points. The point A' seen in view i has its free space violated by A seen in the reference view. B' supports B . C seen in the reference view is occluded by C' . (b) Stability Calculation. In this example, there are two occlusions which raise stability and one free-space violations which lowers it. The stability is +1.

2) Occlusions - сколько заслоняет нас



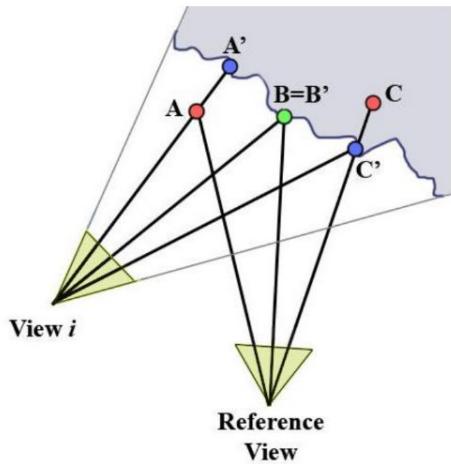
(a) Visibility relations between points



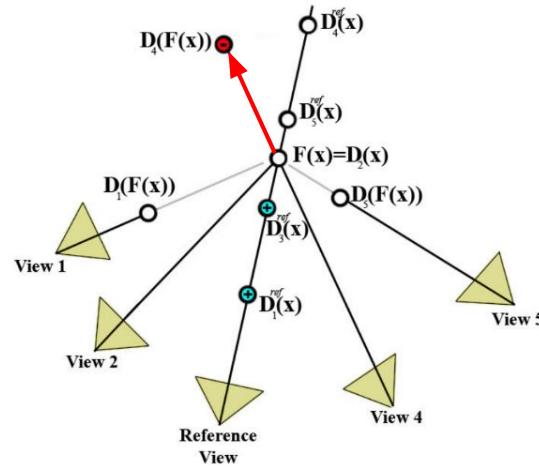
(b) Stability calculation

Figure 2. (a) Visibility relations between points. The point A' seen in view i has its free space violated by A seen in the reference view. B' supports B . C seen in the reference view is occluded by C' . (b) Stability Calculation. In this example, there are two occlusions which raise stability and one free-space violations which lowers it. The stability is +1.

3) Free-space violations - сколько заслоняем мы



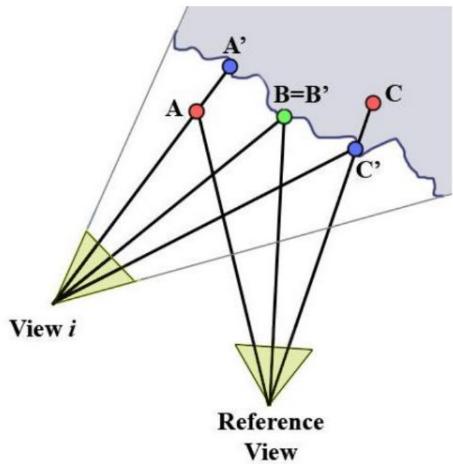
(a) Visibility relations between points



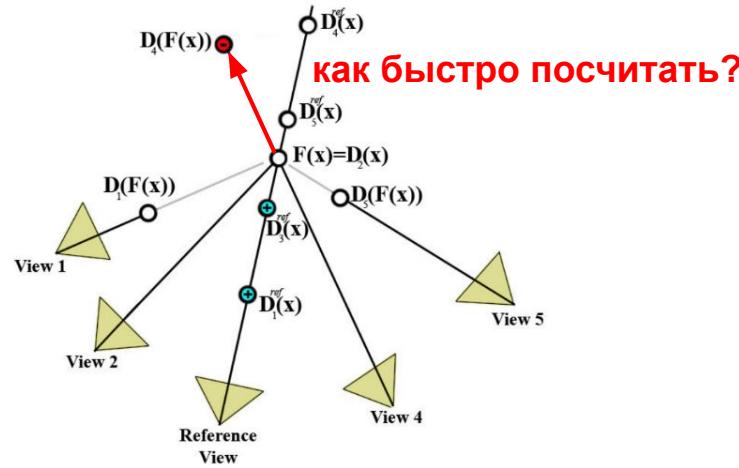
(b) Stability calculation

Figure 2. (a) Visibility relations between points. The point A' seen in view i has its free space violated by A seen in the reference view. B' supports B . C seen in the reference view is occluded by C' . (b) Stability Calculation. In this example, there are two occlusions which raise stability and one free-space violations which lowers it. The stability is +1.

3) Free-space violations - сколько заслоняем мы



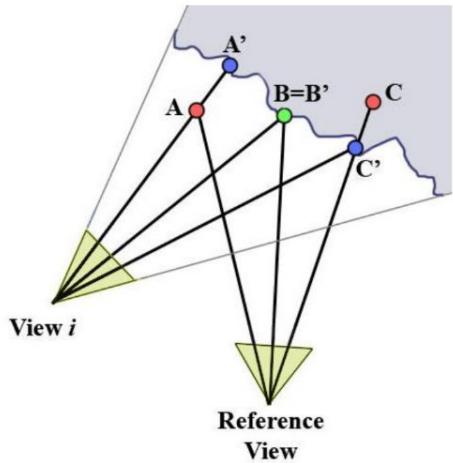
(a) Visibility relations between points



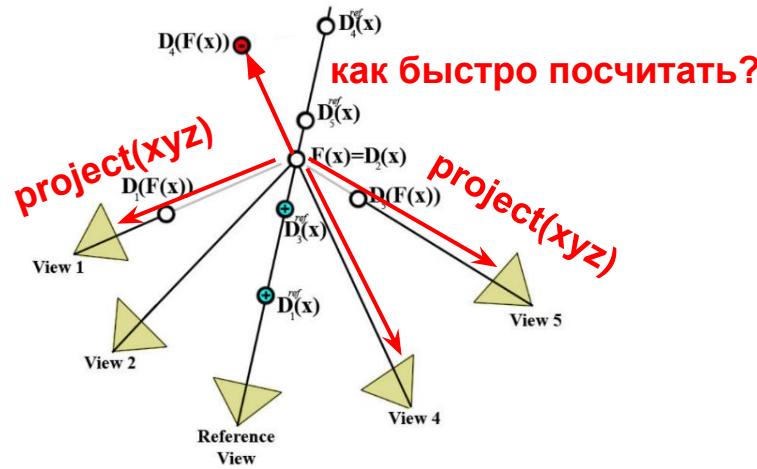
(b) Stability calculation

Figure 2. (a) Visibility relations between points. The point A' seen in view i has its free space violated by A seen in the reference view. B' supports B . C seen in the reference view is occluded by C' . (b) Stability Calculation. In this example, there are two occlusions which raise stability and one free-space violations which lowers it. The stability is +1.

3) Free-space violations - сколько заслоняем мы



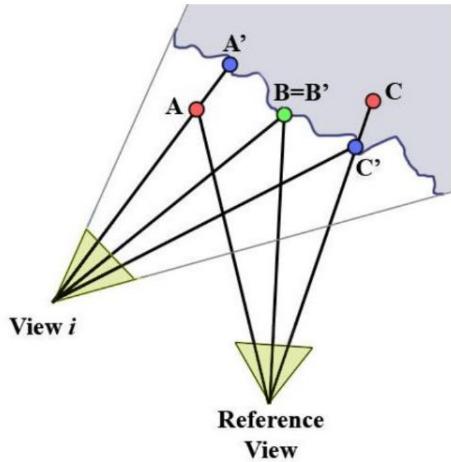
(a) Visibility relations between points



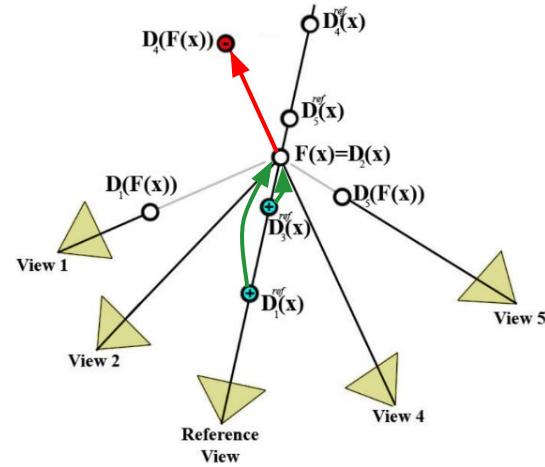
(b) Stability calculation

Figure 2. (a) Visibility relations between points. The point A' seen in view i has its free space violated by A seen in the reference view. B' supports B . C seen in the reference view is occluded by C' . (b) Stability Calculation. In this example, there are two occlusions which raise stability and one free-space violations which lowers it. The stability is +1.

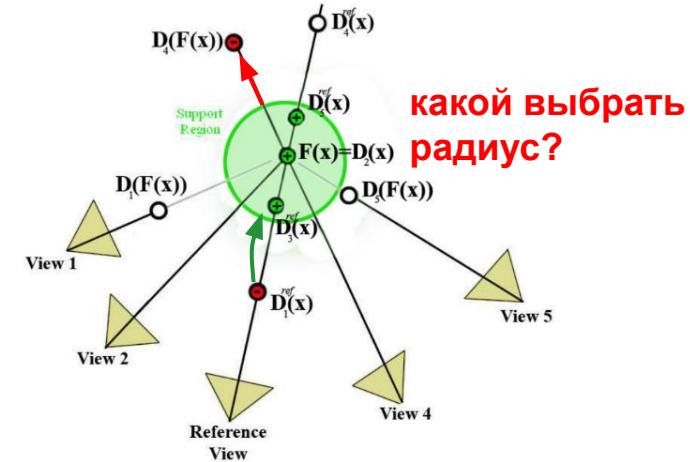
4) Можно склеить гипотезы ради ускорения



(a) Visibility relations between points



(b) Stability calculation



(c) Support estimation

какой выбрать
радиус?

Figure 2. (a) Visibility relations between points. The point A' seen in view i has its free space violated by A seen in the reference view. B' supports B . C seen in the reference view is occluded by C' . (b) Stability Calculation. In this example, there are two occlusions which raise stability and one free-space violations which lowers it. The stability is +1. (c) Support calculation. Three measurements are close to the current estimate and add support to it. Outside the support region, there is one occlusion and one free-space violation which lower the support.

Итого

- 1) Репроецируем все карты глубины в плоскость центральной камеры

The first step of fusion is to render each depth map into the reference view. When multiple depth values project onto the same pixel, the depth is kept. Let D_i^{ref} be the depth map D_i rendered into the reference view and C_i^{ref} be the confidence map rendered in the reference view. Given a

Итого

- 1) Репроецируем все карты глубины в плоскость центральной камеры

The first step of fusion is to render each depth map into the reference view. When multiple depth values project onto the same pixel, ???????????? depth is kept. Let D_i^{ref} be the depth map D_i rendered into the reference view and C_i^{ref} be the confidence map rendered in the reference view. Given a

Итого

- 1) Репроецируем все карты глубины в плоскость центральной камеры

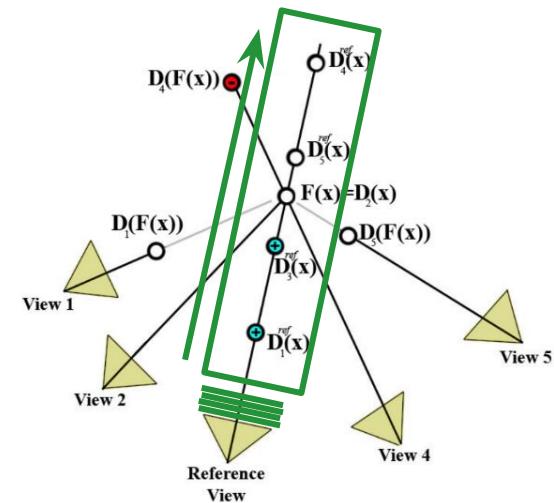
The first step of fusion is to render each depth map into the reference view. When multiple depth values project onto the same pixel, the nearest depth is kept. Let D_i^{ref} be the depth map D_i rendered into the reference view and C_i^{ref} be the confidence map rendered in the reference view. Given a

Итого

- 1) Репроецируем все карты глубины в плоскость центральной камеры
- 2) Для каждого пикселя центральной камеры выполняем фильтрацию:

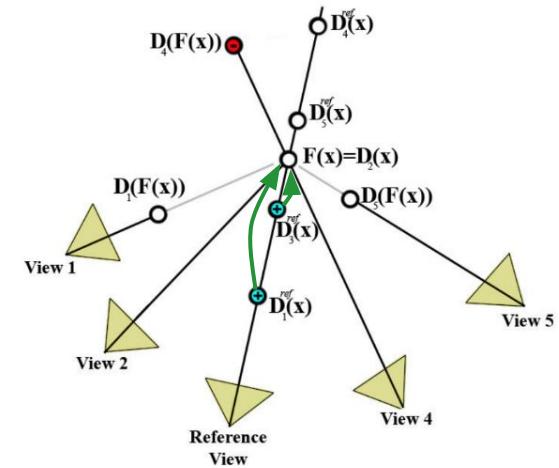
Итого

- 1) Репроецируем все карты глубины в плоскость центральной камеры
- 2) Для каждого пикселя центральной камеры выполняем фильтрацию:
 - 2.1) Сортируем все глубины-кандидаты



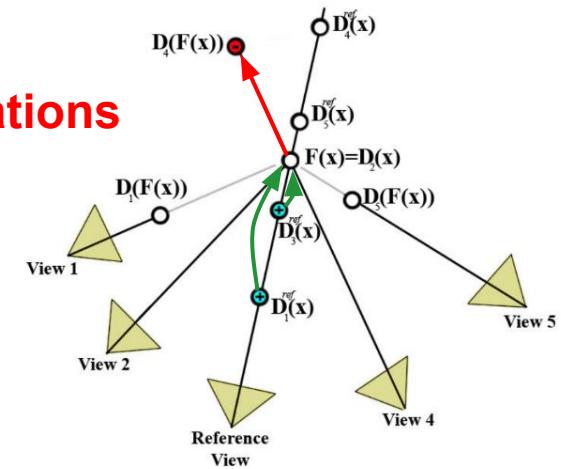
Итого

- 1) Репроецируем все карты глубины в плоскость центральной камеры
- 2) Для каждого пикселя центральной камеры выполняем фильтрацию:
 - 2.1) Сортируем все глубины-кандидаты
 - 2.2) Для каждого из них считаем **Occlusions**



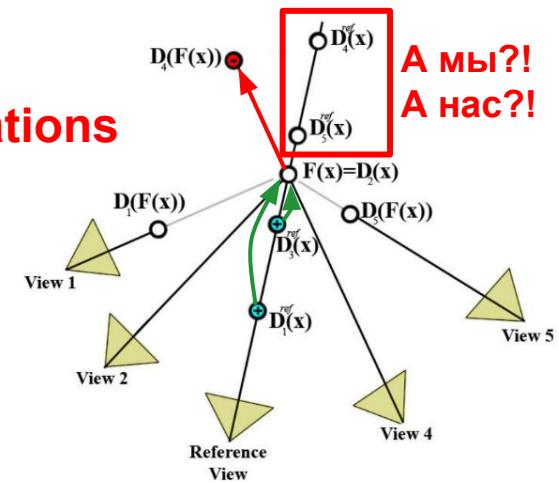
Итого

- 1) Репроецируем все карты глубины в плоскость центральной камеры
- 2) Для каждого пикселя центральной камеры выполняем фильтрацию:
 - 2.1) Сортируем все глубины-кандидаты
 - 2.2) Для каждого из них считаем **Occlusions**
 - 2.3) Для каждого из них считаем **Free-space violations**



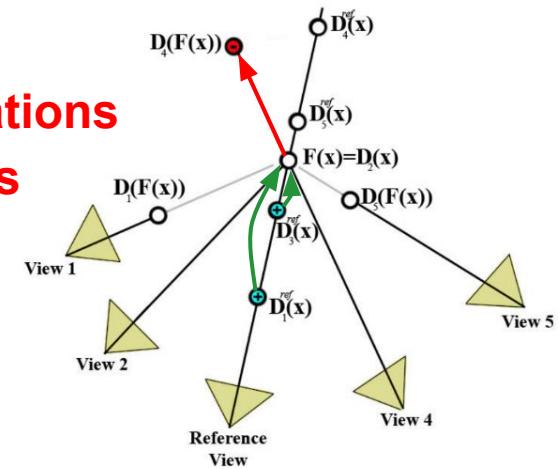
Итого

- 1) Репроецируем все карты глубины в плоскость центральной камеры
- 2) Для каждого пикселя центральной камеры выполняем фильтрацию:
 - 2.1) Сортируем все глубины-кандидаты
 - 2.2) Для каждого из них считаем **Occlusions**
 - 2.3) Для каждого из них считаем **Free-space violations**



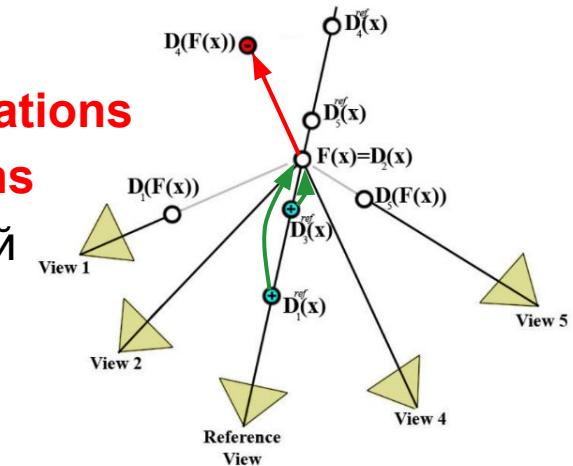
Итого

- 1) Репроецируем все карты глубины в плоскость центральной камеры
- 2) Для каждого пикселя центральной камеры выполняем фильтрацию:
 - 2.1) Сортируем все глубины-кандидаты
 - 2.2) Для каждого из них считаем **Occlusions**
 - 2.3) Для каждого из них считаем **Free-space violations**
 - 2.4) **Stability = Occlusions - Free-space violations**



Итого

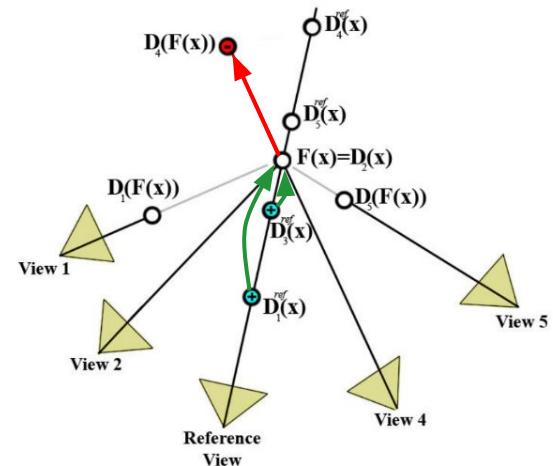
- 1) Репроецируем все карты глубины в плоскость центральной камеры
- 2) Для каждого пикселя центральной камеры выполняем фильтрацию:
 - 2.1) Сортируем все глубины-кандидаты
 - 2.2) Для каждого из них считаем **Occlusions**
 - 2.3) Для каждого из них считаем **Free-space violations**
 - 2.4) **Stability = Occlusions - Free-space violations**
 - 2.5) Победитель: минимальная глубина у которой
Stability >= 0



4.1. Algorithm 1: Stability-Based Fusion

If a depth map occludes a depth hypothesis $\hat{F}(x)$, this indicates that the hypothesis is too far away from the reference view. If the current depth hypothesis violates a free-space constraint, this indicates the hypothesis is too close to the reference view. The stability of a point $S(x)$ is defined as the number of depth maps that occlude $\hat{F}(x)$ minus the number of free-space violations. Stability measures the balance between these two types of visibility violations. A point is stable if the stability is greater than or equal to zero. If the stability is negative, then most of the depth maps indicate that $\hat{F}(x)$ is too close to the camera to be correct. If the stability is positive then at least half of the depth maps indicate that $\hat{F}(x)$ is far enough away from the reference camera. Stability generally increases as the point moves further away from the camera. The final fused depth is selected to be the closest depth to the camera for which stability is non-negative. This depth is not the median depth along the viewing ray since free-space violations are defined on rays that do not come from the reference view. This depth is balanced in the sense that the amount of evidence that indicates it is too close is equal to the amount of evidence that indicates it is too far away.

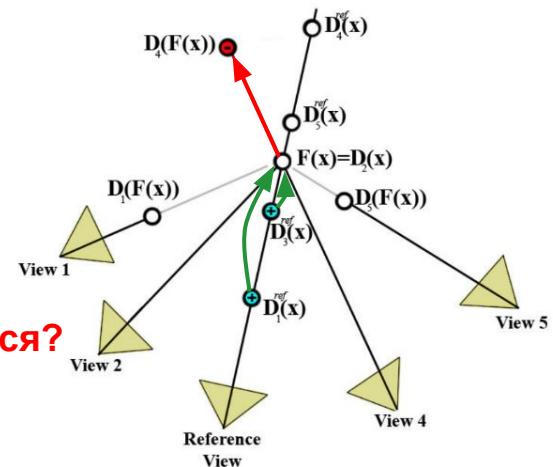
**Stability =
Occlusions - Free-space violations**



4.1. Algorithm 1: Stability-Based Fusion

If a depth map occludes a depth hypothesis $\hat{F}(x)$, this indicates that the hypothesis is too far away from the reference view. If the current depth hypothesis violates a free-space constraint, this indicates the hypothesis is too close to the reference view. The stability of a point $S(x)$ is defined as the number of depth maps that occlude $\hat{F}(x)$ minus the number of free-space violations. Stability measures the balance between these two types of visibility violations. A point is stable if the stability is greater than or equal to zero. If the stability is negative, then most of the depth maps indicate that $\hat{F}(x)$ is too close to the camera to be correct. If the stability is positive then at least half of the depth maps indicate that $\hat{F}(x)$ is far enough away from the reference camera. Stability generally increases as the point moves further away from the camera. The final fused depth is selected to be the closest depth to the camera for which stability is non-negative. This depth is not the median depth along the viewing ray since free-space violations are defined on rays that do not come from the reference view. This depth is balanced in the sense that the amount of evidence that indicates it is too close is equal to the amount of evidence that indicates it is too far away.

**Stability =
Occlusions - Free-space violations**

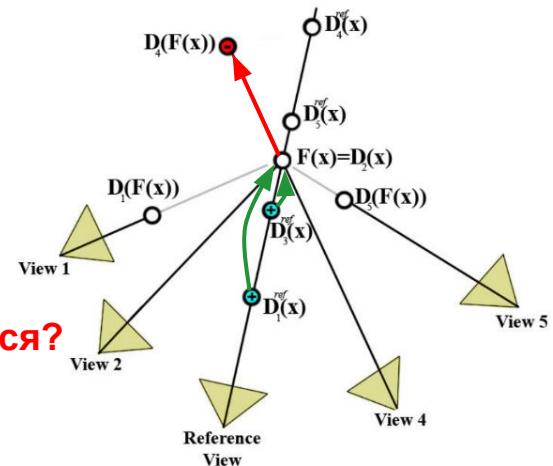


Почему
не всегда
увеличивается?

4.1. Algorithm 1: Stability-Based Fusion

If a depth map occludes a depth hypothesis $\hat{F}(x)$, this indicates that the hypothesis is too far away from the reference view. If the current depth hypothesis violates a free-space constraint, this indicates the hypothesis is too close to the reference view. The stability of a point $S(x)$ is defined as the number of depth maps that occlude $\hat{F}(x)$ minus the number of free-space violations. Stability measures the balance between these two types of visibility violations. A point is stable if the stability is greater than or equal to zero. If the stability is negative, then most of the depth maps indicate that $\hat{F}(x)$ is too close to the camera to be correct. If the stability is positive then at least half of the depth maps indicate that $\hat{F}(x)$ is far enough away from the reference camera. Stability generally increases as the point moves further away from the camera. The final fused depth is selected to be the closest depth to the camera for which stability is non-negative. This depth is not the median depth along the viewing ray since free-space violations are defined on rays that do not come from the reference view. This depth is balanced in the sense that the amount of evidence that indicates it is too close is equal to the amount of evidence that indicates it is too far away.

**Stability =
Occlusions - Free-space violations**



Почему
не всегда
увеличивается?

Проблемный пример



Источник: <https://www.agisoft.com/forum/index.php?topic=3984.0>

Проблемный пример



Источник: <https://www.agisoft.com/forum/index.php?topic=8984.0>

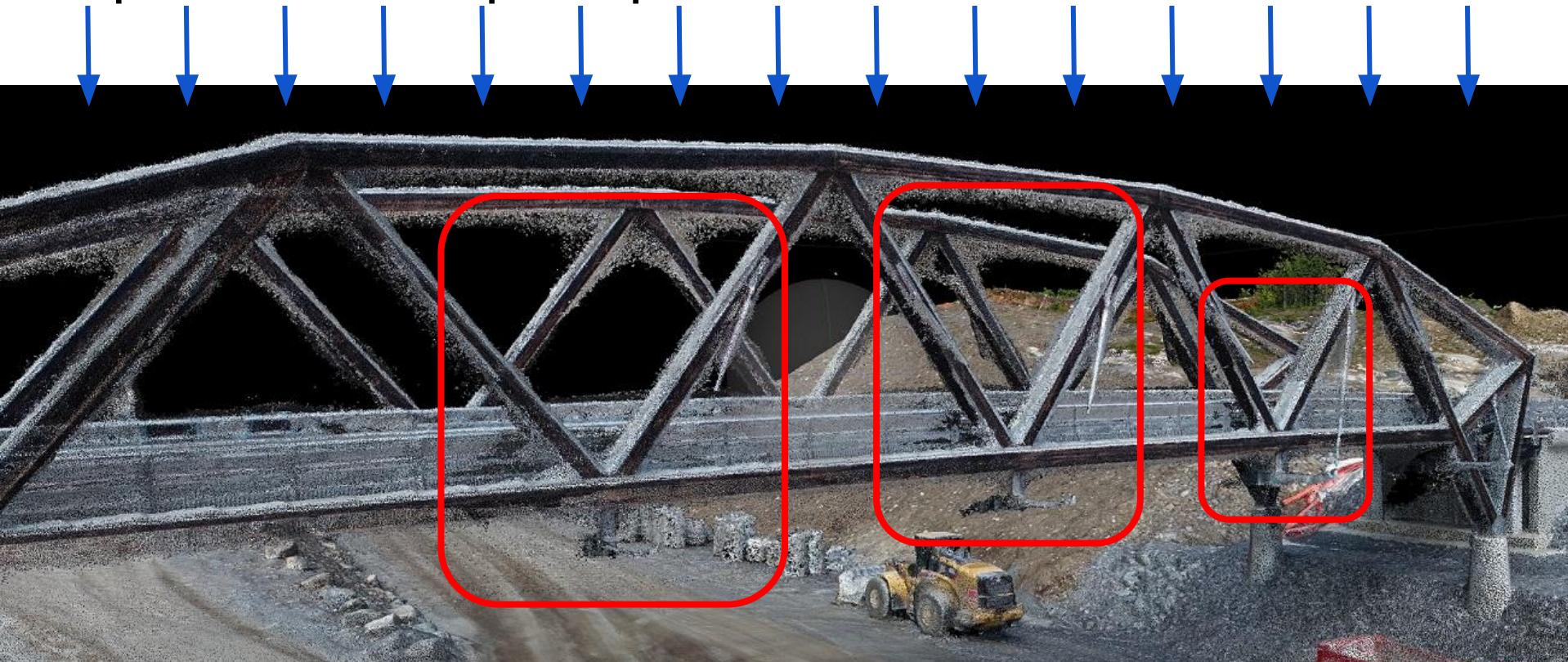
Проблемный пример



Как исследовать проблему? Как искать причину? Как отлаживать?

Источник: <https://www.agisoft.com/forum/index.php?topic=8984.0>

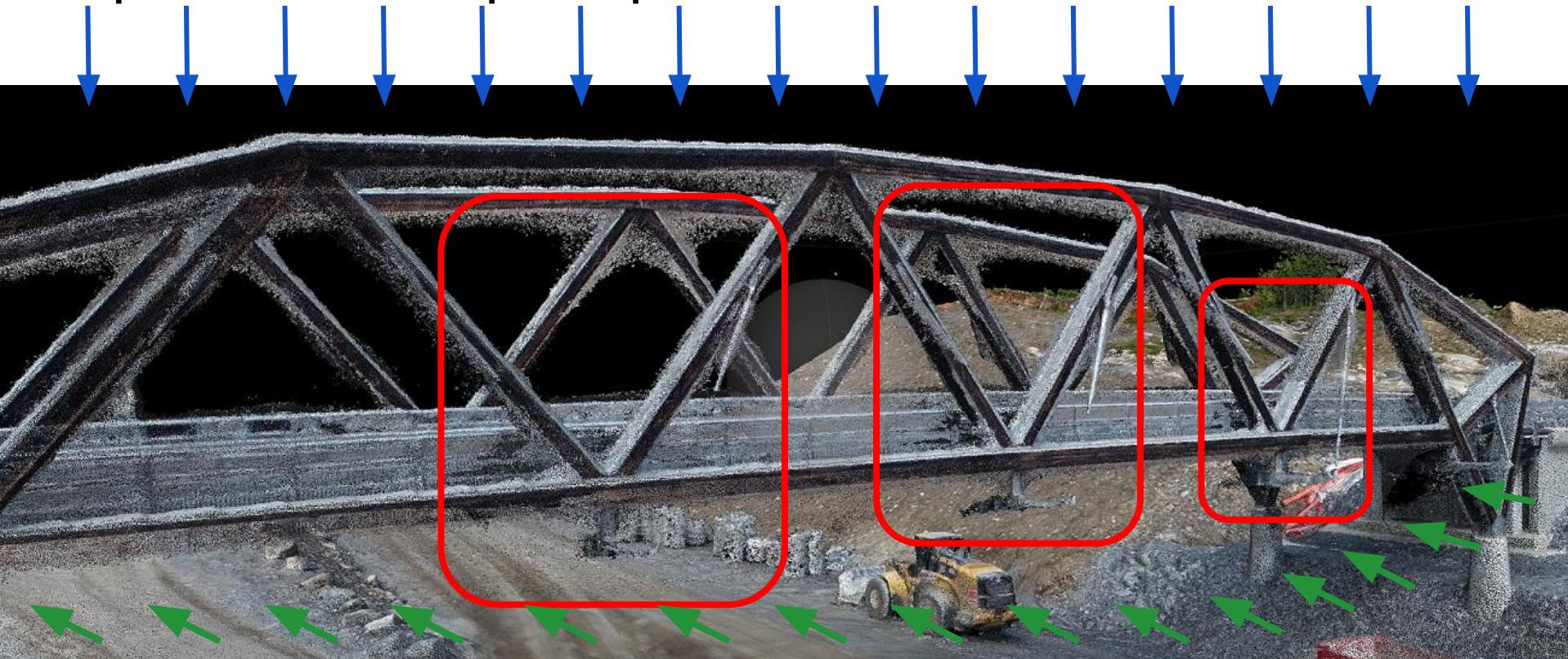
Проблемный пример



Как исследовать проблему? Как искать причину? Как отлаживать?

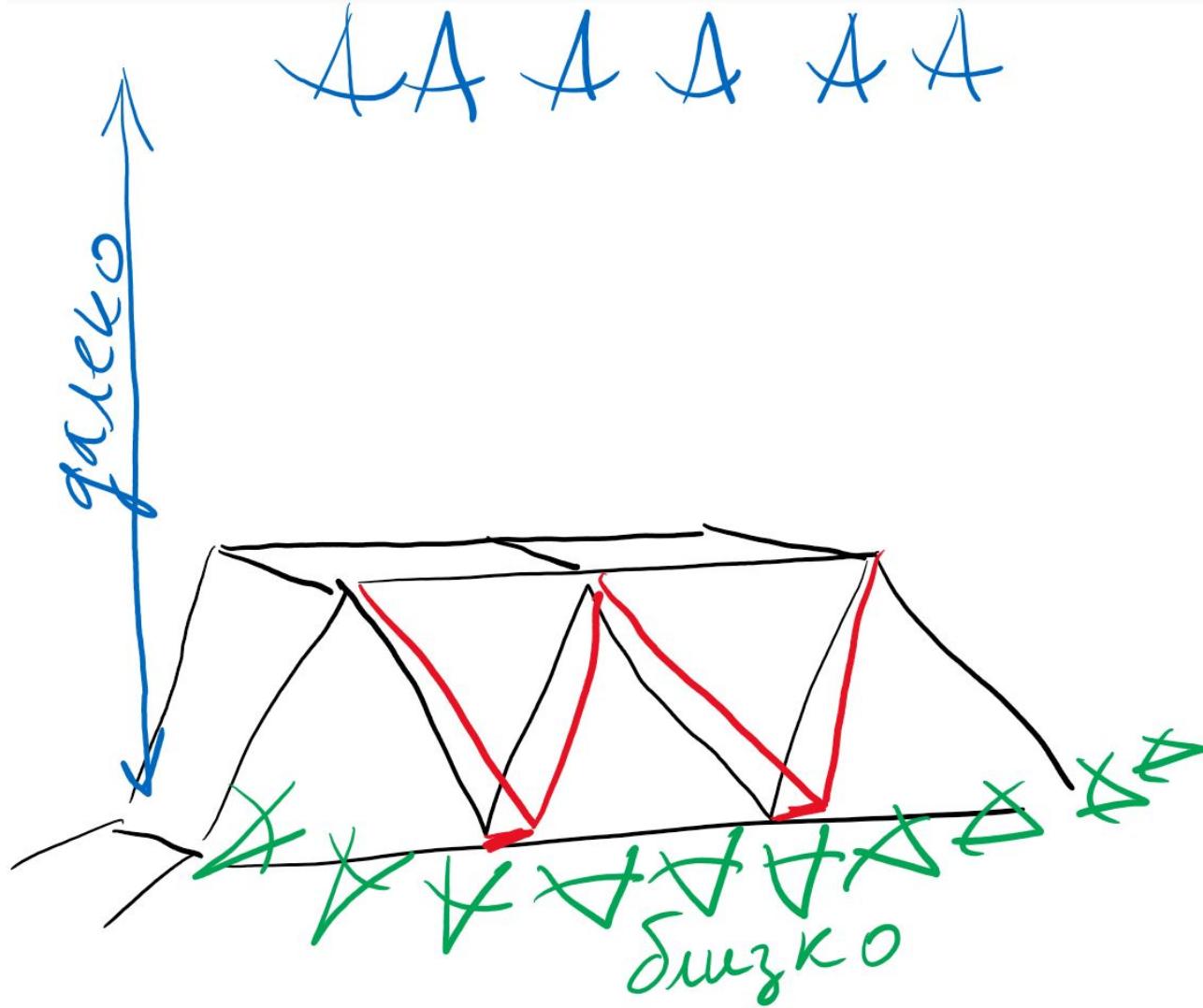
Источник: <https://www.agisoft.com/forum/index.php?topic=8984.0>

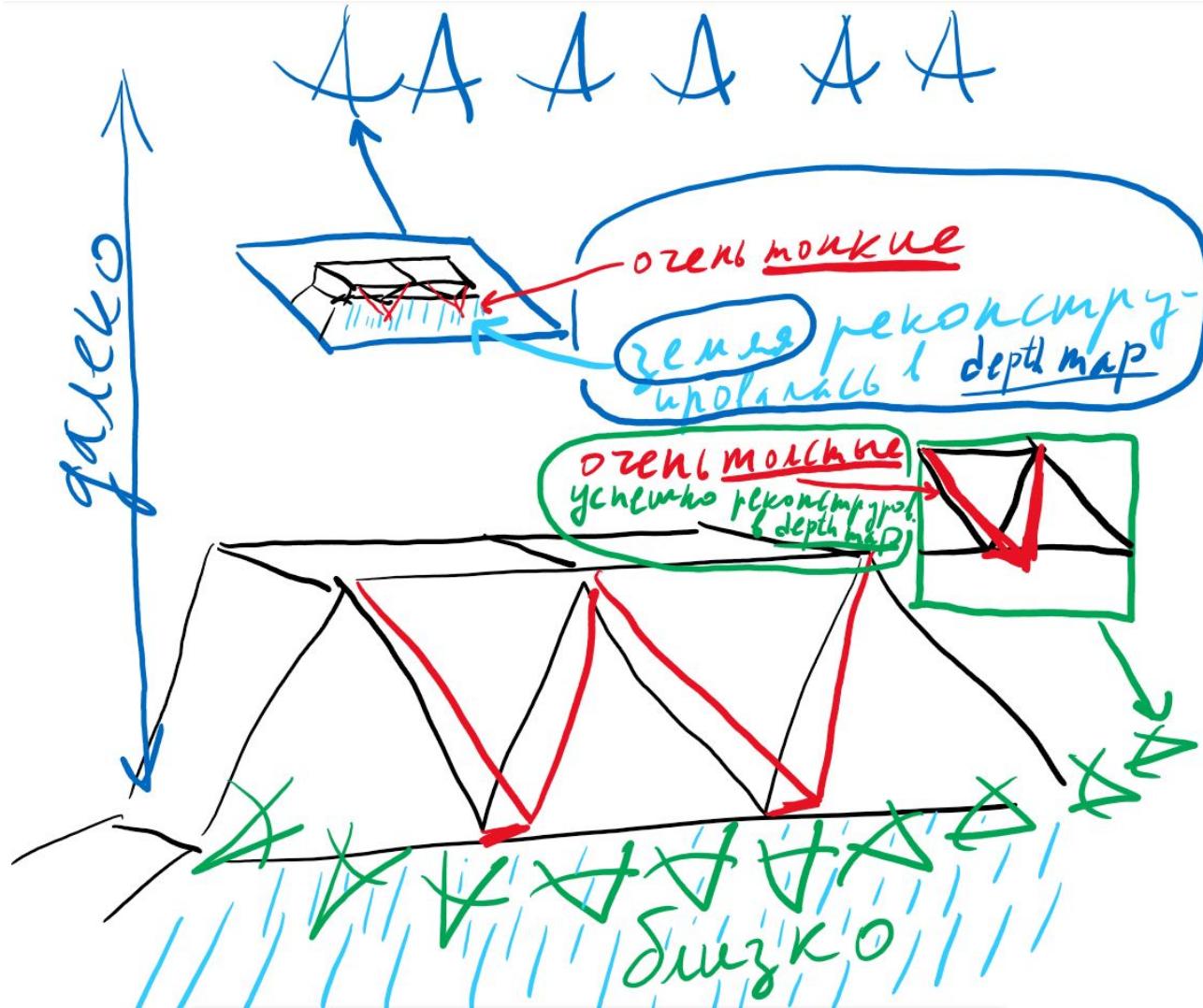
Проблемный пример



Как исследовать проблему? Как искать причину? Как отлаживать?

Источник: <https://www.agisoft.com/forum/index.php?topic=8984.0>





Где мы сейчас?

- Есть точная калибровка камеры и ракурсов фотографирования (intrinsics & extrinsics parameters)
- Есть точное но разреженное (недетальное) облако ключевых точек
- Есть детальная но с ошибками/выбросами карта глубины каждой камеры

Хотим:

- Очистить каждую карту глубины от выбросов
- Совместно отфильтровать карты глубины от выбросов
- **Объединить отфильтрованные карты глубины в облако точек**
- По облаку точек построить полигональную модель

Где мы сейчас?

- Есть точная калибровка камеры и ракурсов фотографирования (intrinsics & extrinsics parameters)
- Есть точное но разреженное (недетальное) облако ключевых точек
- Есть детальная но с ошибками/выбросами карта глубины каждой камеры

Хотим:

- Очистить каждую карту глубины от выбросов
- Совместно отфильтровать карты глубины от выбросов
- Объединить отфильтрованные карты глубины в облако точек
- **По облаку точек построить полигональную модель**

Poisson surface reconstruction

На входе \vec{V} - точки с инвертированными нормалями.



Oriented points

\vec{V}

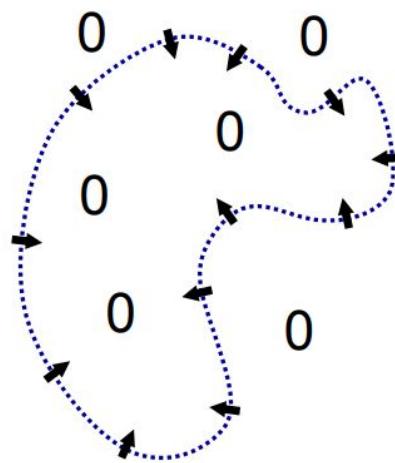
Poisson surface reconstruction

На входе \vec{V} - точки с инвертированными нормалями.

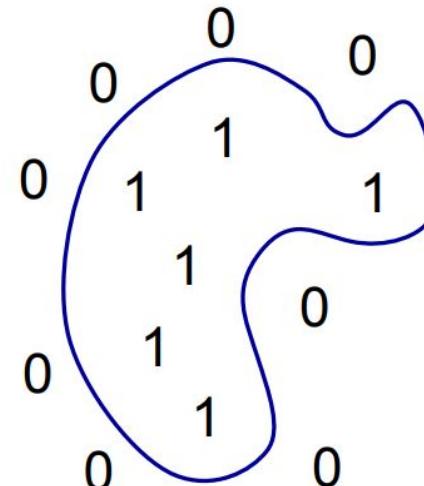
Находим индикатор χ - скалярное поле, чей градиент приближает векторное поле нормалей: $\min_{\chi} \|\nabla \chi - \vec{V}\|$



Oriented points
 \vec{V}



Indicator gradient
 $\nabla \chi_M$



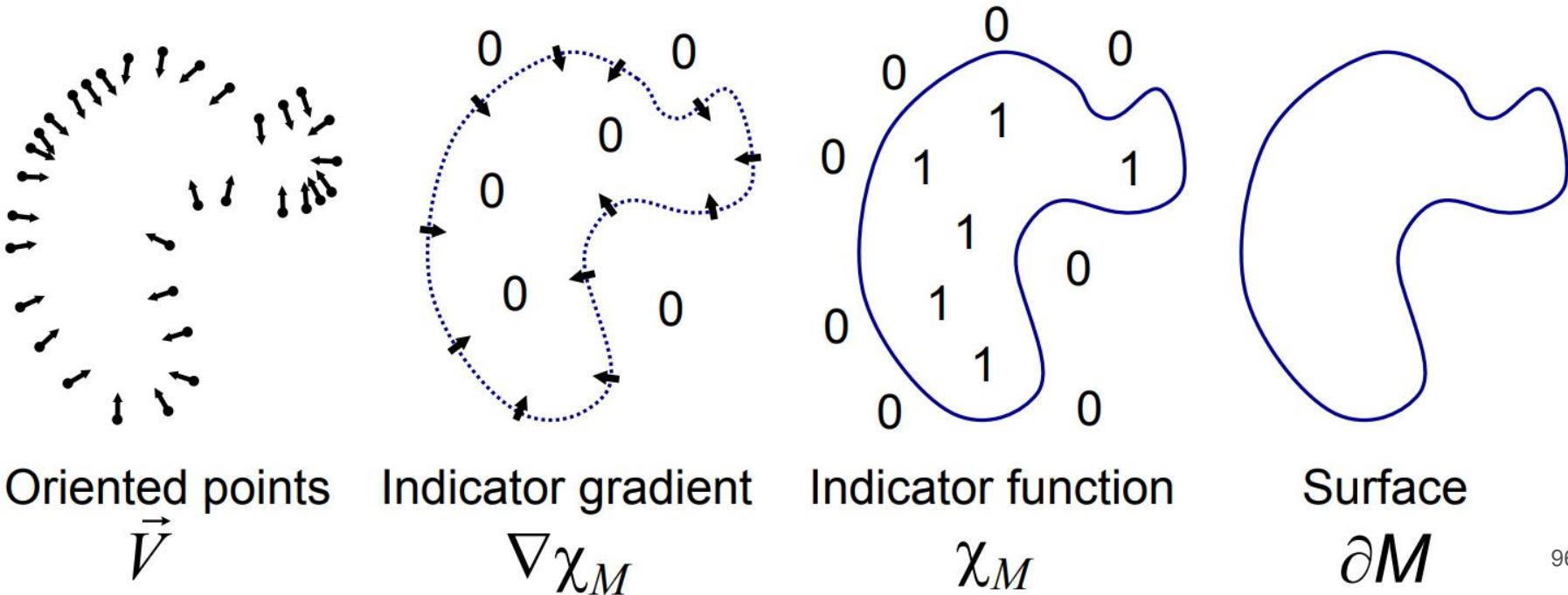
Indicator function
 χ_M

Poisson surface reconstruction

На входе \vec{V} - точки с инвертированными нормалями.

Находим индикатор χ - скалярное поле, чей градиент приближает векторное поле нормалей: $\min_{\chi} \|\nabla \chi - \vec{V}\|$

Результирующая поверхность ∂M - изоповерхность в поле индикатора.



Oriented points
 \vec{V}

Indicator gradient
 $\nabla \chi_M$

Indicator function
 χ_M

Surface
 ∂M

Ссылки

Patch Match для реконструкции волос: [Strand-accurate Multi-view Hair Capture, Nam et. al., 2019](#)

Интересное применение фотограмметрии: [A sandstone block built from lego, blending real objects with 3d prints](#)

Depth Maps filtering:

- [Real-Time Visibility-Based Fusion of Depth Maps, Merrell et. al., 2007](#)

Poisson surface reconstruction:

- [Poisson Surface Reconstruction, Kazhdan et. al., 2006](#)
+ [github.com/PoissonRecon](#)
- [Multilevel Streaming for Out-of-Core Surface Reconstruction, Bolitho et. al., 2007](#)
+ [Code/StreamingRecon_918.zip](#)

Poisson surface reconstruction + GPGPU:

- [Data-Parallel Octrees for Surface Reconstruction, Zhou et al., 2011](#)
- [NV Forum: YouTube: Introduction to CUDA & GPU Poisson Surface Reconstruction](#)
- [YouTube: Introduction to CUDA & GPU Poisson Surface Reconstruction](#)

Вопросы?



Полярный Николай
polarnick239@gmail.com