Самая лучшая лекция. Реконструкция 3D модели из карт глубины. (вариационными методами)

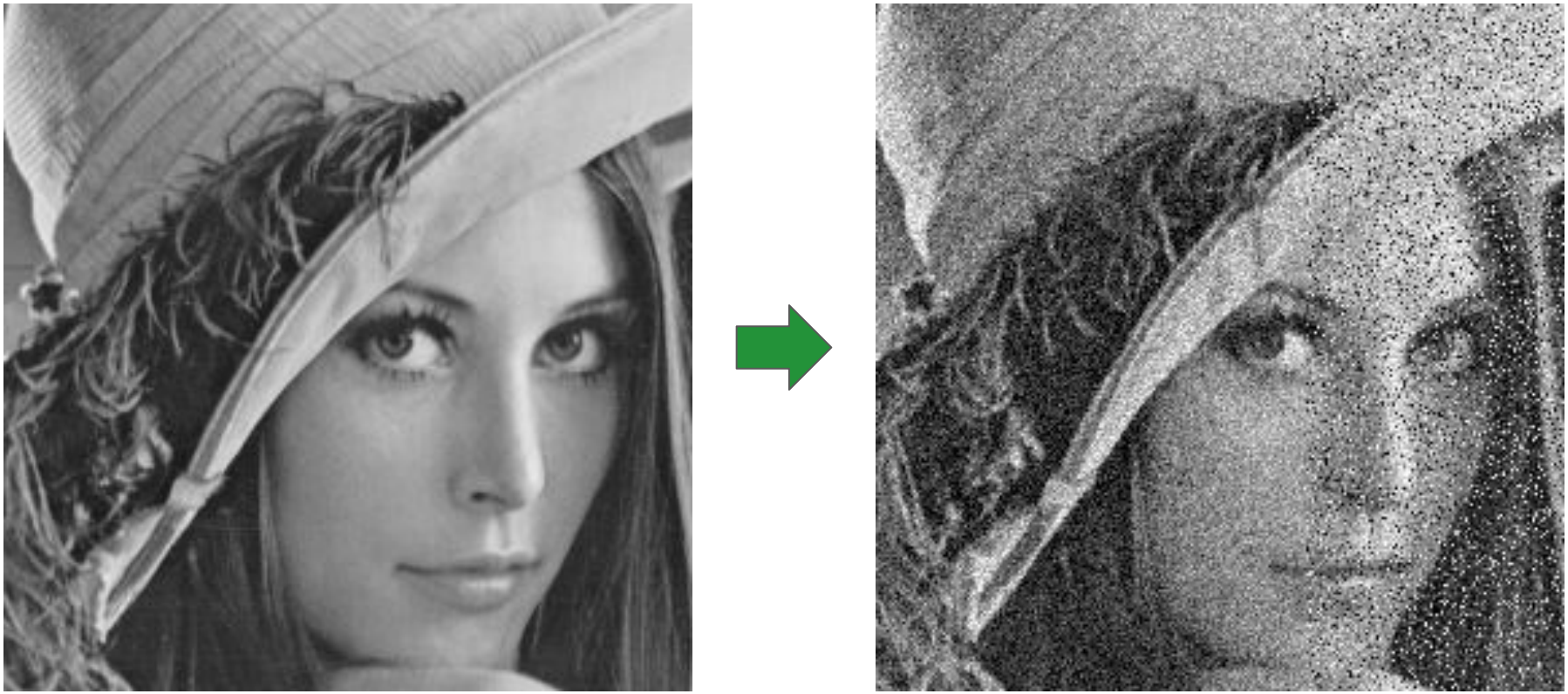
Фотограмметрия. Лекция 10

- **Denoising**
- **Super resolution**
- **2.5D model reconstruction**
- **3D model reconstruction**
- **3D Out-of-Core model reconstruction**

Полярный Николай
polarnick239@gmail.com

Пусть есть шумное изображение

- Гауссовый шум по всей поверхности изображения
- Salt-and-pepper шум на правой части изображения



ROF Image Denoising (TV-L2)

Rudin-Osher-Fatemi model (ROF):

$$\min_x \| \nabla x \| + \frac{\lambda}{2} \| x - f \|^2$$

Где

- f - наблюдаемое изображение с шумом
- x - искомое очищенное от шума изображение
- $\| \nabla x \|$ - полная вариация (регуляризация)
- $\| x - f \|^2$ - L^2 тяготение к данным (наблюдаемому f)
- λ - параметр регуляризации

Как найти x?

Primal-Dual algorithm (TV-L2, ROF)

Primal-Dual алгоритм минимизации в общем виде решает:

$$\min_x F(Kx) + G(x)$$

Где: F G- выпуклые функции, K- линейный оператор.

Тогда итеративная схема (модель - выпуклая):

$$\begin{cases} p_{n+1} = (I + \sigma \partial F^*)^{-1}(p_n + \sigma K x_n) \\ \hat{x}_{n+1} = (I + \tau \partial G)^{-1}(x_n - \tau K^T p_{n+1}) \\ x_{n+1} = \hat{x}_{n+1} + \theta(\hat{x}_{n+1} - x_n) \end{cases}$$

$$(I + \sigma \partial F^*)^{-1}(p) = \mathbf{project}_P(p)$$

$$(I + \tau \partial G_{ROF})^{-1}(x) = \frac{x + \lambda \tau f}{1 + \lambda \tau}$$

$$\mathbf{project}_P(p) = \frac{p}{max(\| p \|, 1)}$$

$$K x_n = \nabla x_n$$

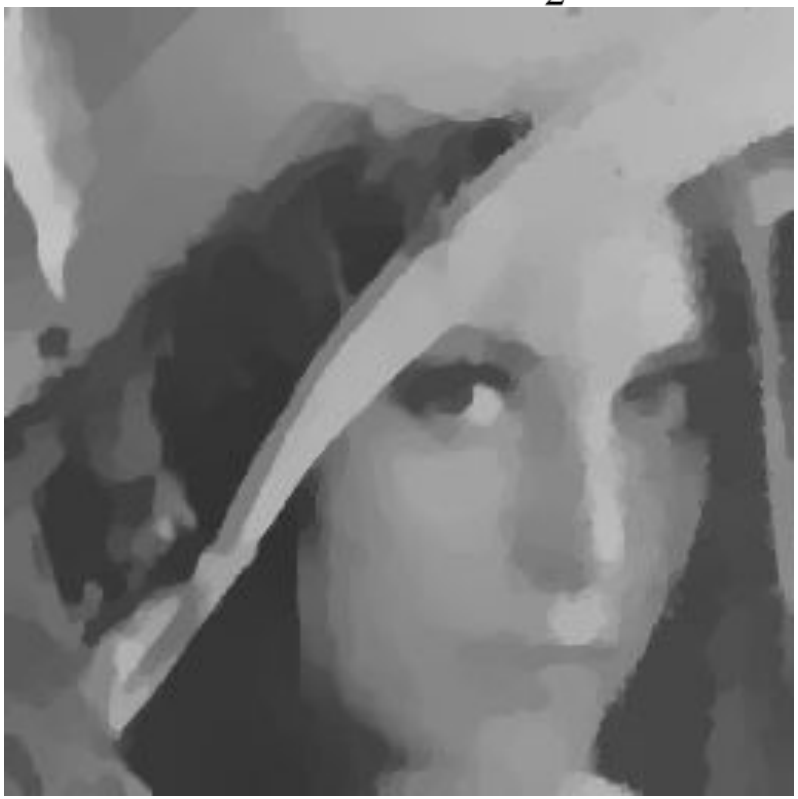
$$K^T p_{n+1} = \nabla^T p_{n+1}$$

Подробнее: [IPython notebook with ROF and TVL1 denoising (with math!)]

ROF Image Denoising (TV-L2)

Rudin-Osher-Fatemi model (ROF):

$$\min_x \| \nabla x \| + \frac{\lambda}{2} \| x - f \|^2$$



$\lambda = 4$

$\lambda = 8$

Image Denoising (TV-L1)

TV-L1 model:

$$\min_x \| \nabla x \| + \lambda \| x - f \|$$

Т.е. тяготение к шумам гораздо слабее.

Primal-Dual algorithm (TV-L1)

Primal-Dual алгоритм минимизации в общем виде решает:

$$\min_x F(Kx) + G(x)$$

Где: F G- выпуклые функции, K- линейный оператор.

Тогда итеративная схема (модель - выпуклая):

$$\begin{cases} p_{n+1} = (I + \sigma \partial F^*)^{-1}(p_n + \sigma K x_n) & (I + \sigma \partial F^*)^{-1}(p) = \mathbf{project}_P(p) \\ \hat{x}_{n+1} = (I + \tau \partial G)^{-1}(x_n - \tau K^T p_{n+1}) & (I + \tau \partial G_{TV-L1})^{-1}(x) = \mathbf{shrink}(x, f, \lambda\tau) \\ x_{n+1} = \hat{x}_{n+1} + \theta(\hat{x}_{n+1} - x_n) \end{cases}$$

$$\mathbf{project}_P(p) = \frac{p}{max(\| p \|, 1)}$$

$$\mathbf{shrink}(x, f, \lambda\sigma) = \begin{cases} x - \lambda\sigma & x > f + \lambda\sigma \\ x + \lambda\sigma & x < f - \lambda\sigma \\ f & |x - f| \le \lambda\sigma \end{cases}$$

$$Kx_n = \nabla x_n$$

$$K^T p_{n+1} = \nabla^T p_{n+1}$$

Подробнее: [IPython notebook with ROF and TVL1 denoising (with math!)]

Image Denoising (TV-L1)

TV-L1 model:

$$\min_x \| \nabla x \| + \lambda \| x - f \|$$

$\lambda = 1$

$\lambda = 1$

Image Denoising (TV-L1, **много наблюдений**)

TV-L1 model:

$$\min_x \| \nabla x \| + \lambda \sum_i \| x - f_i \|$$

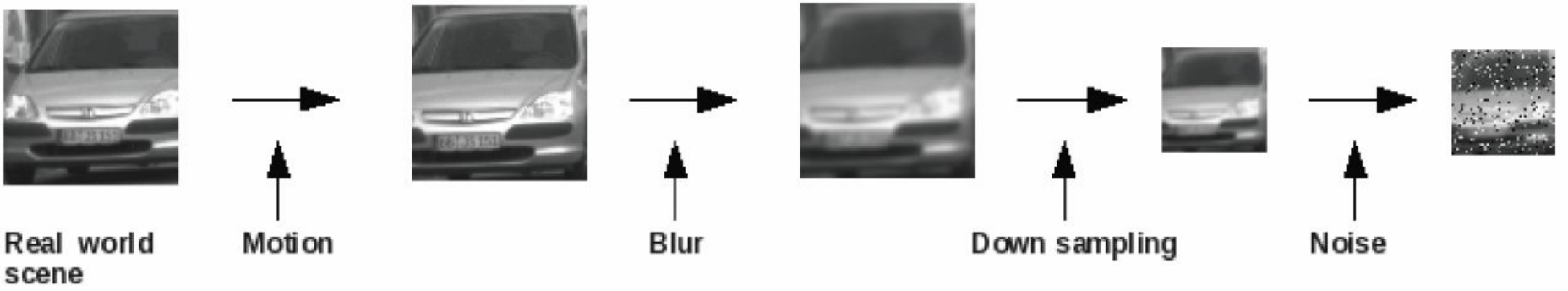
5 очень шумных наблюдений:

Подробнее: [IPython notebook with ROF and TVL1 denoising (with math!)]

Ссылки

- An introduction to Total Variation for Image Analysis, Chambolle et al., 2009
- IPython notebook with ROF and TVL1 denoising (with math!)
- Google Scholar: Thomas Pock (lots of research with Primal Dual method)

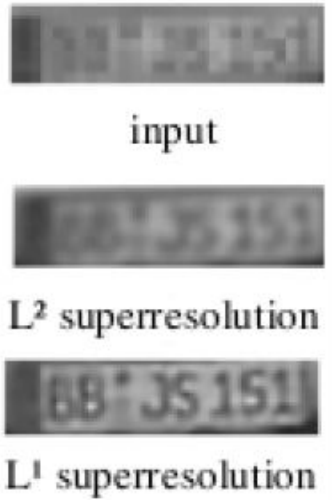
Image super resolution from multiple images
TV-L2, TV-L1




input


L² superresolution


L¹ superresolution



Подробнее: Video Super Resolution using Duality Based TV-L1 Optical Flow, Mitzel et al., 2009

Huber loss

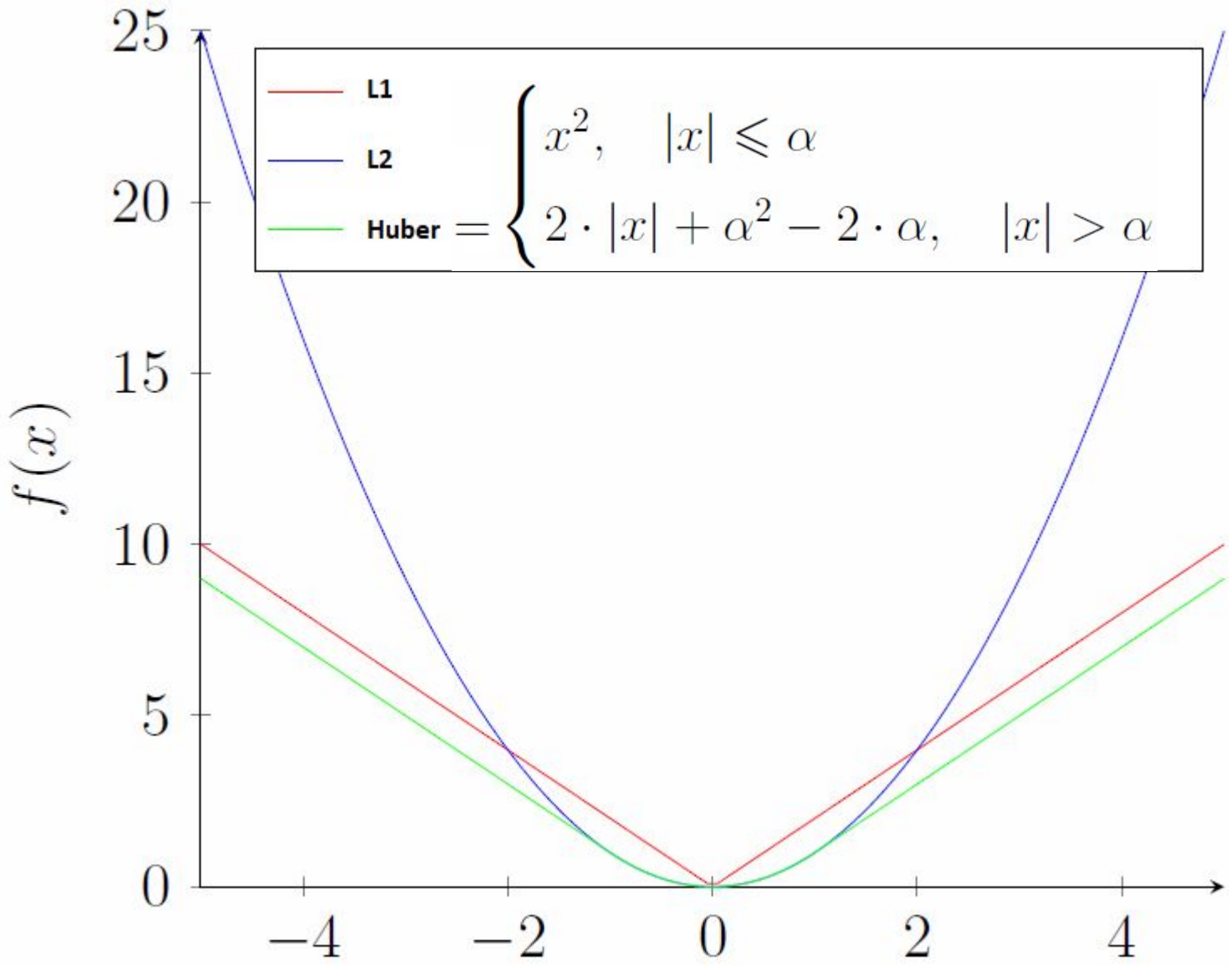

L1
L2
Huber
$= \begin{cases} x^2, & |x| \leqslant \alpha \\ 2 \cdot |x| + \alpha^2 - 2 \cdot \alpha, & |x| > \alpha \end{cases}$
$f(x)$
25
20
15
10
5
0
−4
−2
0
2
4

Image super resolution from multiple images

Huber model:

$$\min_x \| \nabla x \|^h + \lambda \| x - f \|^h$$

Подробнее: A Convex Approach for Variational Super-Resolution, Unger et al., 2010

Digital **S**urface **M**odel reconstruction

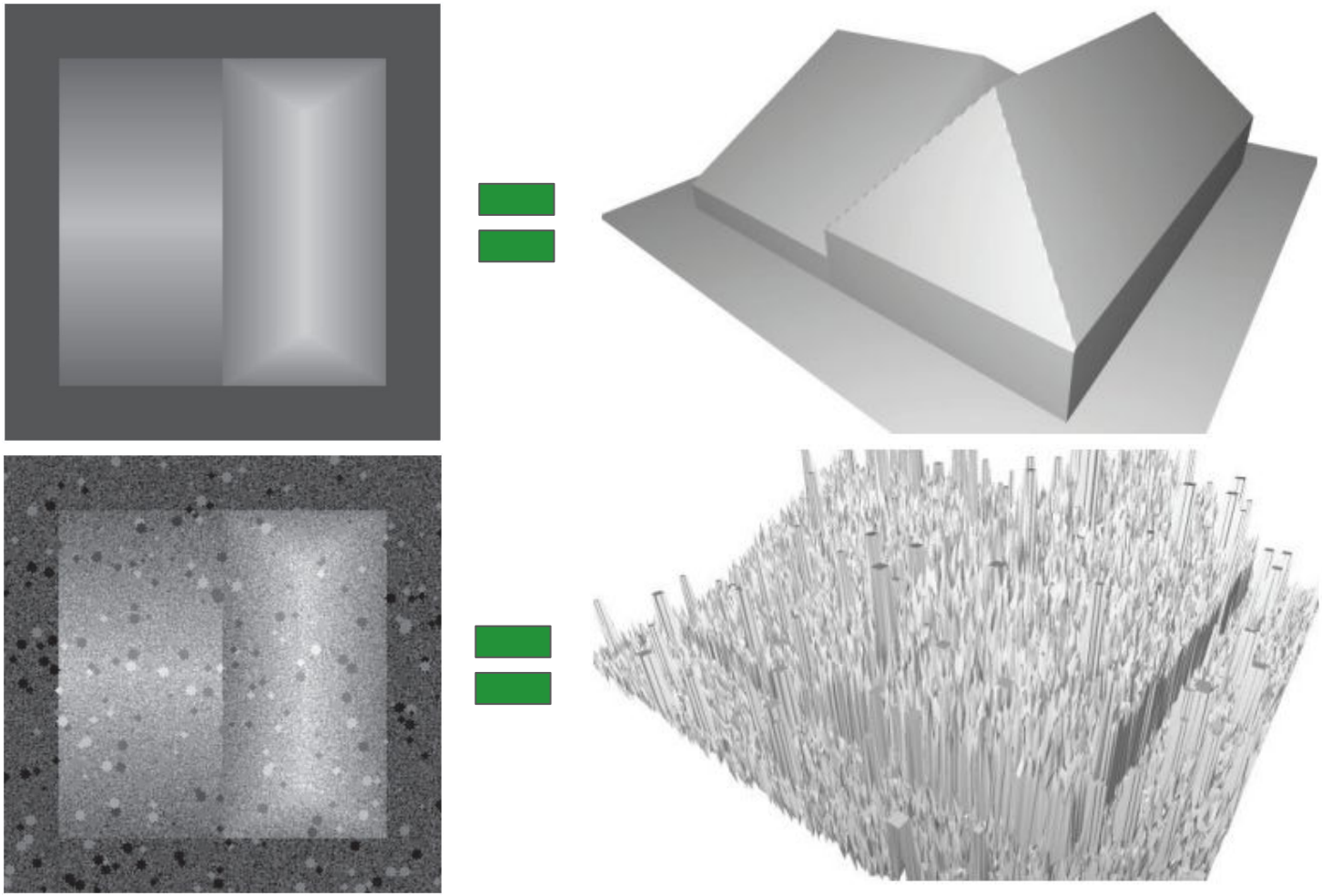
В image super resolution:

- Много шумных наблюдений - **много шумных картинок**
- Нужно найти одну наиболее правдоподобную - **четкую чистую картинку**

В случае **2.5D** реконструкции поверхности ландшафта (**DSM**):

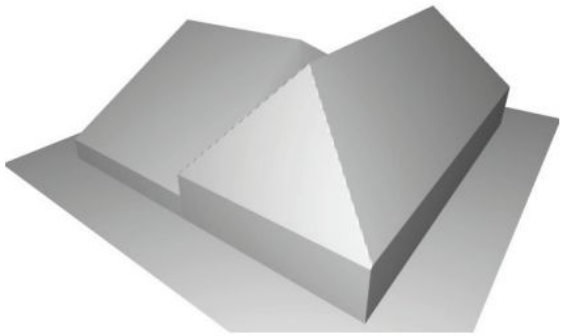
- Много шумных наблюдений - **шумные множества 2.5D точек**
  (на каждый сканер/фотографию - свое множество точек)
- Нужно найти одну наиболее правдоподобную - **2.5D карту высот**
  (т.е. как картинка, но в каждом пикселе - высота ландшафта в этой точке)

DSM (**D**igital **S**urface **M**odel, 2.5D height map)

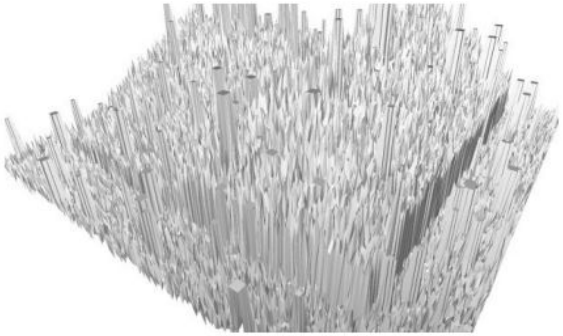


Gussian noise + outliers

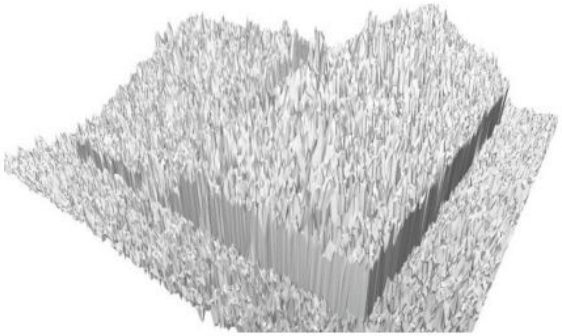
DSM from 5 observations (from 5 noisy height maps)



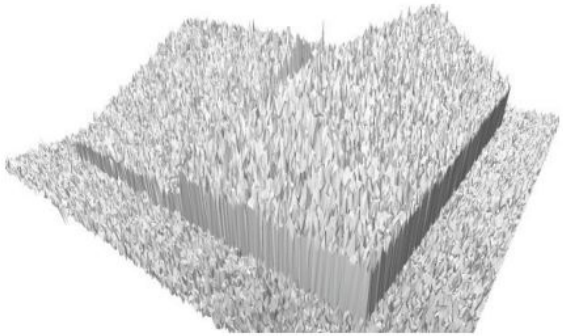
(a) Ground truth



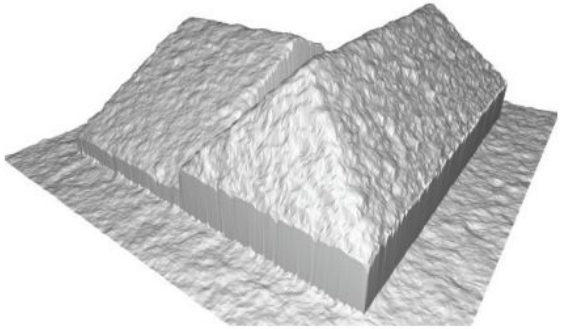
(b) Input image



(c) Average



(d) Median



(e) Huber

(f) TGV^2

TV-L1 -> Huber model -> TGV-Huber model

$TV\text{-}L^1$ model:

$$\min_u \left\{ \alpha \int_\Omega |\nabla u| dx + \sum_{l=1}^{K} \int_\Omega |u - f_l| \, dx \right\}$$

Huber model:

$$\min_u \left\{ \alpha \int_\Omega |\nabla u|_\varepsilon \, dx + \sum_{l=1}^{K} \int_\Omega |u - f_l|_\delta \, dx \right\}$$

TGV^2 **model:**

$$\min_{u,v} \left\{ \alpha_1 \int_\Omega |\nabla u - v| dx + \alpha_0 \int_\Omega |\mathcal{E}(v)| dx + \sum_{l=1}^{K} \int_\Omega |u - f_l|_\delta \, dx \right\}$$

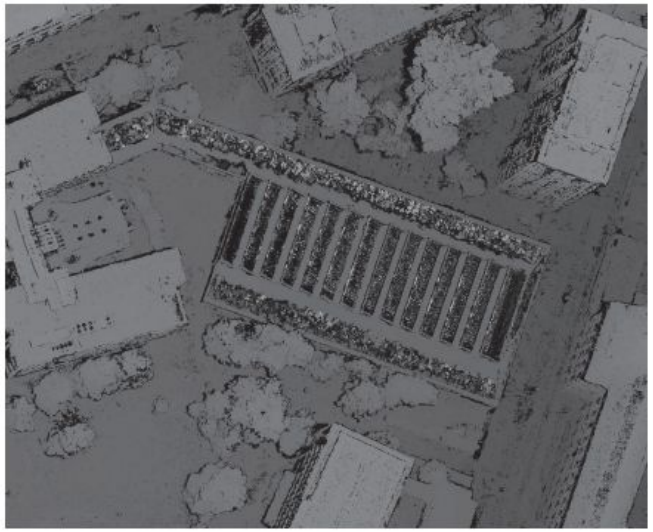
Т.е. регуляризация второго порядка:

$\alpha_1 \int_\Omega |\nabla u - v| dx$ => $v \approx \nabla u$

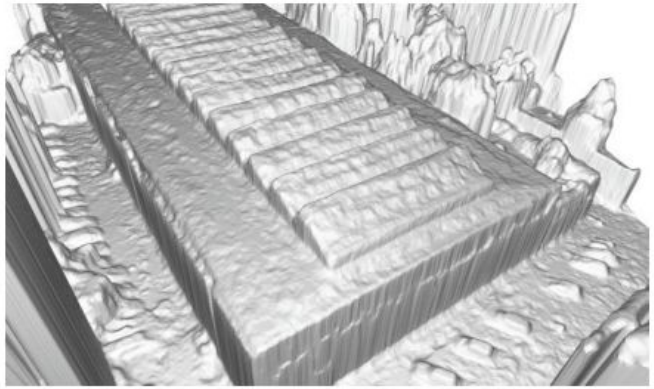
$\alpha_0 \int_\Omega |\mathcal{E}(v)| dx$ - полная вариация v ≈ полная вариация ∇u

Real-world examples

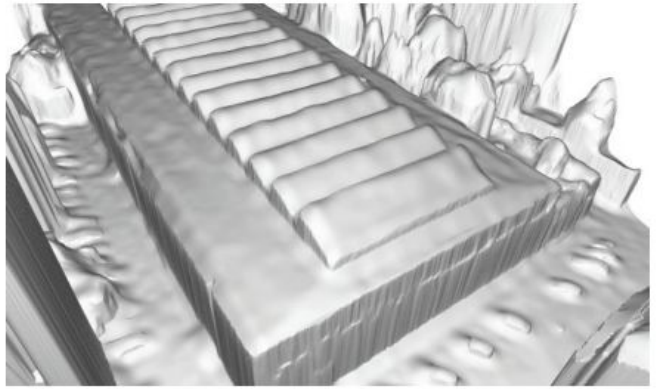
(a)



(b)



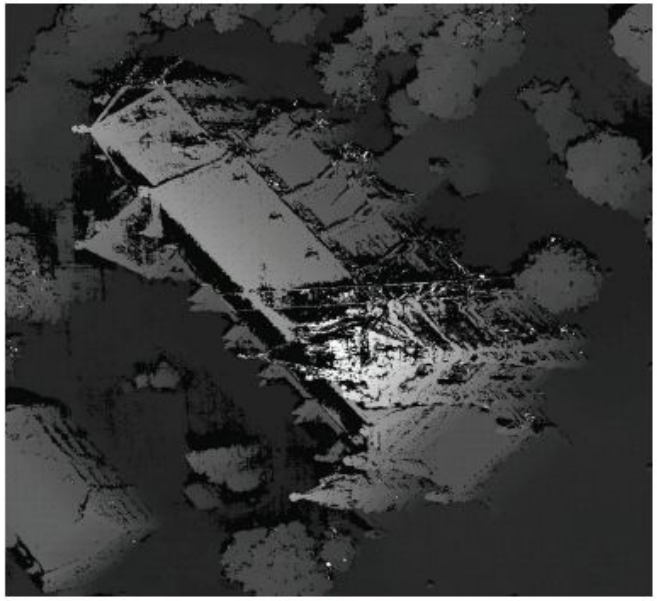
(c) Huber model



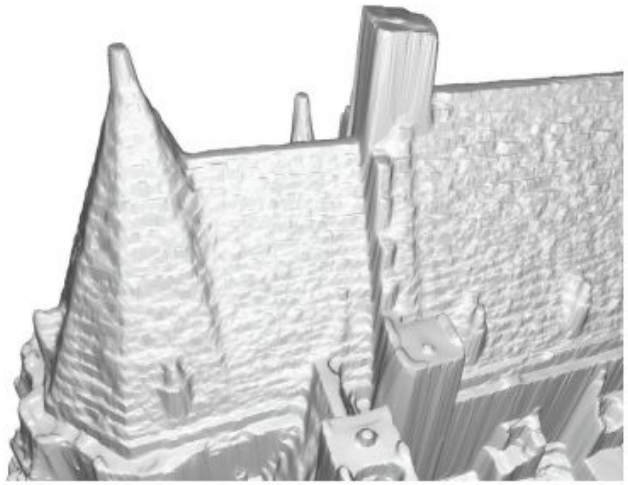
(d) TGV^2

TGV-Fusion, Pock et al., 2011

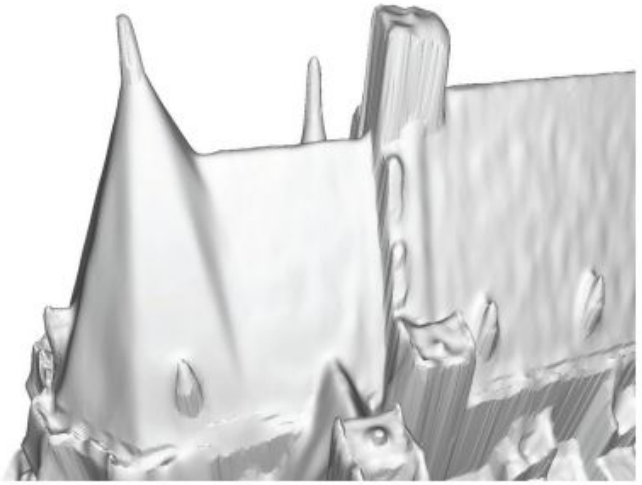
(a)



(b)



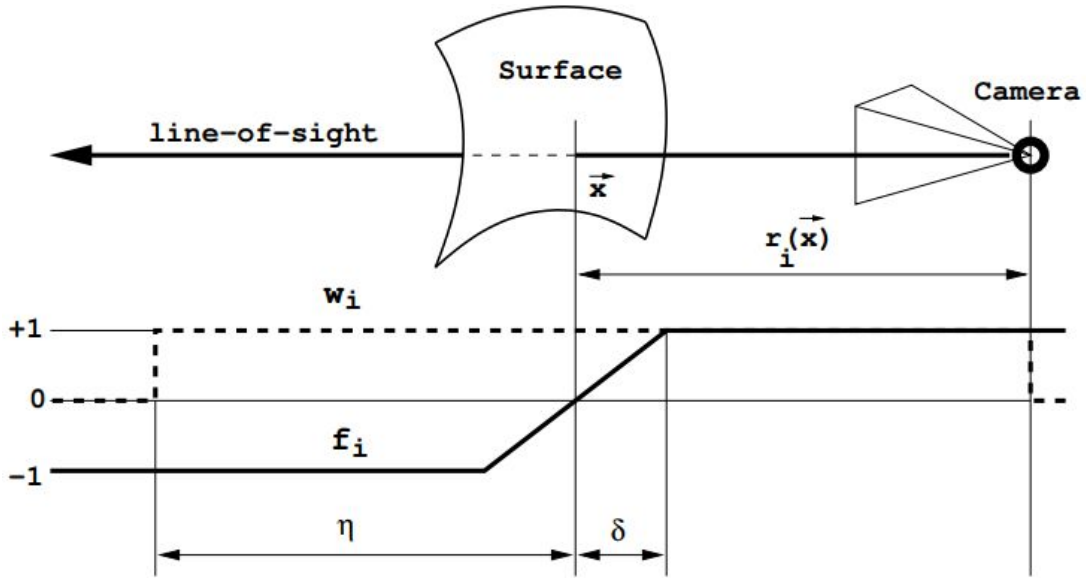
(c) Huber model



(d) TGV^2

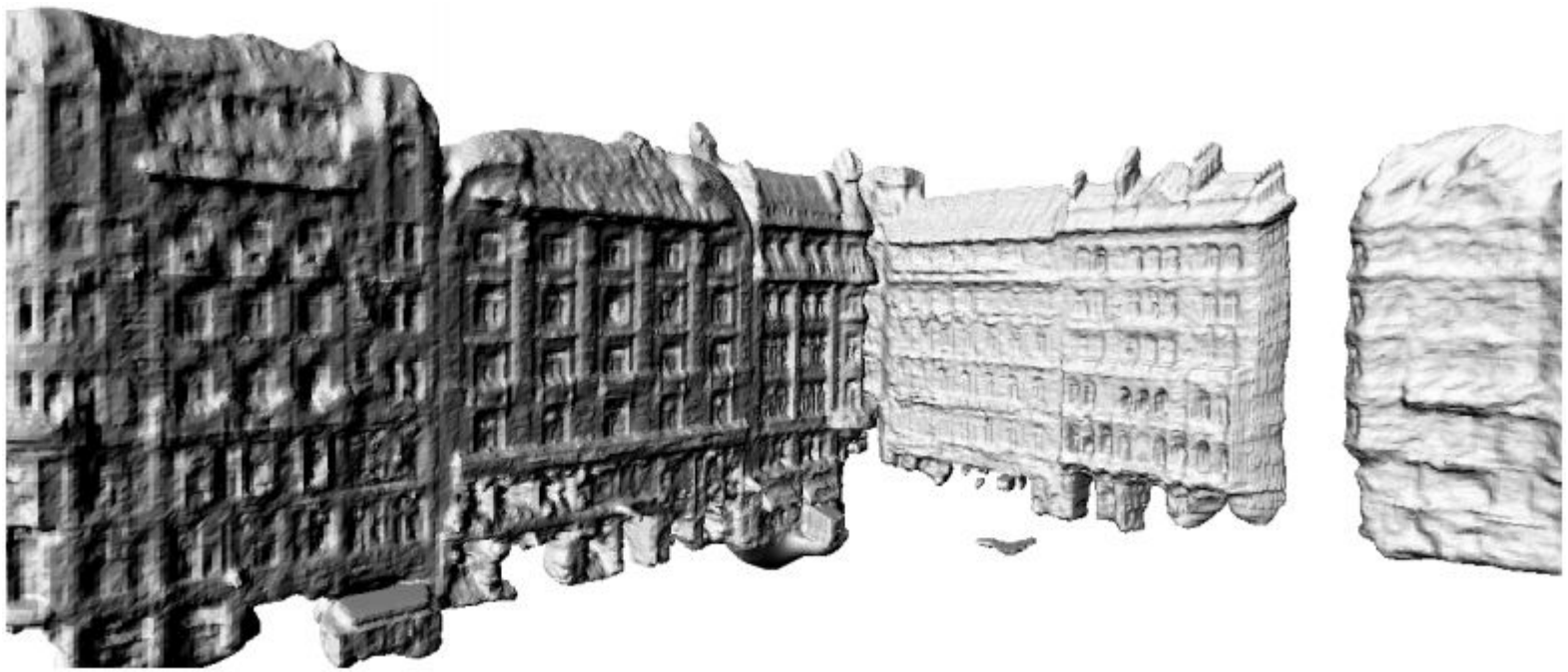
TGV-Fusion, Pock et al., 2011

TV-L1 for **3D** model



$$E = \int_{\Omega} \left\{ |\nabla u| + \lambda \sum_{i \in \mathcal{I}(\vec{x})} w_i(\vec{x}) |u - f_i| \right\} d\vec{x}$$

A Globally Optimal Algorithm for Robust TV-L1 Range Image Integration, Zach et al., 2007



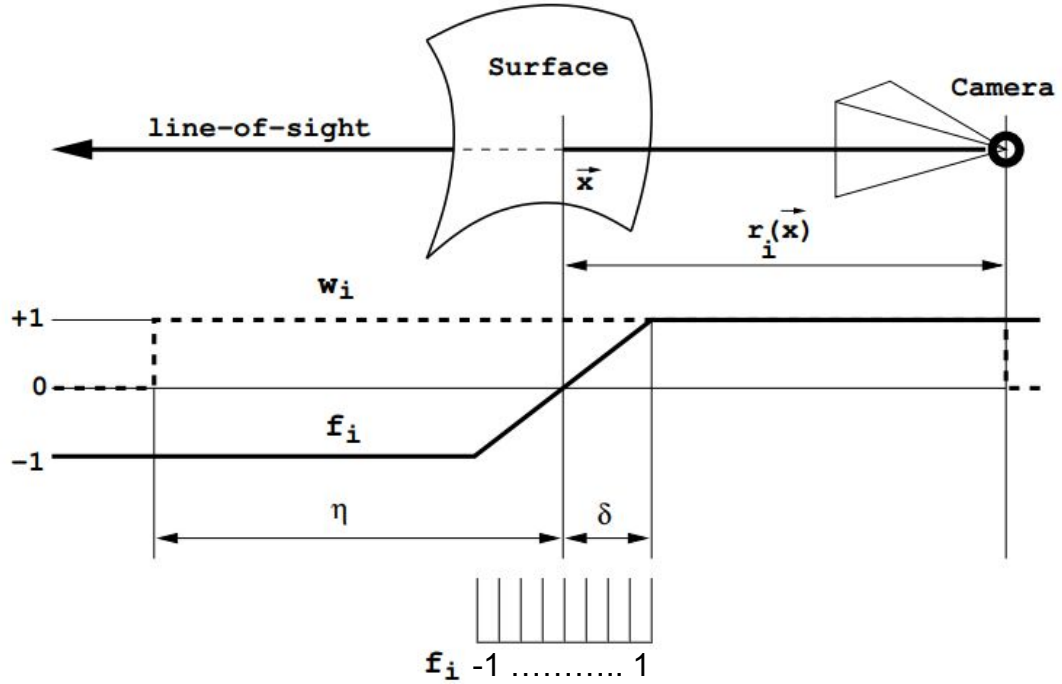
A Globally Optimal Algorithm for Robust TV-L1 Range Image Integration, Zach et al., 2007

3D Model reconstruction

Как убрать потребность держать в видеопамяти все наблюдения для обрабатываемого региона?

Можно сохранить релевантную информацию обо всех наблюдениях в каждом кубике регулярной решетке (вокселе).

Как это сделать? Гистограммами:



Fast and High Quality Fusion of Depth Maps, Zach, 2008

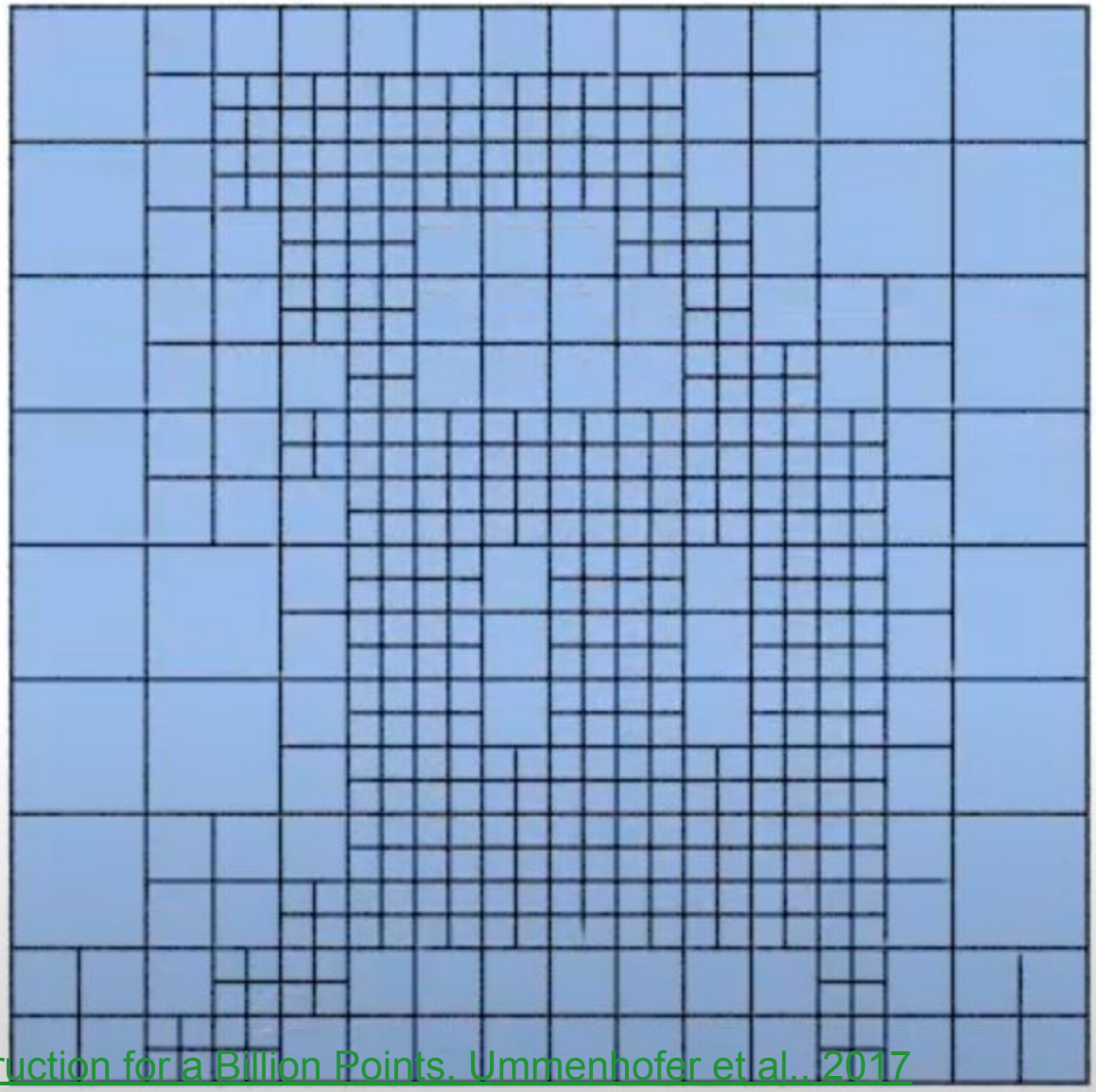
3D Model reconstruction (для миллиарда точек)

1) Адаптивное октодерево:

Global, Dense Multiscale Reconstruction for a Billion Points, Ummenhofer et al., 2017

3D Model reconstruction (для миллиарда точек)

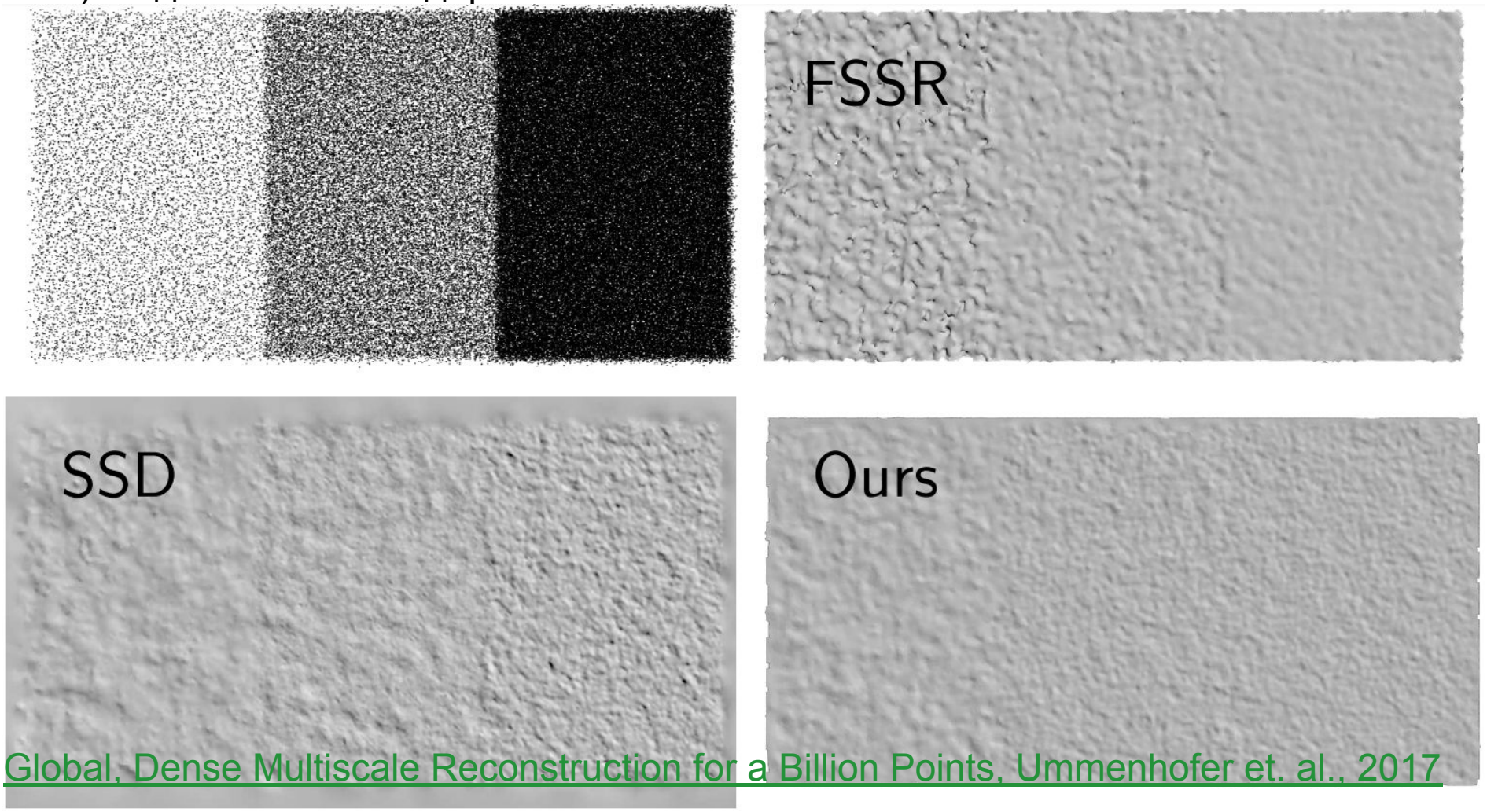
1) Адаптивное октодерево:



Global, Dense Multiscale Reconstruction for a Billion Points, Ummenhofer et al., 2017

3D Model reconstruction (для миллиарда точек)

1) Адаптивное октодерево:



Global, Dense Multiscale Reconstruction for a Billion Points, Ummenhofer et. al., 2017

3D Model reconstruction (для миллиарда точек)

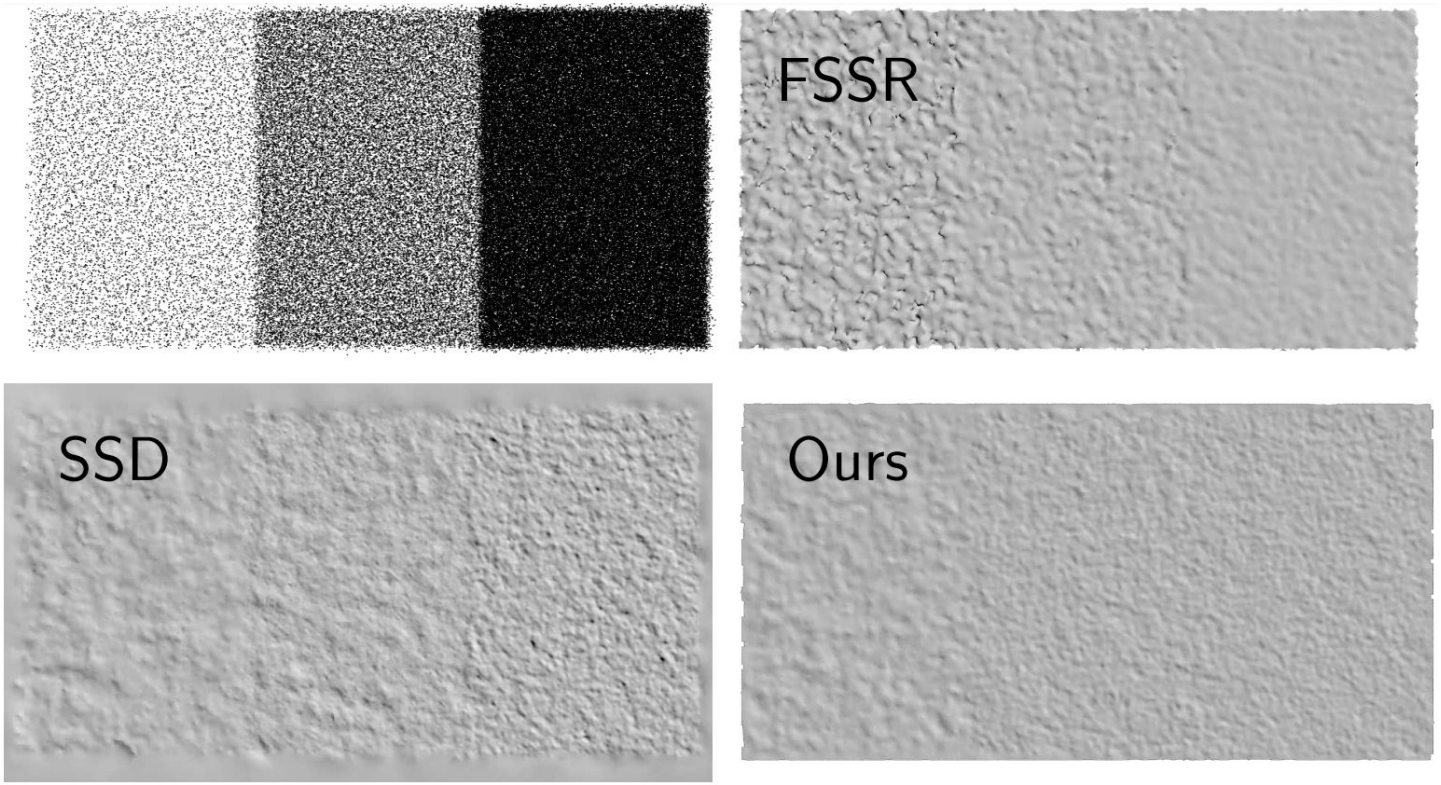


Figure 11. Reconstruction of a plane with three regions, each with a different uniform point density. The scale value assigned to the points corresponds to the sampling density of the central region. Gaussian white noise was added to the points' position and normal. **Top left:** Input point cloud. **FSSR:** With increasing density, FSSR effectively cancels out noise. In the low density region it suffers from a too sparse sampling. **SSD:** SSD adapts the scale to the point density and therefore models the noise in the high density region. In the low density region noise is suppressed by using a coarser scale for reconstruction. **Ours:** Our reconstruction looks more even in the high density region than that of the other methods. The high density leads to a strong data term but is also effective to cancel out noise. In the low density region the smoothness term dominates and the reconstruction looks smoother than with SSD.

Global, Dense Multiscale Reconstruction for a Billion Points, Ummenhofer et. al., 2017

3D Model reconstruction (для миллиарда точек)

1) **Сбалансированное** адаптивное октодерево

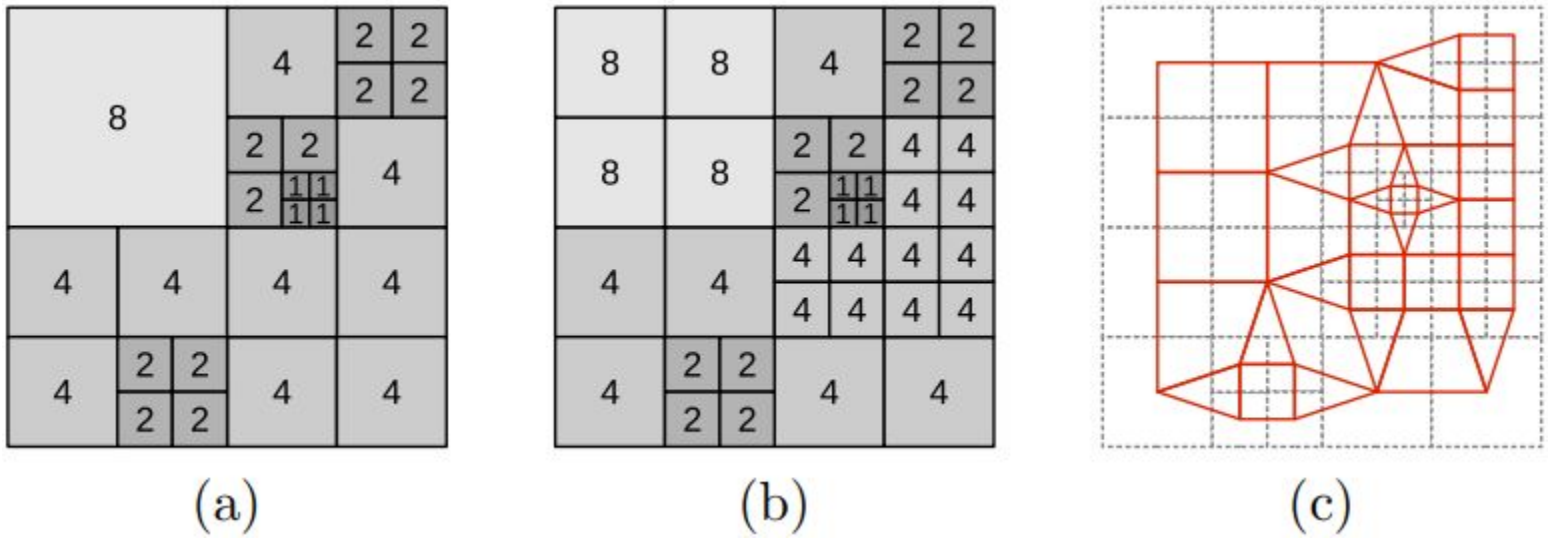


Figure 3. **(a)** Quadtree before balancing. Each node stores the scale. **(b)** Quadtree after balancing. The difference of the quadtree level of adjacent nodes is limited to 1 by recursive splitting of nodes. **(c)** Balanced quadtree (dashed) and its corresponding dual structure (solid red).

Global, Dense Multiscale Reconstruction for a Billion Points, Ummenhofer et al., 2017

3D Model reconstruction (для миллиарда точек)

1) Сбалансированное адаптивное октодерево
2) Гистаграммы нормалей:

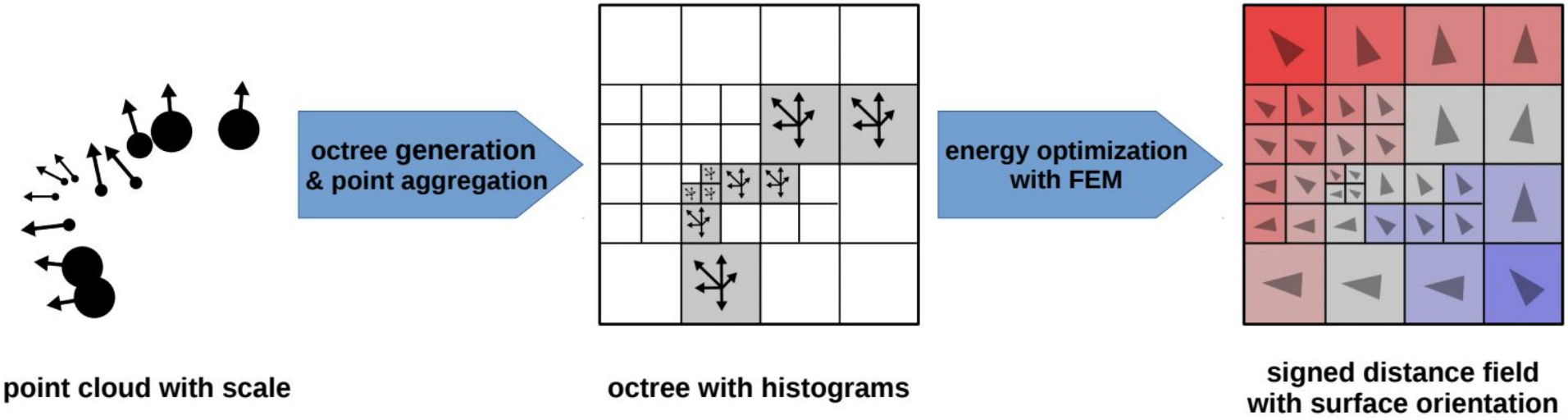


Figure 2. Method overview. The input data is a point cloud with scale and normal information. First, we generate an octree based on the scale information in the point cloud and aggregate the information into orientation and distance histograms (with compact support). Second, the implicit representation is computed, which is a signed distance function and a vector field describing the surface orientation.

Global, Dense Multiscale Reconstruction for a Billion Points, Ummenhofer et al., 2017

3D Model reconstruction (для миллиарда точек)

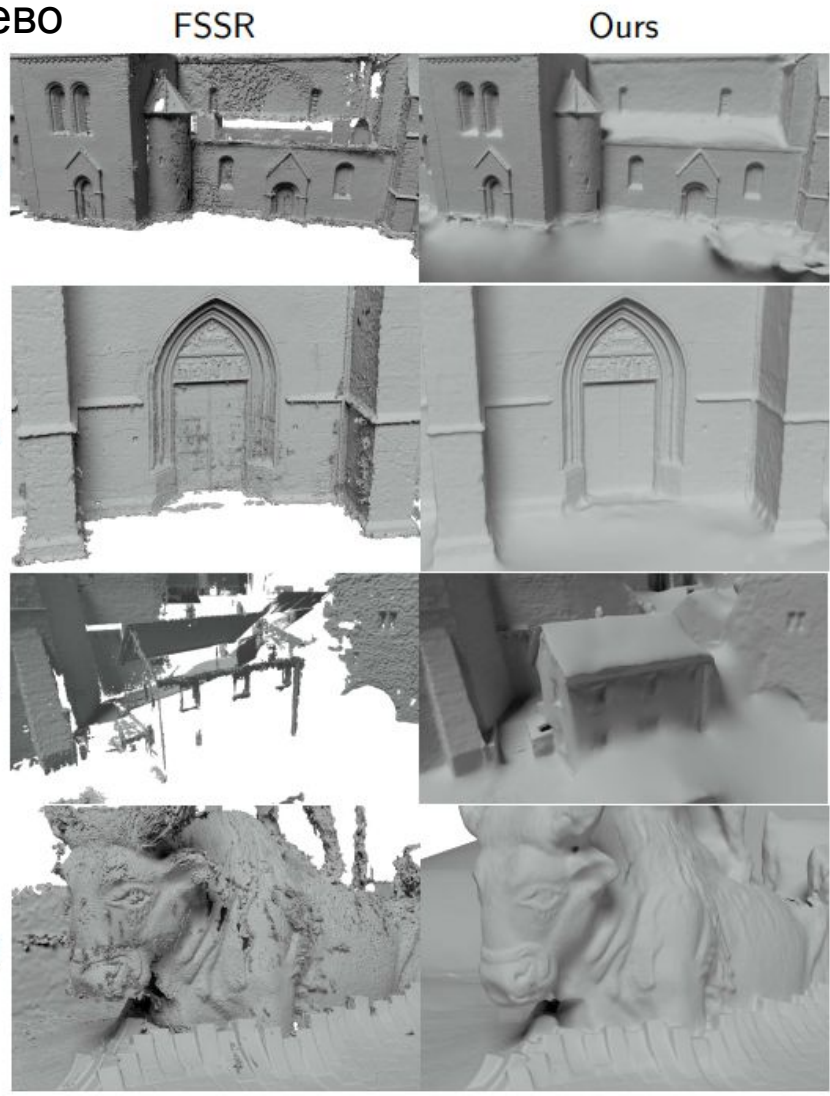
1) Сбалансированное адаптивное октодерево
2) Гистаграммы нормалей
3) Breisach dataset (1.5 миллиарда точек):

		Duration
Octree generation	Density estimation	74.6 min
	Balancing	7.9 min
	Histograms	782.4 min
Surface comp.	Dual grid generation	19.9 min
	Energy minimization	3678.0 min
	Dual contouring	16.3 min
Other		23.5 min
Total		4602.9 min

17%

80%

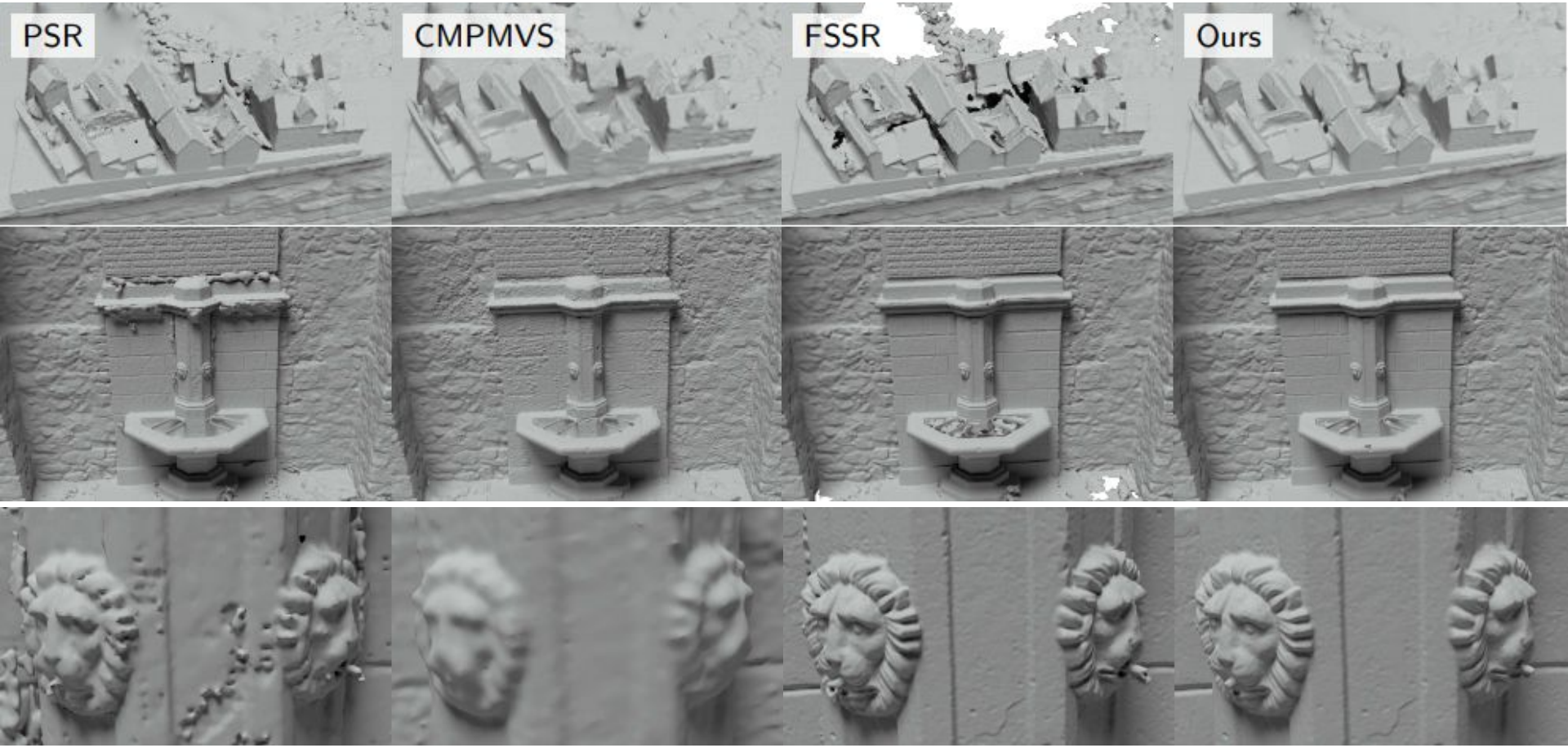
Table 1. Runtime breakdown for the Breisach data set.



Global, Dense Multiscale Reconstruction for a Billion

3D Model reconstruction (для миллиарда точек)

1) Сбалансированное адаптивное октодерево
2) Гистаграммы нормалей
3) Breisach dataset (1.5 миллиарда точек)
4) Citywall dataset (256 миллионов точек):



Global, Dense Multiscale Reconstruction for a Billion Points, Ummenhofer et al., 2017

3D Model reconstruction (для любого числа точек)

1) Как построить адаптивное октодерево **out-of-core**?

Out-of-Core Surface Reconstruction via Global TGV Minimization, Poliarnyi, 2021

3D Model reconstruction (для любого числа точек)

1) Как построить адаптивное октодерево **out-of-core**?
   Если **ввести порядок** на кубиках (каждый кубик представим числом), то останется лишь научиться делать **out-of-core merge sort**. Morton Code!

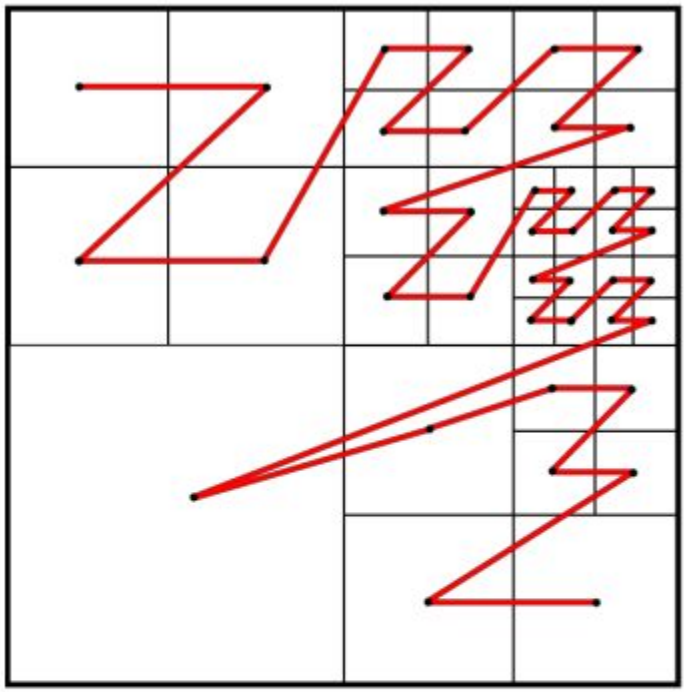
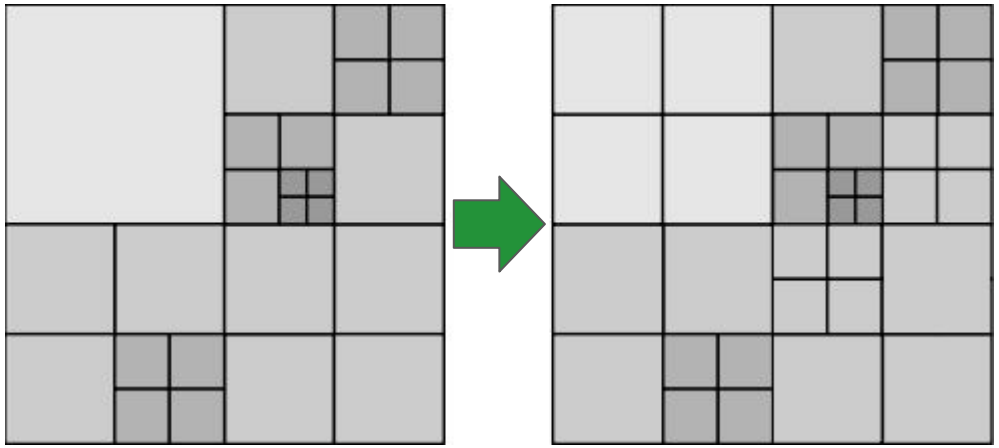


Figure 7. Z-curve example on a balanced 2D quadtree. Note that each cube has a limited number of neighbors (two or less) over each edge. In a balanced 3D octree, each cube has four or fewer neighbors over each cube's face.

Out-of-Core Surface Reconstruction via Global TGV Minimization, Poliarnyi, 2021

3D Model reconstruction (для любого числа точек)

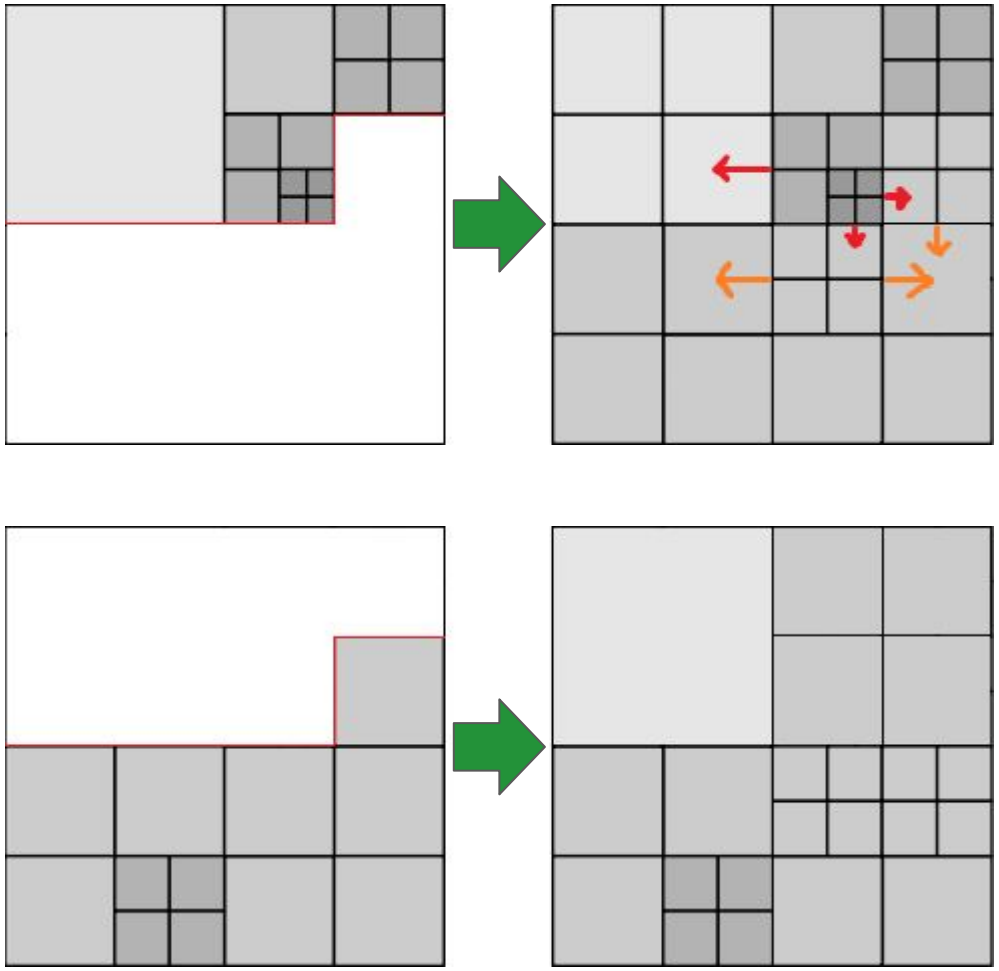
1) Строим адаптивное октодерево.
2) Как сбалансировать построенное дерево **out-of-core**?



Out-of-Core Surface Reconstruction via Global TGV Minimization, Poliarnyi, 2021

3D Model reconstruction (для любого числа точек)

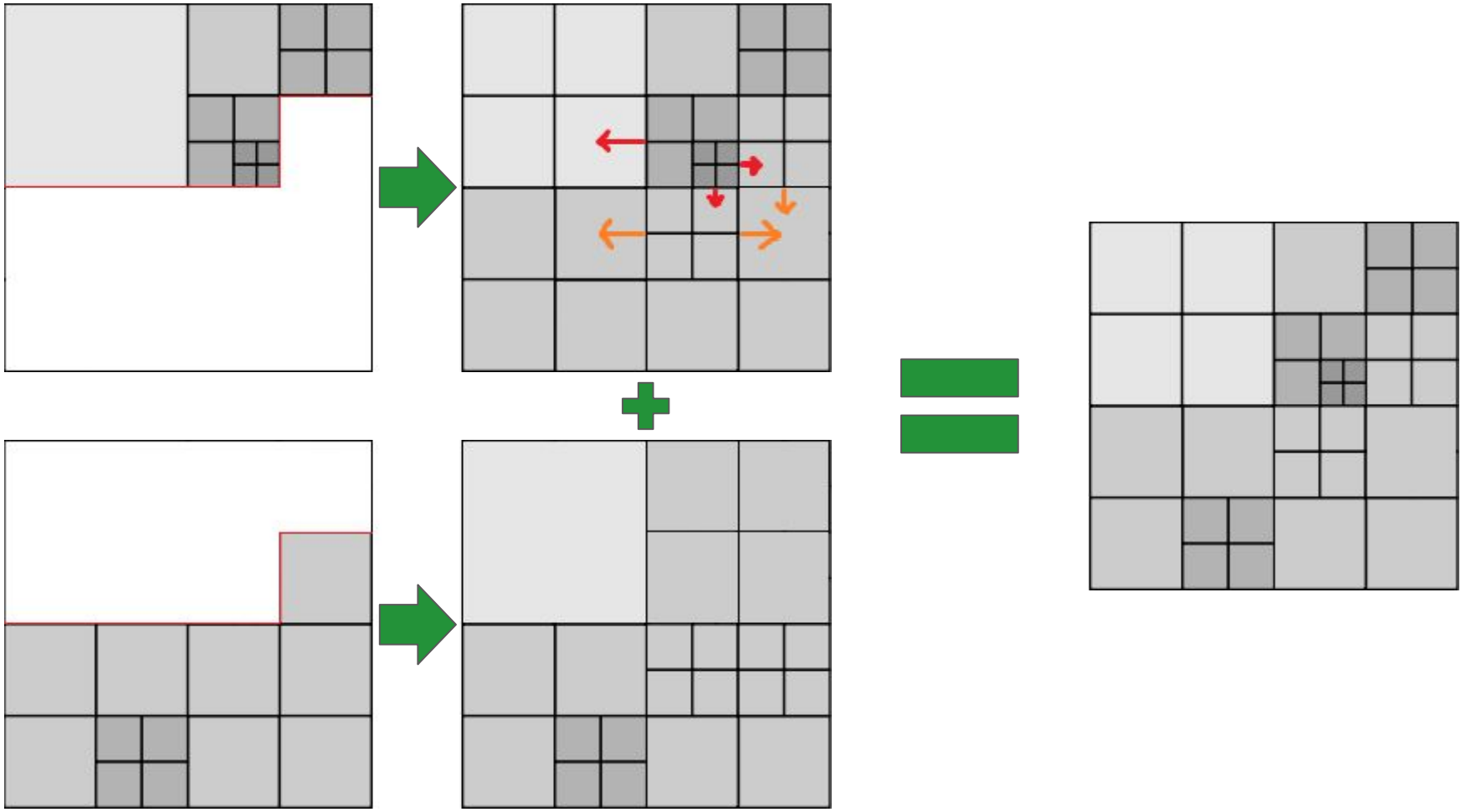
1) Строим адаптивное октодерево.
2) Как сбалансировать построенное дерево **out-of-core**?



Out-of-Core Surface Reconstruction via Global TGV Minimization, Poliarnyi, 2021

3D Model reconstruction (для любого числа точек)

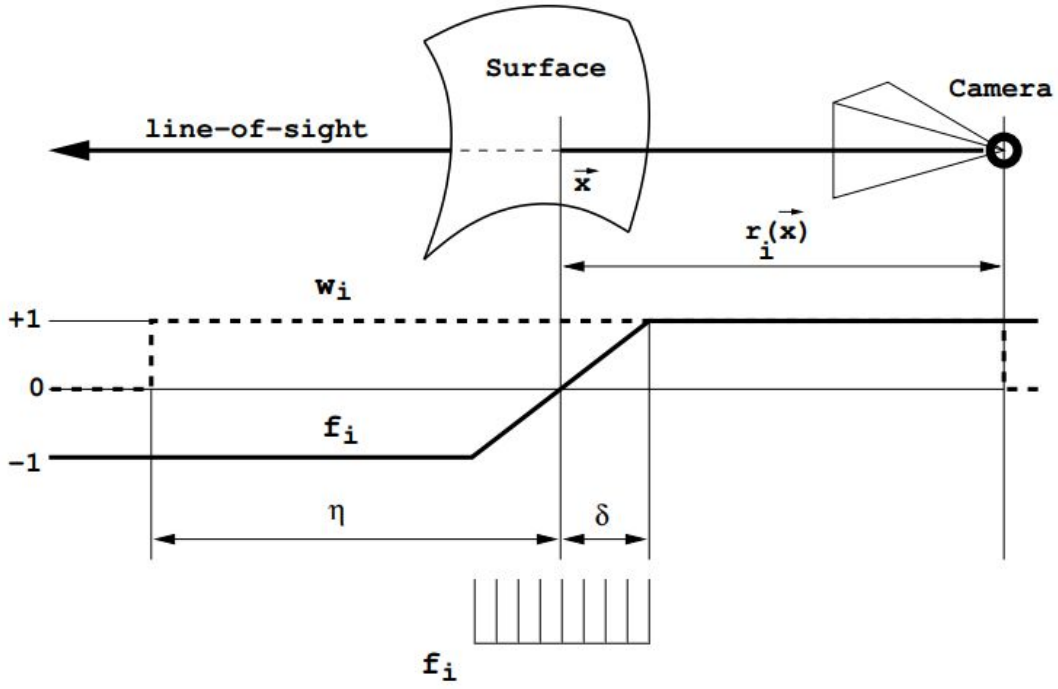
1) Строим адаптивное октодерево.
2) Как сбалансировать построенное дерево **out-of-core**?



Out-of-Core Surface Reconstruction via Global TGV Minimization, Poliarnyi, 2021

3D Model reconstruction (для любого числа точек)

1) Строим адаптивное октодерево.
2) Балансируем октодерево.
3) Как построить SDF гистограммы для сбалансированного дерева с учетом всех карт глубины?



Out-of-Core Surface Reconstruction via Global TGV Minimization, Poliarnyi, 2021

3D Model reconstruction (для любого числа точек)

1) Строим адаптивное октодерево.
2) Балансируем октодерево.
3) Как построить SDF гистограммы для сбалансированного дерева с учетом всех карт глубины?
4) Как делать primal-dual итерации?

Out-of-Core Surface Reconstruction via Global TGV Minimization, Poliarnyi, 2021

3D Model reconstruction (для любого числа точек)

1) Строим адаптивное октодерево.
2) Балансируем октодерево.
3) Как построить SDF гистограммы для сбалансированного дерева с учетом всех карт глубины?
4) Как делать primal-dual итерации?
5) Как выделять поверхность алгоритмом marching cubes?

Out-of-Core Surface Reconstruction via Global TGV Minimization, Poliarnyi, 2021

3D Model reconstruction (для любого числа точек)

1) Строим адаптивное октодерево.
2) Балансируем октодерево.
3) **Строим индексированную верхушку дерева!**

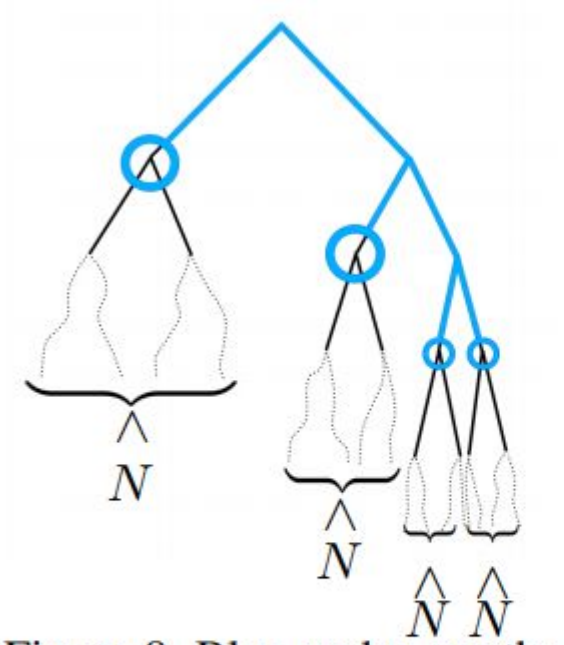


Figure 8. Blue nodes are the treetop leaves with less than $N_{cubesPerTreetopLeaf}$ cubes under each subtree.

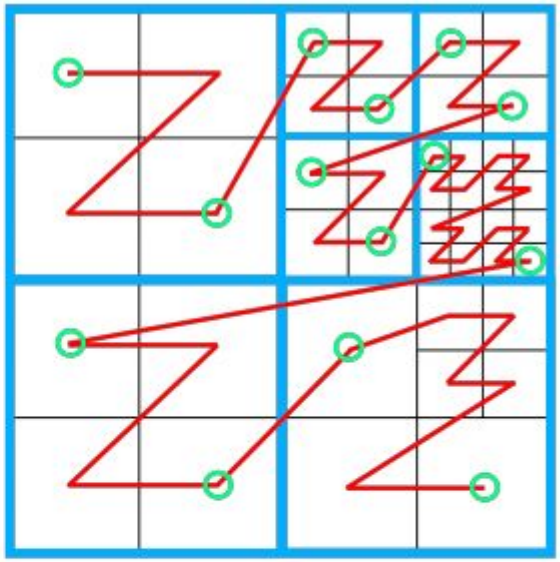


Figure 9. Note that the Z-curve enters and leaves cubes (green circles) from each treetop leaf (blue boxes) exactly once.

We build independent treetops for all linear balanced octree parts, and then merge those treetops into a global one. At this stage, we can easily save indices of all covered relevant cubes for each treetop leaf. Moreover, these indices are consecutive due to Z-curve ordering of Morton codes – see Fig. 9. Hence, we only need to save two indices with each treetop leaf –indices of the first and the last relevant cubes from the linear balanced octree. This gives us an ability to load cubes relevant for current treetop leaf from balanced octree in IO-friendly consecutive way.

Out-of-Core Surface Reconstruction via Global TGV Minimization, Poliarnyi, 2021

3D Model reconstruction (для любого числа точек)

1) Строим адаптивное октодерево.
2) Балансируем октодерево.
3) Строим индексированную верхушку дерева.
4) Строим SDF гистограммы для каждой части октодерева (для каждого листа верхушки). **[GPU]**

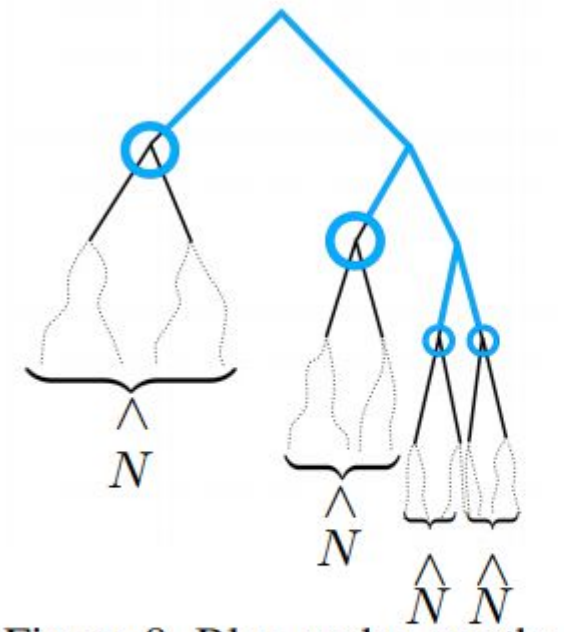


Figure 8. Blue nodes are the treetop leaves with less than $N_{cubesPerTreetopLeaf}$ cubes under each subtree.

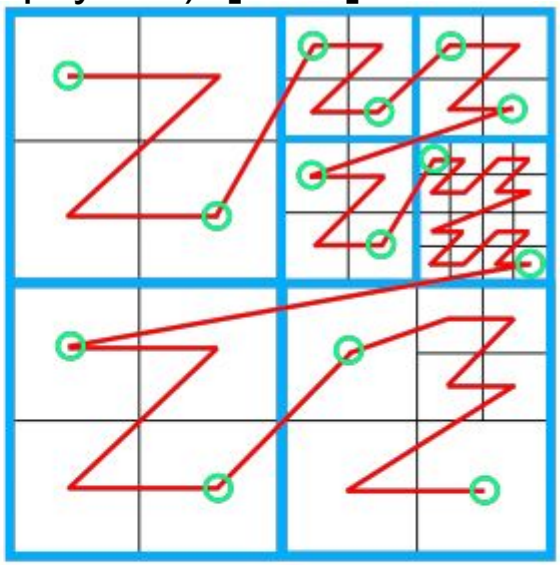
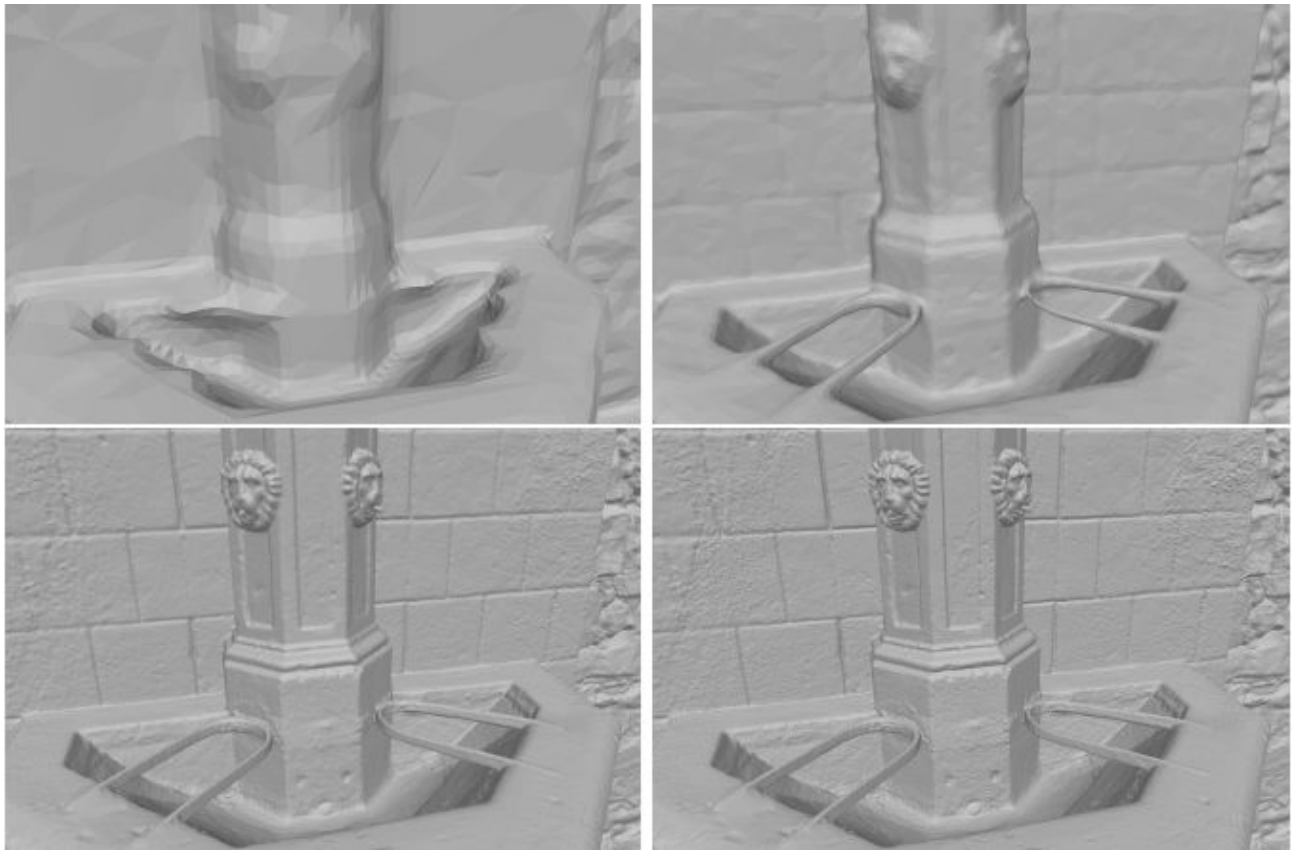


Figure 9. Note that the Z-curve enters and leaves cubes (green circles) from each tree-top leaf (blue boxes) exactly once.

Out-of-Core Surface Reconstruction via Global TGV Minimization, Poliarnyi, 2021

3D Model reconstruction (для любого числа точек)

1) Строим адаптивное октодерево.
2) Балансируем октодерево.
3) Строим индексированную верхушку дерева.
4) Строим SDF гистограммы для каждой части октодерева (для каждого листа верхушки). **[GPU]**
5) Прогрессивно по уровням (coarse-to-fine) выполняем primal-dual итерации (опять по частям, по листьям верхушки). **[GPU]**



3D Model reconstruction (для любого числа точек)

1) Строим адаптивное октодерево.
2) Балансируем октодерево.
3) Строим индексированную верхушку дерева.
4) Строим SDF гистограммы для каждой части октодерева (для каждого листа верхушки). **[GPU]**
5) Прогрессивно по уровням (coarse-to-fine) выполняем primal-dual итерации (опять по частям, по листьям верхушки). **[GPU]**
6) Извлекаем результат - поверхность из треугольников алгоритмом marching cubes (опять по частям, по листьям верхушки).

Out-of-Core Surface Reconstruction via Global TGV Minimization, Poliarnyi, 2021

3D Model reconstruction (для любого числа точек)

1) Строим адаптивное октодерево.
2) Балансируем октодерево.
3) Строим индексированную верхушку дерева.
4) Строим SDF гистограммы для каждой части октодерева (для каждого листа верхушки). **[GPU]**
5) Прогрессивно по уровням (coarse-to-fine) выполняем primal-dual итерации (опять по частям, по листьям верхушки). **[GPU]**
6) Извлекаем результат - поверхность из треугольников алгоритмом marching cubes (опять по частям, по листьям верхушки).
7) В самом конце подгружаем все части поверхности и склеиваем их в одну связную геометрию-результат.

Table 1. Breakdown of Breisach dataset processing: 2111 photos, 2642 million cubes from input depth maps, 4 hours 20 minutes of processing on a computer with an 8-core CPU and a GeForce GTX 1080 GPU with the peak RAM usage 10.07 GB.

Processing stage	**Time**	**Time in %**
Linear octree + merge	30 + 11 min	11% + 4%
Balance octree + merge	7 + 11 min	3% + 4%
Index treetop	8 min	3%
Histograms (GPU)	49 min	19%
Primal-dual method (GPU)	88 min	34%
Marching cubes	59 min	22%

Out-of-Core Surface Reconstruction via Global TGV Minimization, Poliarnyi, 2021

3D Model reconstruction (для любого числа точек)

Table 2. List of presented datasets. For depthmaps estimation speedup, we downscaled original photos for some datasets before running SGM-based [16] depthmap reconstruction. For example, original 2111 photos in Breisach dataset had the resolution of 5184x3456, and we downscaled them with x2 factor, down to 2592x1728 pixels. Note that the 'Initial cubes' column can be interpreted as 'the number of non-empty depth pixels in depthmaps', because each initial cube (before merging and octree balancing) corresponds to one sample from a depthmap.

Dataset name	Images resolution after downscale (and downscale factor)	Input data	Initial cubes	Merged and balanced cubes	Faces after marching cubes	Decimated faces	Peak RAM (GB)	Processing time
Citywall [10]	2000x1500 (x1)	564 depth maps	1205 mil	404 mil	135 mil	15 mil	13.17	63 min
Breisach [27]	2592x1728 (x2)	2111 depth maps	2642 mil	1457 mil	558 mil	57 mil	10.07	260 min
Tomb of Tu Duc (LIDAR) [6]	8000x4000 (x1)	42 LIDAR scans	661 mil	1304 mil	672 mil	48 mil	10.05	160 min
Palacio Tschudi [7]	1840x1228 (37%) 1500x1000 (63%) (x4)	13703 depth maps	16 billion	6 billion	3159 mil	243 mil	16.75	1213 min
Copenha-gen city [8]	3368x2168 (26%) 2575x1925 (74%) (x4)	27472 depth maps	28 billion	24 billion	7490 mil	267 mil	13.35	1758 min

Dataset name	Input data	GDMR Peak RAM	GDMR time	**Our** Peak RAM	**Our** time	SSR Peak RAM	SSR time
Citywall	564 depth maps	75 GB	19 h	13.17 GB	63 min	32*8.9 GB	58 h
Breisach	2111 depth maps	64 GB	76 h	10.07 GB	260 min	N/A	N/A

3D Model reconstruction (для любого числа точек)

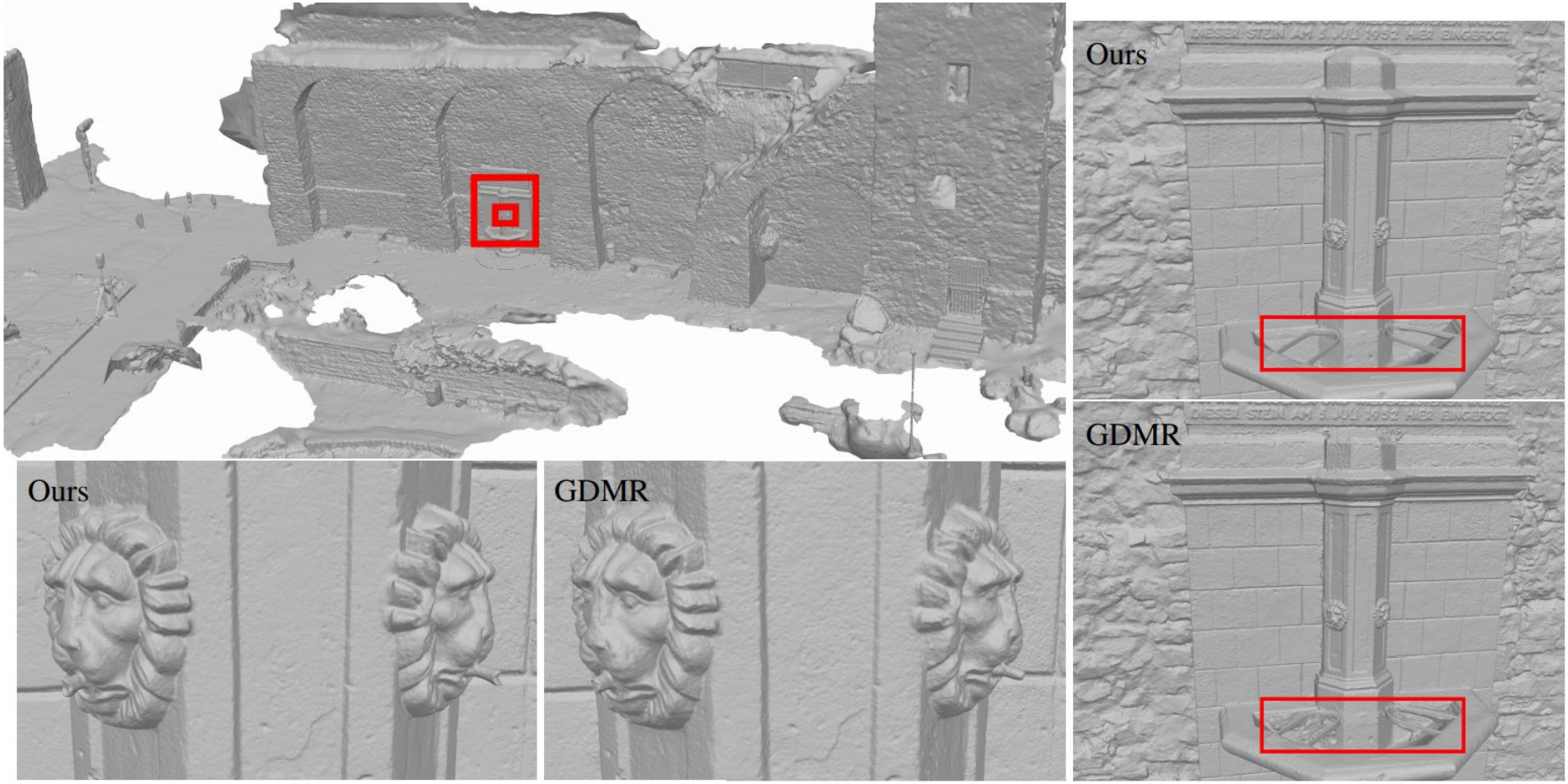


Figure 4. Results on the Citywall dataset and comparison with GDMR [27] results. The results are comparable. Note that due to strong visibility-based noise-filtering properties our method led to the cleaner basin of the fountain.

Out-of-Core Surface Reconstruction via Global TGV Minimization, Poliarnyi, 2021

3D Model reconstruction (для любого числа точек)

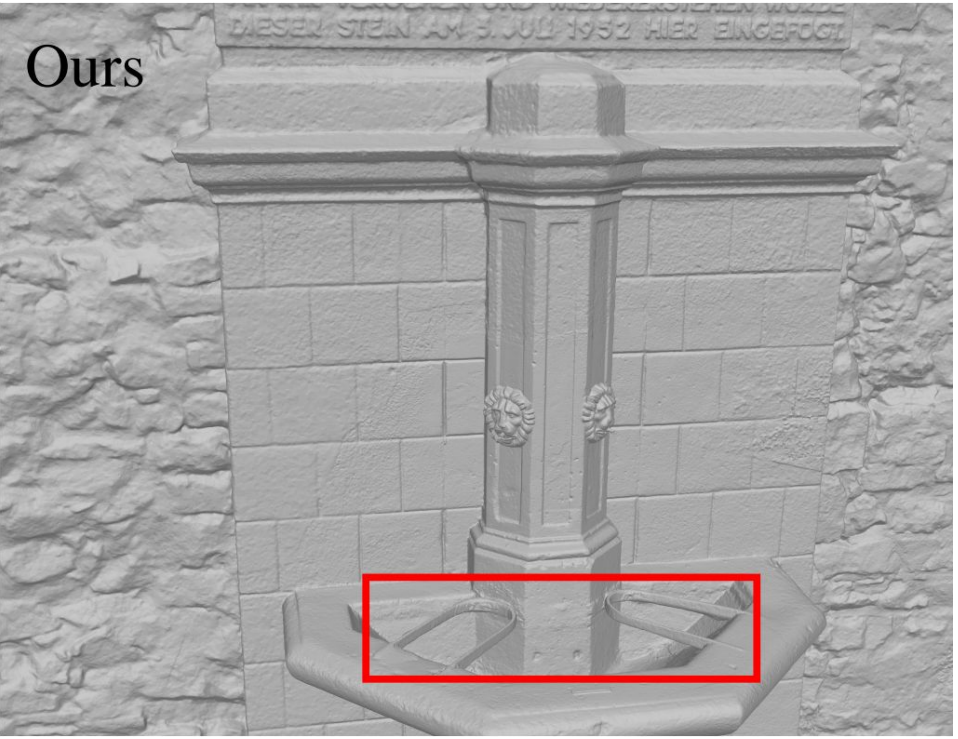


Figure 4. Results on the Citywall dataset and comparison with GDMR [27] results. The results are comparable. Note that due to strong visibility-based noise-filtering properties our method led to the cleaner basin of the fountain.

Out-of-Core Surface Reconstruction via Global TGV Minimization, Poliarnyi, 2021

3D Model reconstruction (для любого числа точек)

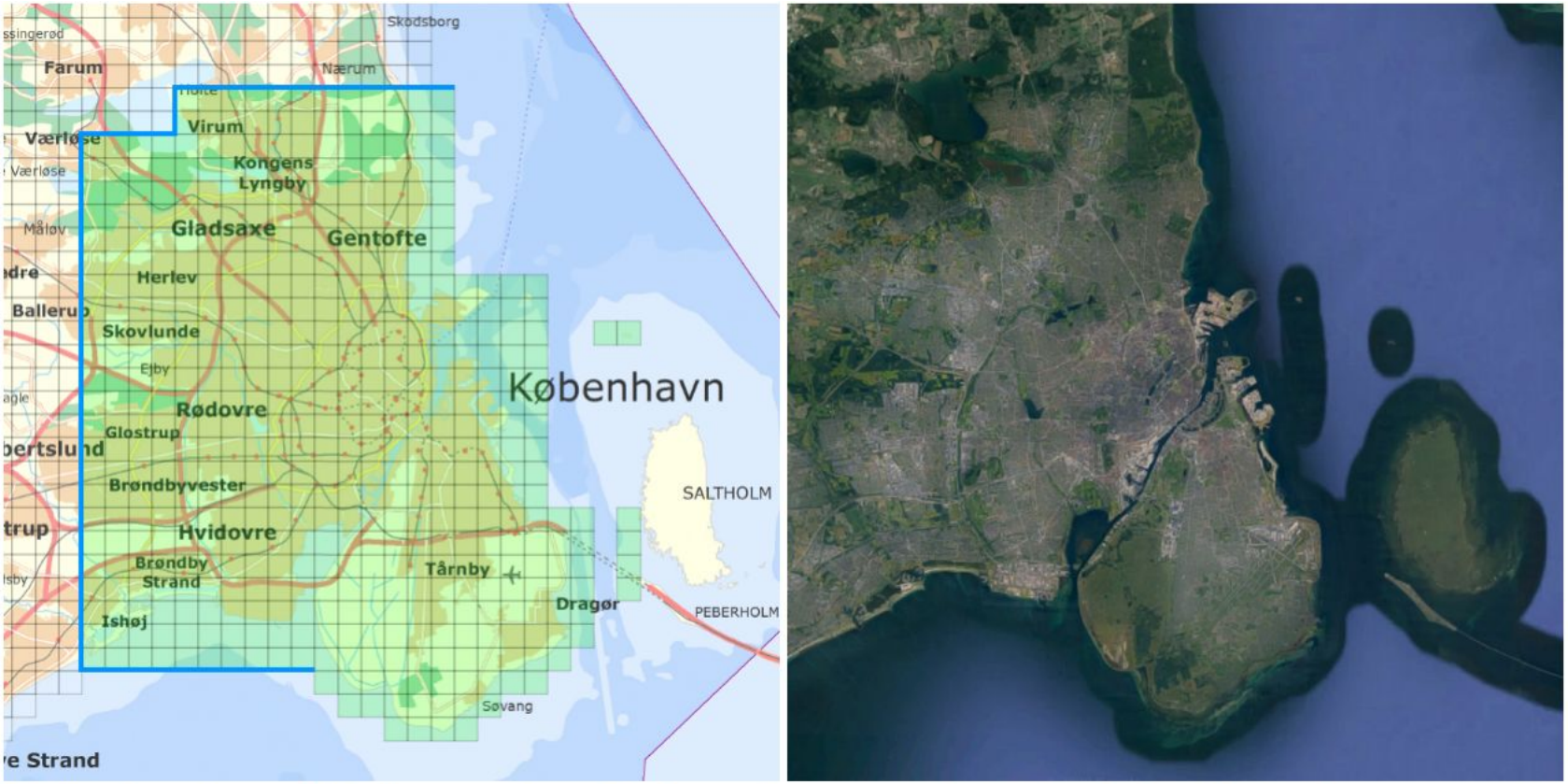


Figure 1. Copenhagen dataset [1]: 500 processed green blocks are shown (outlined with a blue border). To the right, a corresonding google map is shown [3].

Out-of-Core Surface Reconstruction via Global TGV Minimization, Poliarnyi, 2021

3D Model reconstruction (для любого числа точек)

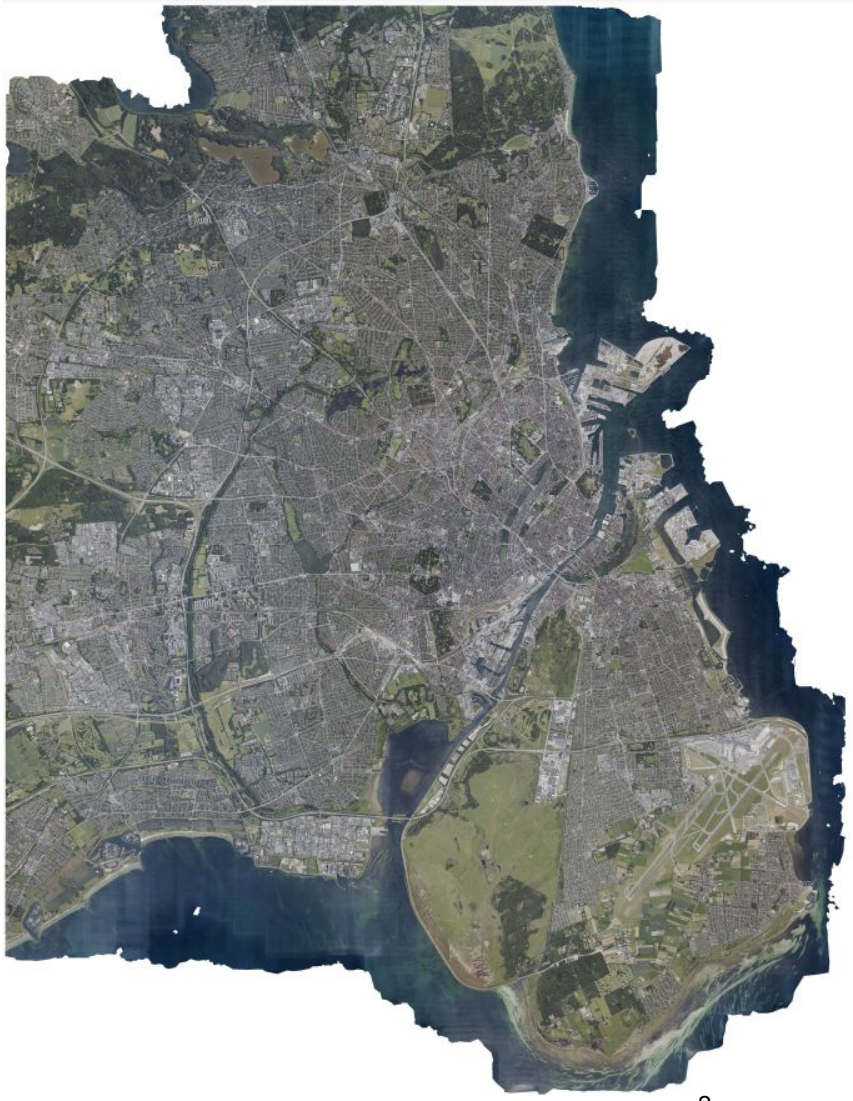


Figure 2. Our method can handle an arbitrary large scene – even 425 km^2 of the Copenhagen city. This polygonal model, consisting of 267 million triangles with per-vertex colors, was reconstructed from depth maps of 27472 aerial photos in 29 hours.

Out-of-Core Surface Reconstruction via Global TGV Minimization, Poliarnyi, 2021

3D Model reconstruction (для любого числа точек)

Figure 4. Frederiksberg Forsyning A/S closeup. Note that both pipes were reconstructed well.

Out-of-Core Surface Reconstruction via Global TGV Minimization, Poliarnyi, 2021

3D Model reconstruction (для любого числа точек)

Figure 7. Airport closeup.

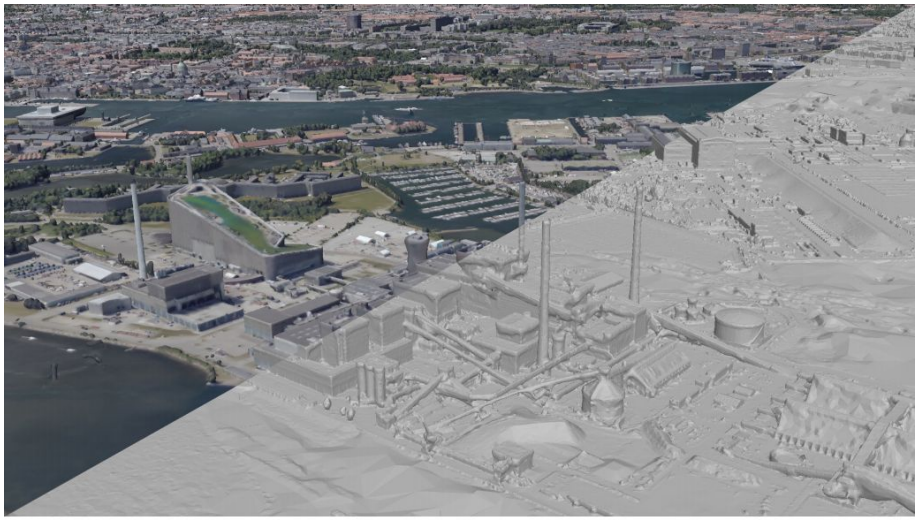


Figure 8. Hofor Amagerverket closeup.

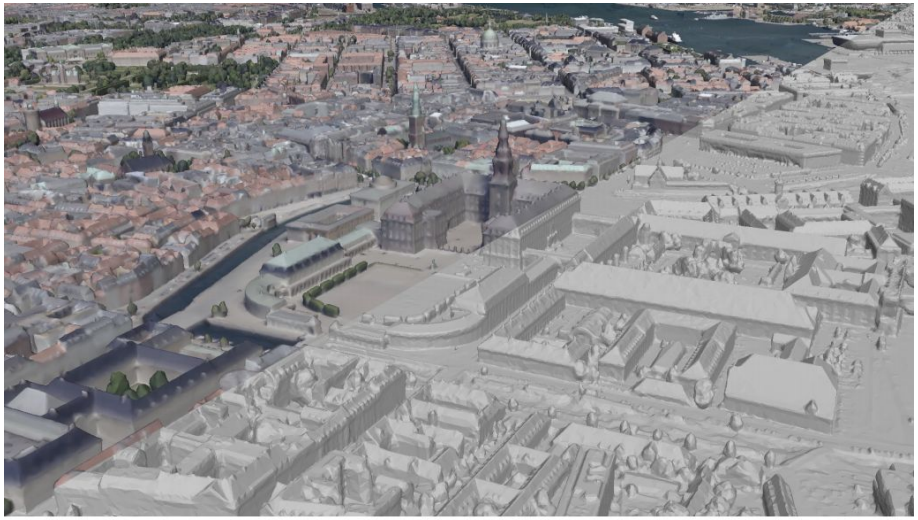


Figure 9. Christiansborg Palace closeup.

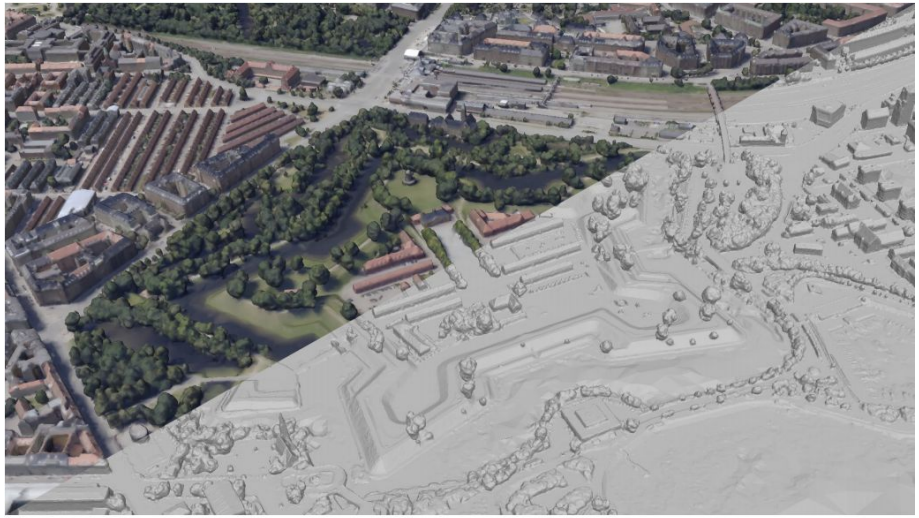


Figure 10. Kastellet closeup.

Out-of-Core Surface Reconstruction via Global TGV Minimization, Poliarnyi, 2021

Ссылки

- TGV-Fusion, Pock et al., 2011
- A Globally Optimal Algorithm for Robust TV-L1 Range Image Integration, Zach et al., 2007
- Fast and High Quality Fusion of Depth Maps, Zach, 2008
- Global, Dense Multiscale Reconstruction for a Billion Points, Ummenhofer et al., 2017
- Out-of-Core Surface Reconstruction via Global TGV Minimization, Poliarnyi, 2021

Вопросы?

Полярный Николай
polarnick239@gmail.com