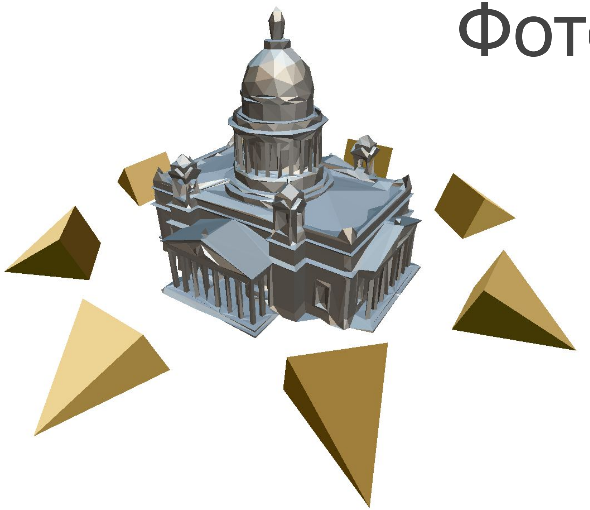
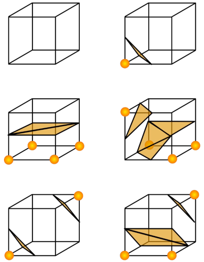
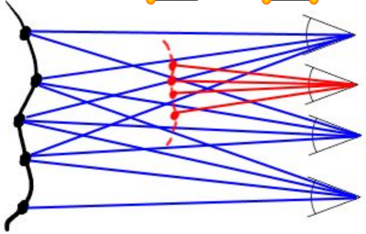
Введение в фотограмметрию
3D модель из карт глубины (TV-L1)

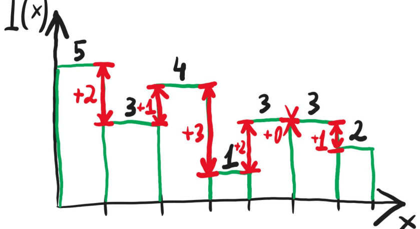
Фотограмметрия. Лекция 19

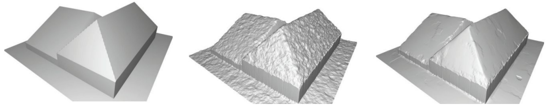


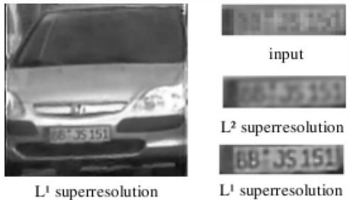
- **Удаление шума из картинок (ROF)**
- **Primal-Dual оптимизация TV-L2, TV-L1**
- **Super-resolution для видео**
- **2.5D DEM / DSM из карт глубины**
- **Маршировка кубов**
- **3D модель из карт глубины**

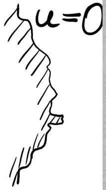






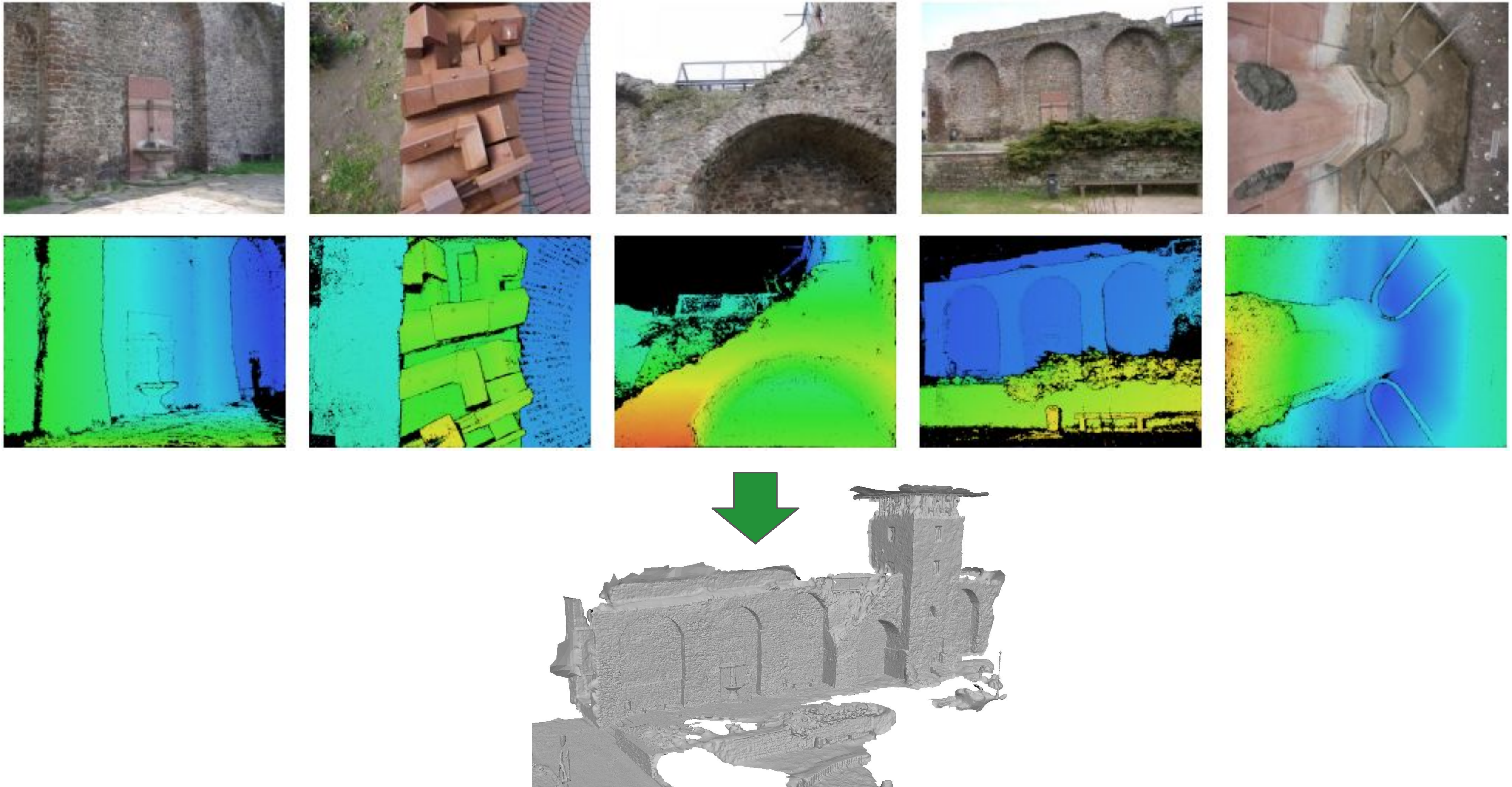






Полярный Николай
polarnick239@gmail.com

Входные данные: карты глубины

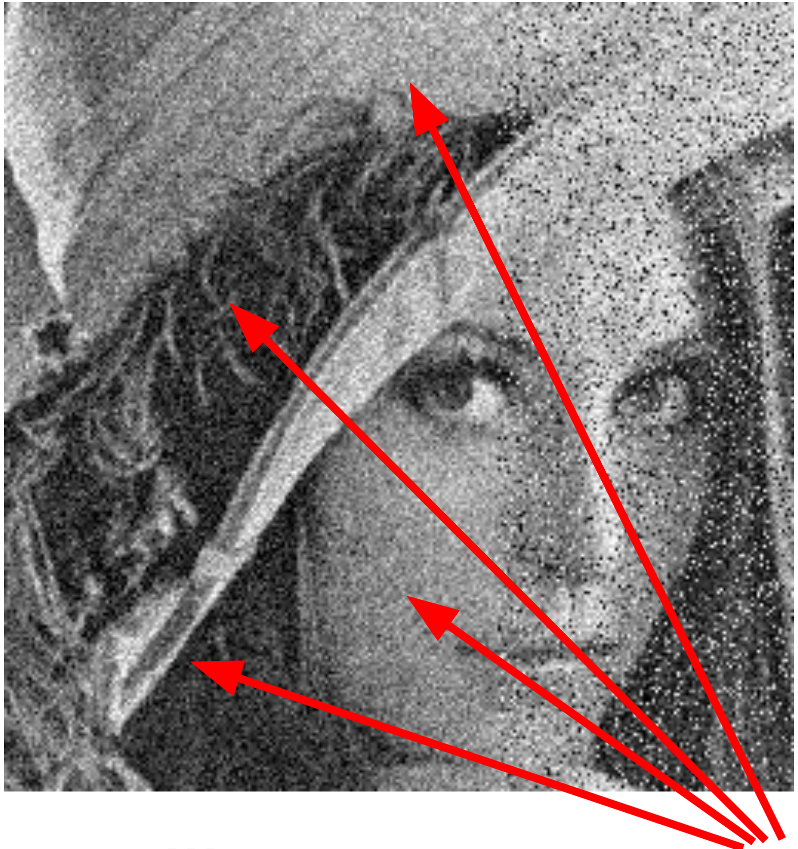


Удаление шума с картинки

Шумная картинка

Эталон

Удаление шума с картинки

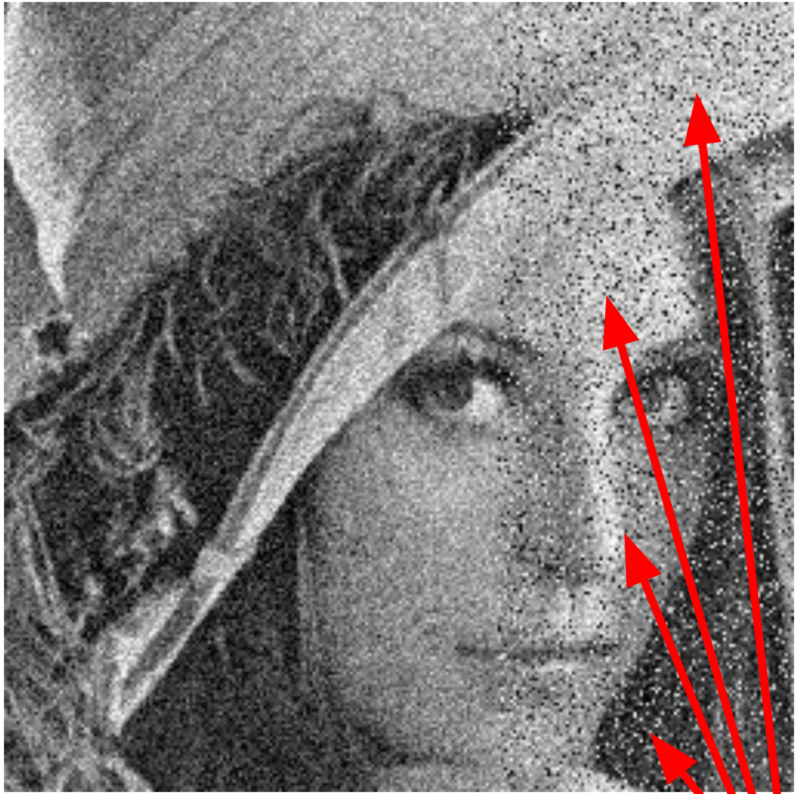


Шумная картинка

Гауссов шум

Эталон

Удаление шума с картинки

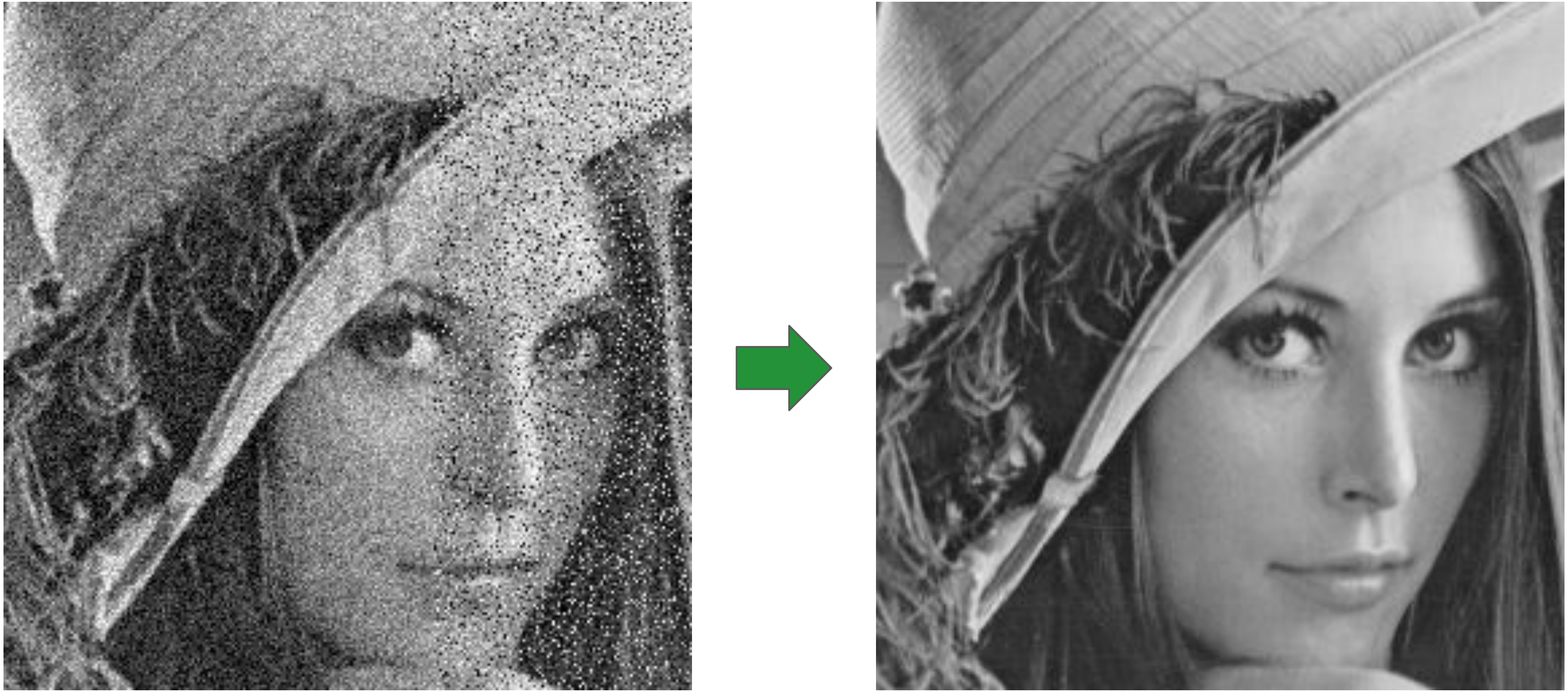


Шумная картинка

Соль & перец
(выбросы)

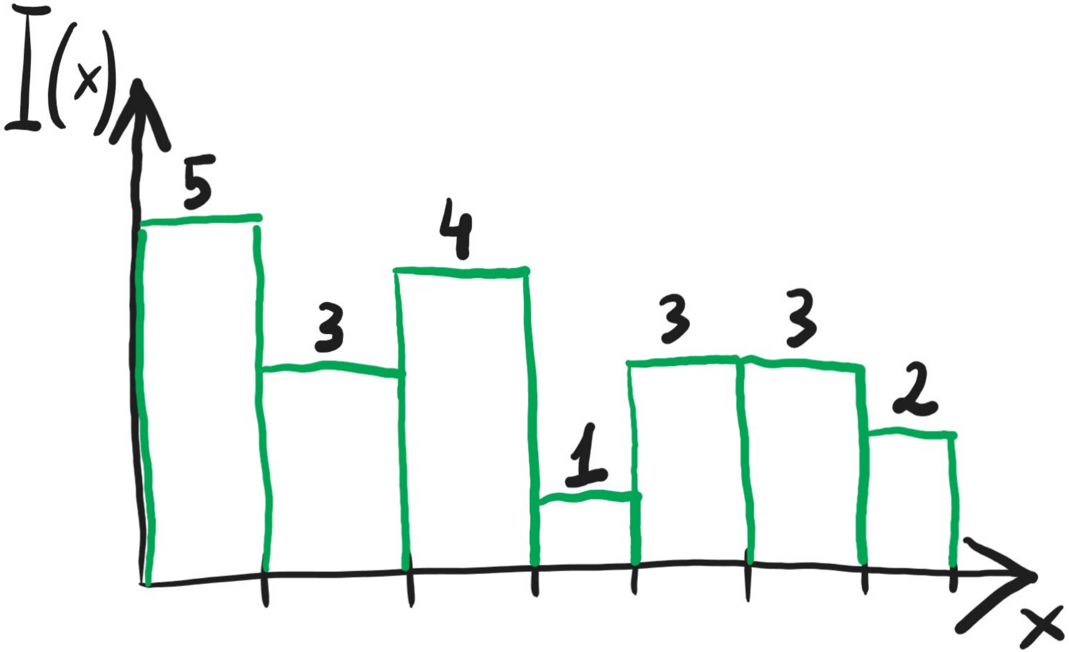
Эталон

Удаление шума с картинки

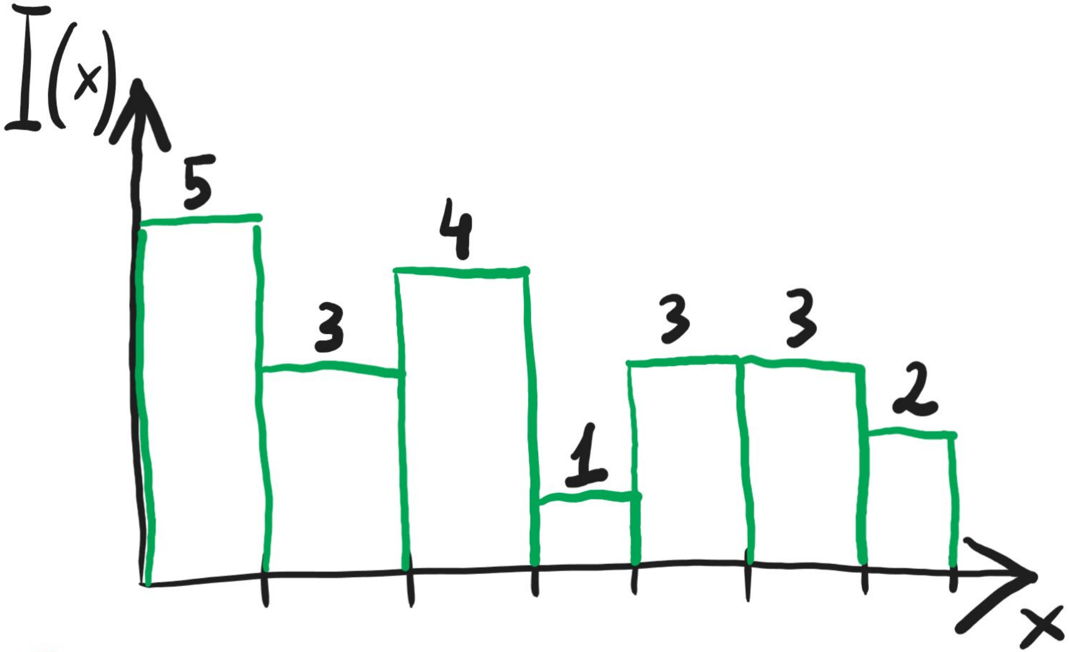


За счет чего можно удалить шум и выбросы?

Полная вариация, Total Variation (**TV**)

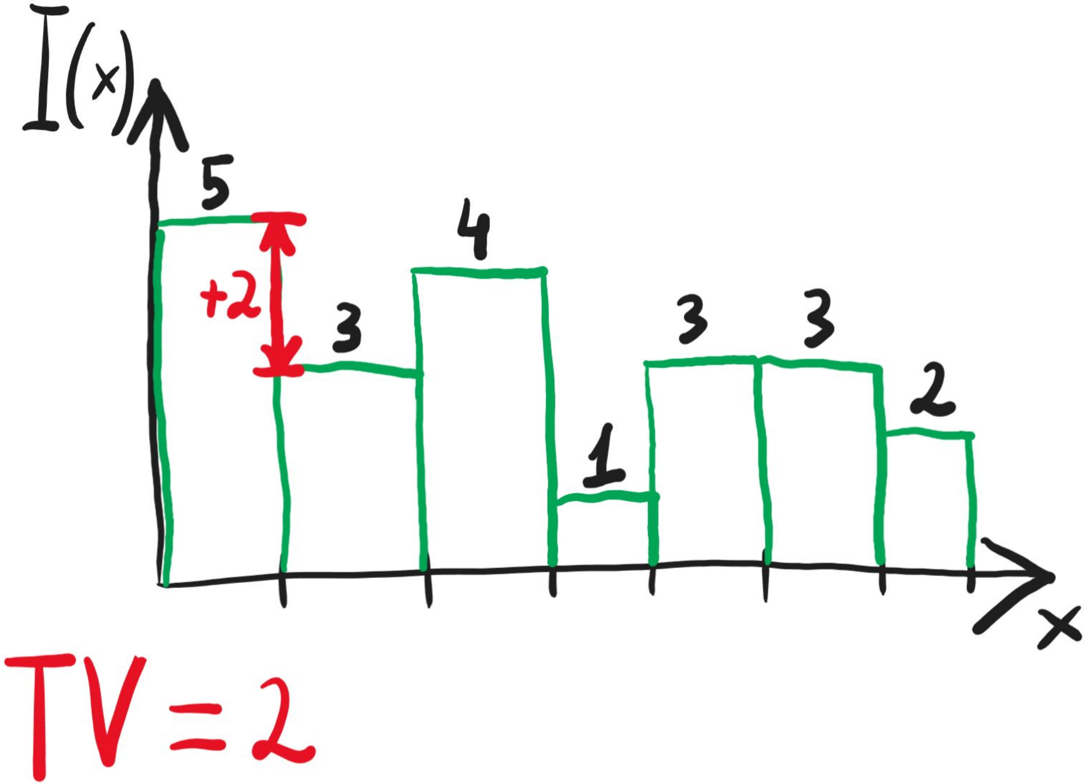


Полная вариация, Total Variation (**TV**)

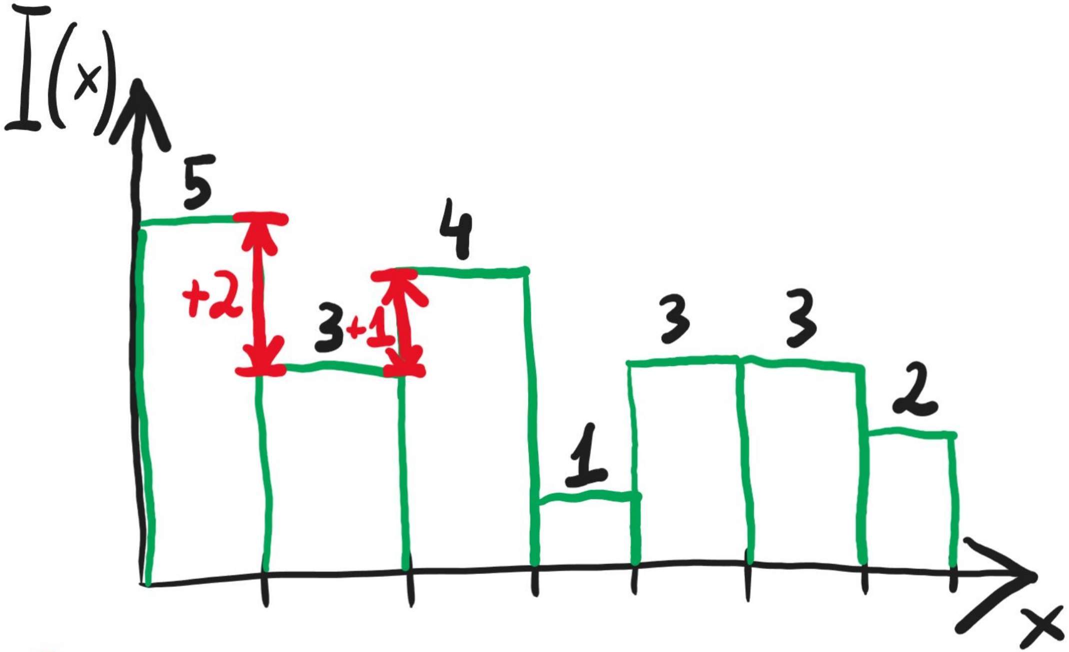


TV =

Полная вариация, Total Variation (**TV**)

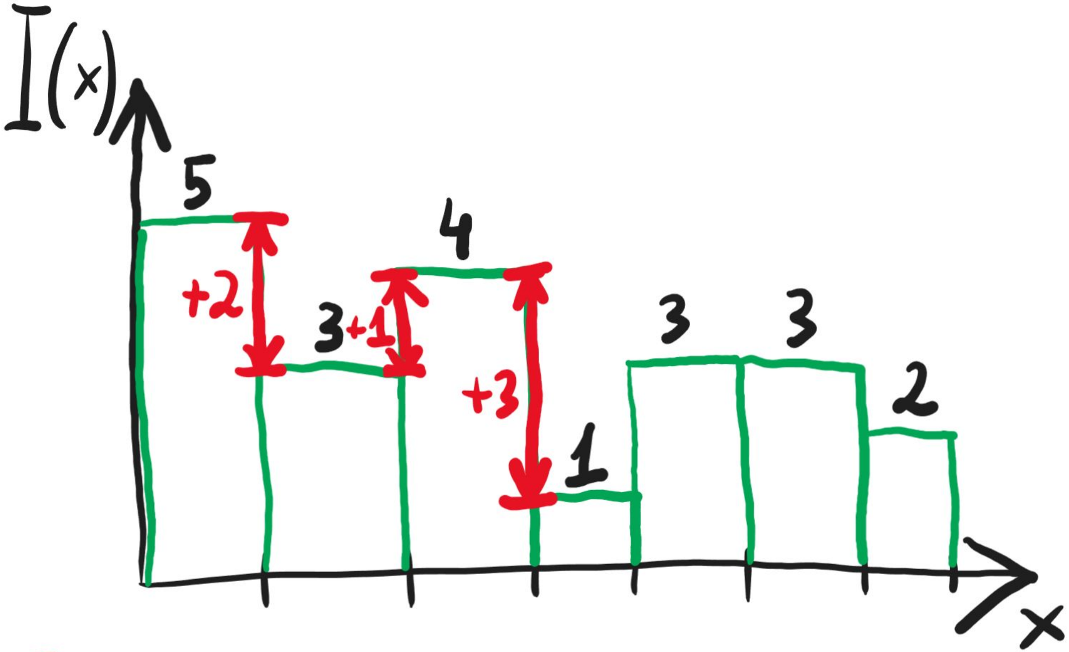


Полная вариация, Total Variation (**TV**)



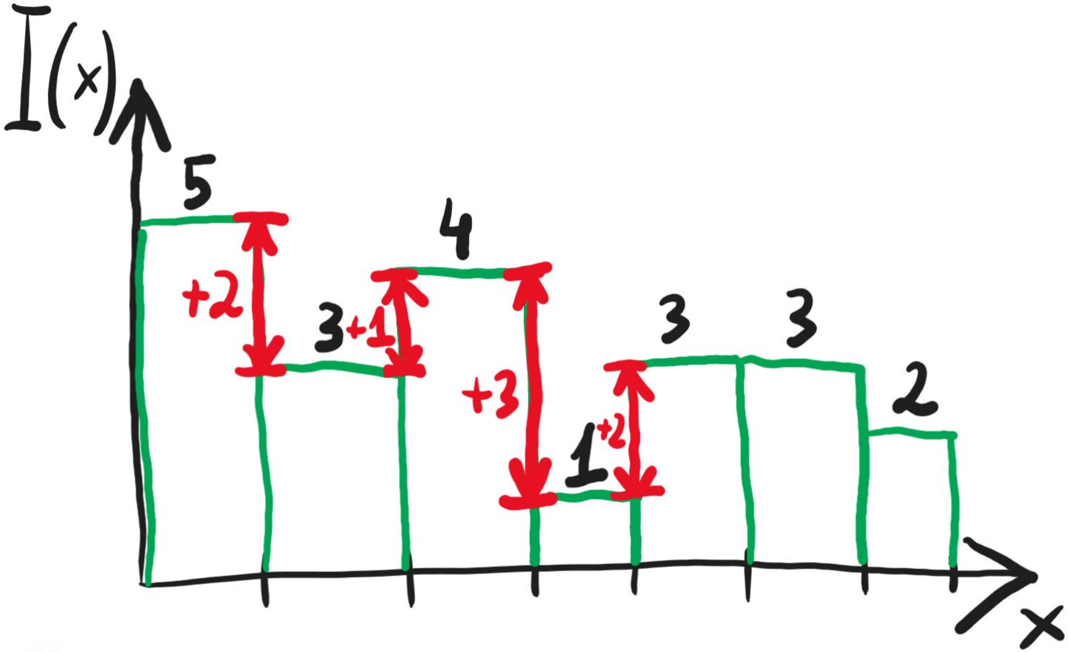
TV = 2 + 1

Полная вариация, Total Variation (**TV**)



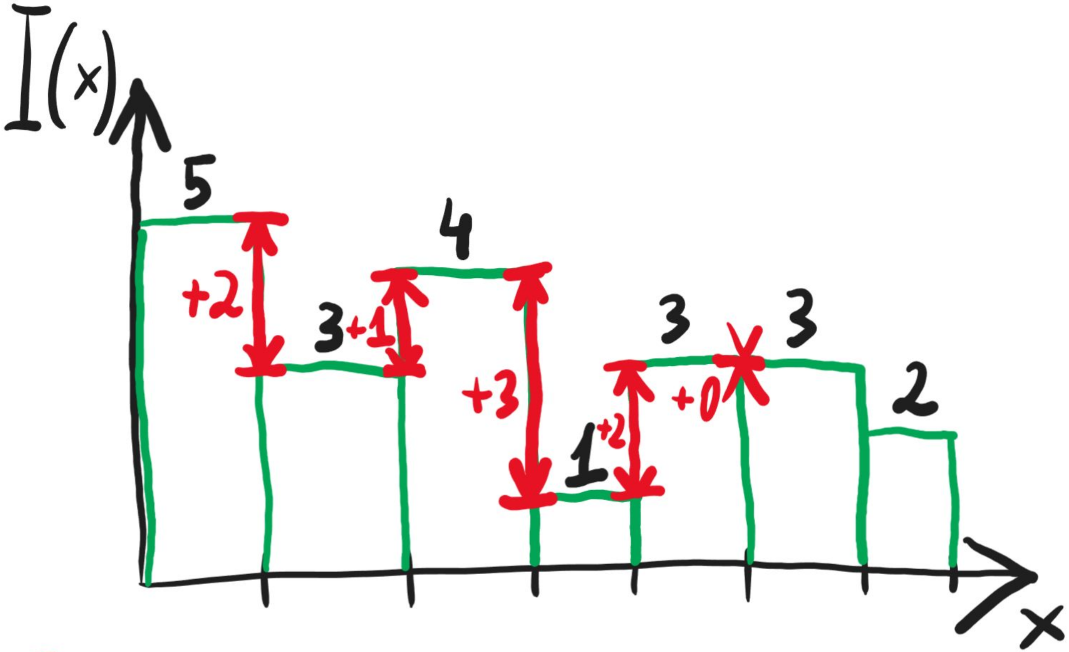
$TV = 2 + 1 + 3$

Полная вариация, Total Variation (**TV**)



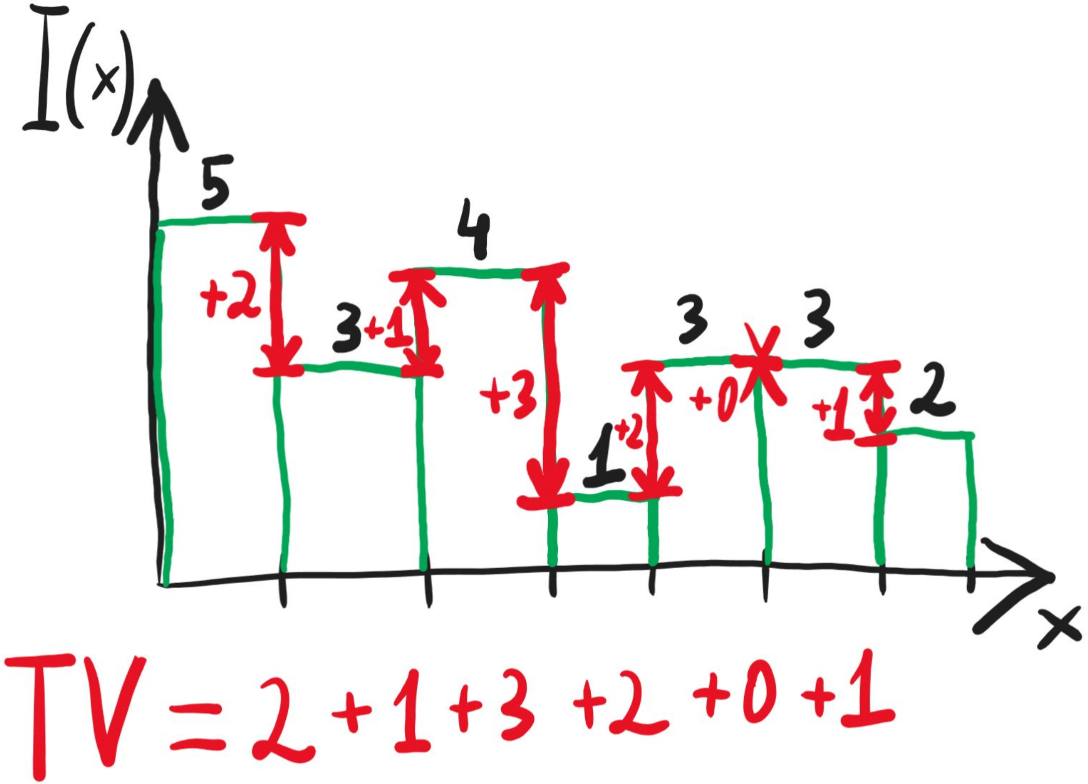
$$TV = 2 + 1 + 3 + 2$$

Полная вариация, Total Variation (**TV**)

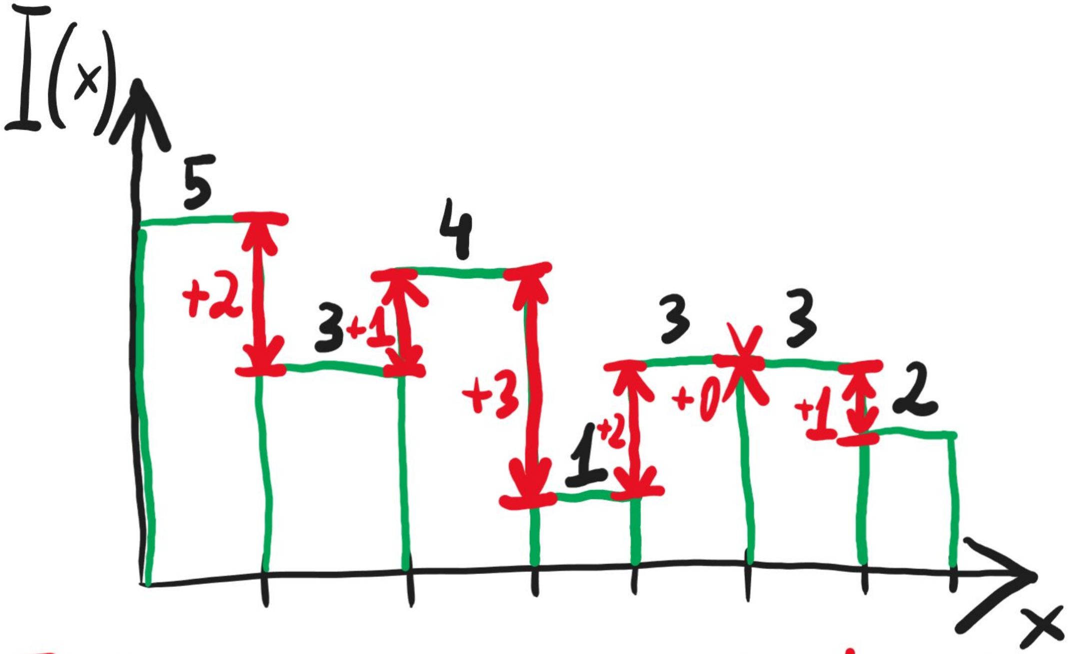


$TV = 2 + 1 + 3 + 2 + 0$

Полная вариация, Total Variation (**TV**)



Полная вариация, Total Variation (**TV**)



$$TV = 2 + 1 + 3 + 2 + 0 + 1 = 9$$

TV-L2 Image Denoising (**ROF**)

Rudin-Osher-Fatemi model (ROF):

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

f - данные

x - искомое

Где:

- $\| x - f \|^2$ - L^2 тяготение к данным (наблюдаемому f)

TV-L2 Image Denoising (**ROF**)

Rudin-Osher-Fatemi model (ROF):

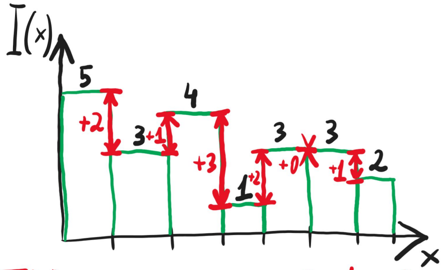
$$\min_x \boxed{\| \nabla x \|} + \frac{\lambda}{2} \boxed{\| x - f \|^2}$$

f - данные

x - искомое

Где:

- $\| x - f \|^2$ - L^2 тяготение к данным (наблюдаемому f)
- $\| \nabla x \|$ - полная вариация (регуляризация)



$TV = 2 + 1 + 3 + 2 + 0 + 1 = 9$

TV-L2 Image Denoising (**ROF**)

Rudin-Osher-Fatemi model (ROF):

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

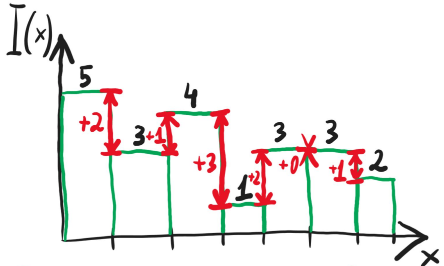
Где:

- $\| x - f \|^2$ - L^2 тяготение к данным (наблюдаемому f)
- $\| \nabla x \|$ - полная вариация (регуляризация)

f - данные

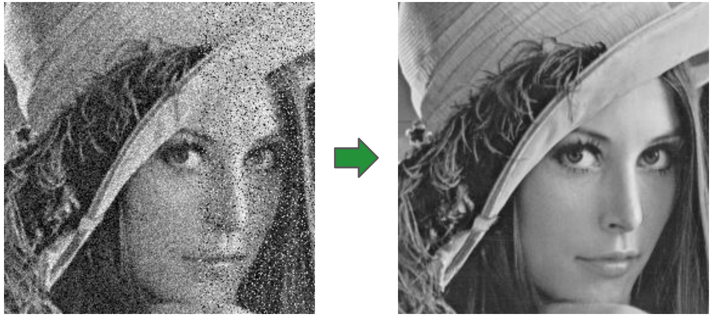
x - искомое

Что будет если убрать второе слагаемое?



$TV = 2+1+3+2+0+1=9$

TV-L2 Image Denoising (**ROF**)



f - данные $\qquad$ x - искомое

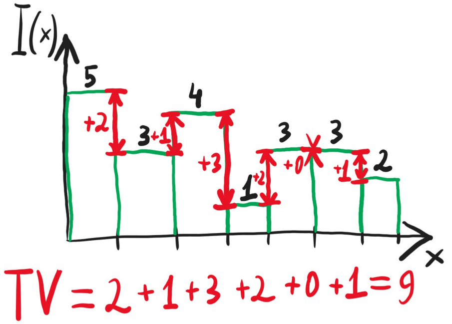
Rudin-Osher-Fatemi model (ROF):

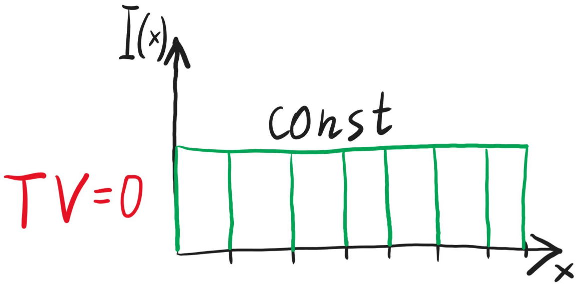
$$\min_x \boxed{\| \nabla x \|} + \frac{\lambda}{2} \boxed{\| x - f \|^2}$$

Где:

- $\| x - f \|^2$ - L^2 тяготение к данным (наблюдаемому f)
- $\| \nabla x \|$ - полная вариация (регуляризация)

Что будет если убрать второе слагаемое?





TV-L2 Image Denoising (**ROF**)

Rudin-Osher-Fatemi model (ROF):

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

f - данные

x - искомое

Где:

- $\| x - f \|^2$ - L^2 тяготение к данным (наблюдаемому f)
- $\| \nabla x \|$ - полная вариация (регуляризация)

Что такое λ? Как ее выбрать?

TV-L2 Image Denoising (**ROF**)

Rudin-Osher-Fatemi model (ROF):

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

f - данные

x - искомое

Где:

- $\| x - f \|^2$ - L^2 тяготение к данным (наблюдаемому f)
- $\| \nabla x \|$ - полная вариация (регуляризация)
- λ - параметр регуляризации (вес данных)

TV-L2 = Total Variation + L^2 невязка по данным

Справится ли минимизация такого функционала/энергии с шумами?

TV-L2 Image Denoising (**ROF**)

Rudin-Osher-Fatemi model (ROF):

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

Где:

- $\| x - f \|^2$ - L^2 тяготение к данным (наблюдаемому f)
- $\| \nabla x \|$ - полная вариация (регуляризация)
- λ - параметр регуляризации (вес данных)

f - данные

x - искомое

TV-L2 = Total Variation + L^2 невязка по данным

Справится ли минимизация такого функционала/энергии с шумами?
А с выбросами?

TV-L2 Image Denoising (**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):

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

Итеративная численная схема

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

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

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

$$K x_n = \nabla x_n$$

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

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

TV-L2 Image Denoising (**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):

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

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

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

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

$$K x_n = \nabla x_n$$

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

На каждой итерации - два шага (Primal-Dual)

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

TV-L2 Image Denoising (**ROF**)

Primal-Dual алгоритм минимизации в общем виде решает:

$$\min_x \boxed{F(Kx)} + \boxed{G(x)}$$

Где: $\boxed{F}$ $\boxed{G}$ - выпуклые функции, K - линейный оператор.

Тогда итеративная схема (модель - выпуклая):

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

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

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

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

$$K x_n = \nabla x_n$$

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

На каждой итерации - два шага (Primal-Dual)

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

TV-L2 Image Denoising (**ROF**)

$\boxed{\text{Primal}}$-$\boxed{\text{Dual}}$ алгоритм минимизации в общем виде решает:

$$\min_x \boxed{F(Kx)} + \boxed{G(x)}$$

Где: $\boxed{F}$ $\boxed{G}$ - выпуклые функции, K - линейный оператор.

Тогда итеративная схема (модель - выпуклая):

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

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

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

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

$$K x_n = \nabla x_n$$

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

На каждой итерации - два шага (Primal-Dual)

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

TV-L2 Image Denoising (**ROF**)

Primal-Dual алгоритм минимизации в общем виде решает:

$$\min_x \boxed{F(Kx)} + \boxed{G(x)}$$

Где: $\boxed{F}$ $\boxed{G}$ - выпуклые функции, K - линейный оператор.

Тогда итеративная схема (модель - выпуклая):

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

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

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

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

$$K x_n = \nabla x_n$$

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

На каждой итерации - два шага (Primal-Dual)

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

TV-L2 Image Denoising (**ROF**)

$\boxed{\text{Primal}}$-$\boxed{\text{Dual}}$ алгоритм минимизации в общем виде решает:

$$\min_x \boxed{F(Kx)} + \boxed{G(x)}$$

Где: $\boxed{F}$ $\boxed{G}$ - выпуклые функции, K - линейный оператор.

Тогда итеративная схема (модель - выпуклая):

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

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

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

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

$$\boxed{K x_n = \nabla x_n}$$

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

На каждой итерации - два шага (Primal-Dual)

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

TV-L2 Image Denoising (**ROF**)

Primal-Dual алгоритм минимизации в общем виде решает:

$$\min_x \boxed{F(Kx)} + \boxed{G(x)}$$

Где: $\boxed{F}$ $\boxed{G}$ - выпуклые функции, K - линейный оператор.

Тогда итеративная схема (модель - выпуклая):

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

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

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

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

$$\boxed{K x_n = \nabla x_n}$$

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

На каждой итерации - два шага (Primal-Dual)
Каждый пиксель зависит только от себя и соседей

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

TV-L2 Image Denoising (**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):

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

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

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

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

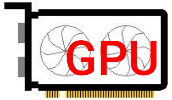
$$K x_n = \nabla x_n$$

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

На каждой итерации - два шага (Primal-Dual)
Каждый пиксель зависит только от себя и соседей
Как ускорить сходимость?

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

TV-L2 Image Denoising (**ROF**)

Primal-Dual алгоритм минимизации в общем виде решает: 

$$\min_x \boxed{F(Kx)} + \boxed{G(x)}$$

Где: $\boxed{F}$ $\boxed{G}$ - выпуклые функции, K - линейный оператор.

Тогда итеративная схема (модель - выпуклая):

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

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

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

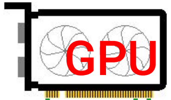
$$\boxed{\mathbf{project}_P(p) = \frac{p}{max(\|p\|, 1)}}$$

$$\boxed{K x_n = \nabla x_n}$$

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

На каждой итерации - два шага (Primal-Dual)

Каждый пиксель зависит только от себя и соседей

Как ускорить сходимость? 

+ Coarse-to-Fine!

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

TV-L2 Image Denoising (**ROF**)

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

$\lambda = 4$

$\lambda = 8$

TV-L2 Image Denoising (**ROF**)

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

$\lambda = 8$

$\lambda = 4$

TV-L1 Image Denoising

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

Как изменится результат?

TV-L1 Image Denoising

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

Более устойчивы к выбросам - не идем им на поводу.

TV-L1 Image Denoising

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

$\lambda = 1$

$\lambda = 1$

TV-L1 Image Denoising

$$\min_x \| \nabla x \| + \lambda \boxed{\sum_i} \| x - \boxed{f_i} \|$$

5 очень шумных наблюдений:

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

TV-L1 Image Denoising

Primal-Dual алгоритм минимизации в общем виде решает:

$$\min_x F(Kx) + \boxed{G(x)}$$

невязка по данным

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

Тогда итеративная схема (модель - выпуклая):

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

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

$$(I + \tau\partial \boxed{G_{TV-L1}})^{-1}(x) = \mathbf{shrink}(x, f, \lambda\tau)$$

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

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

$$K x_n = \nabla x_n$$

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

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

TV-L2 Image Denoising (**ROF**)

Primal-Dual алгоритм минимизации в общем виде решает:

$$\min_x F(Kx) + \boxed{G(x)}$$ невязка по данным

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

Тогда итеративная схема (модель - выпуклая):

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

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

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

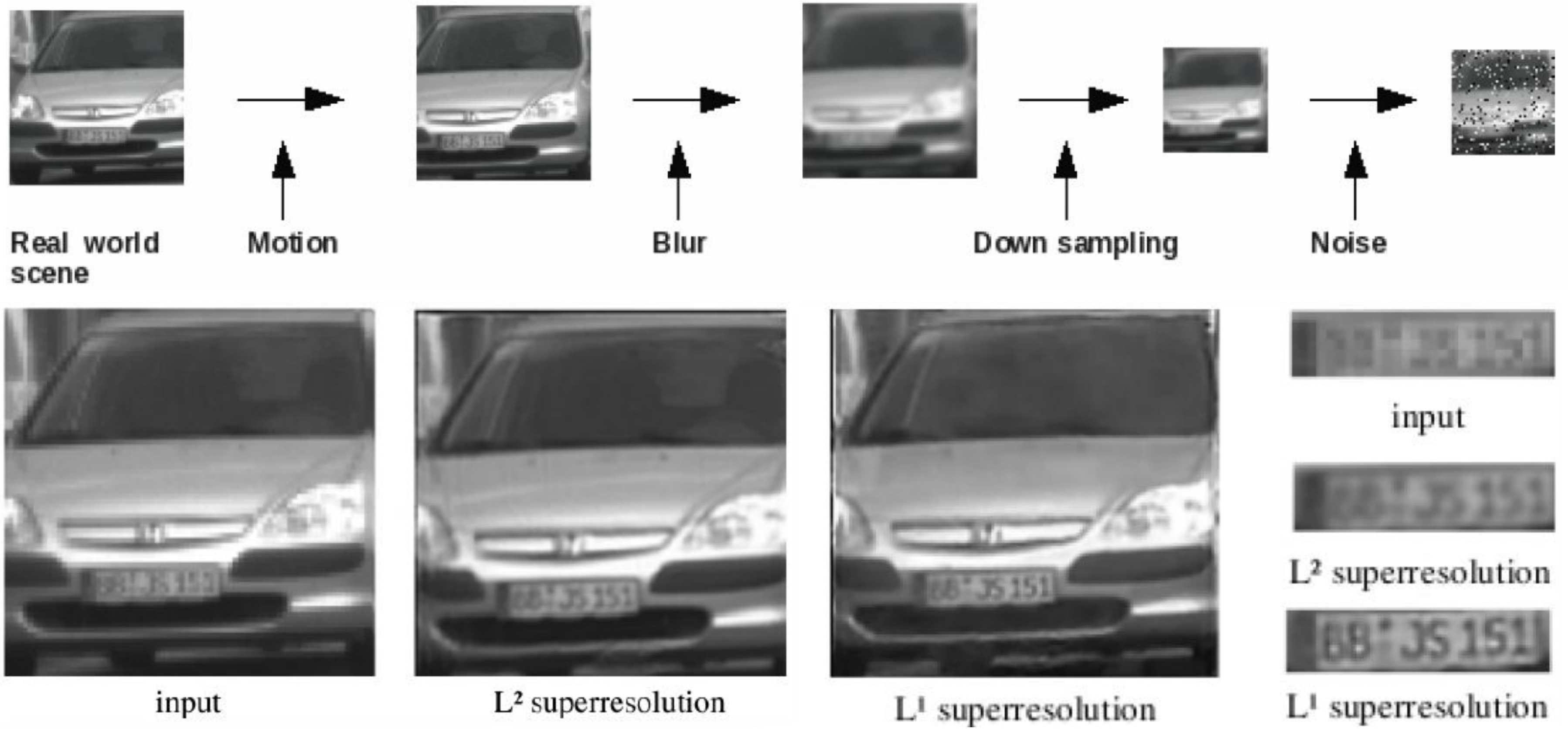
$$\mathbf{project}_P(p) = \frac{p}{max(\| p \|, 1)}$$

$$K x_n = \nabla x_n$$

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

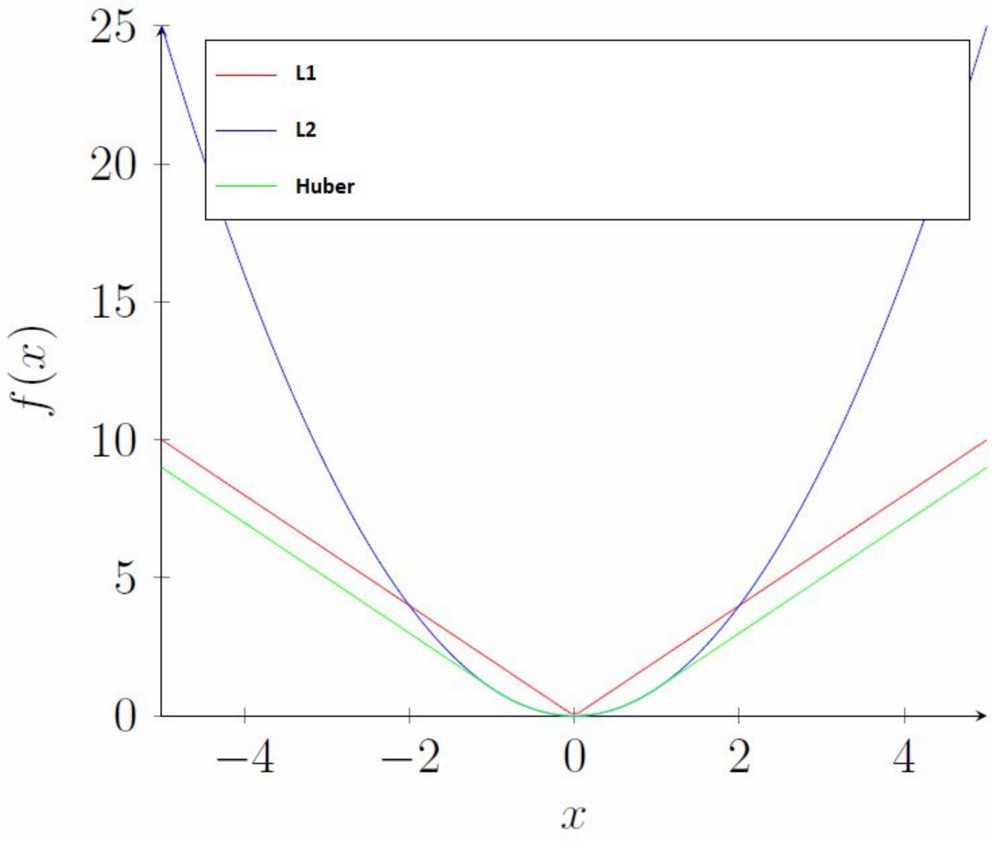
Подробнее: [IPython notebook with ROF and TVL1 denoising (with math!)]

Image super resolution from multiple images (**TV-L1, TV-L2**)



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

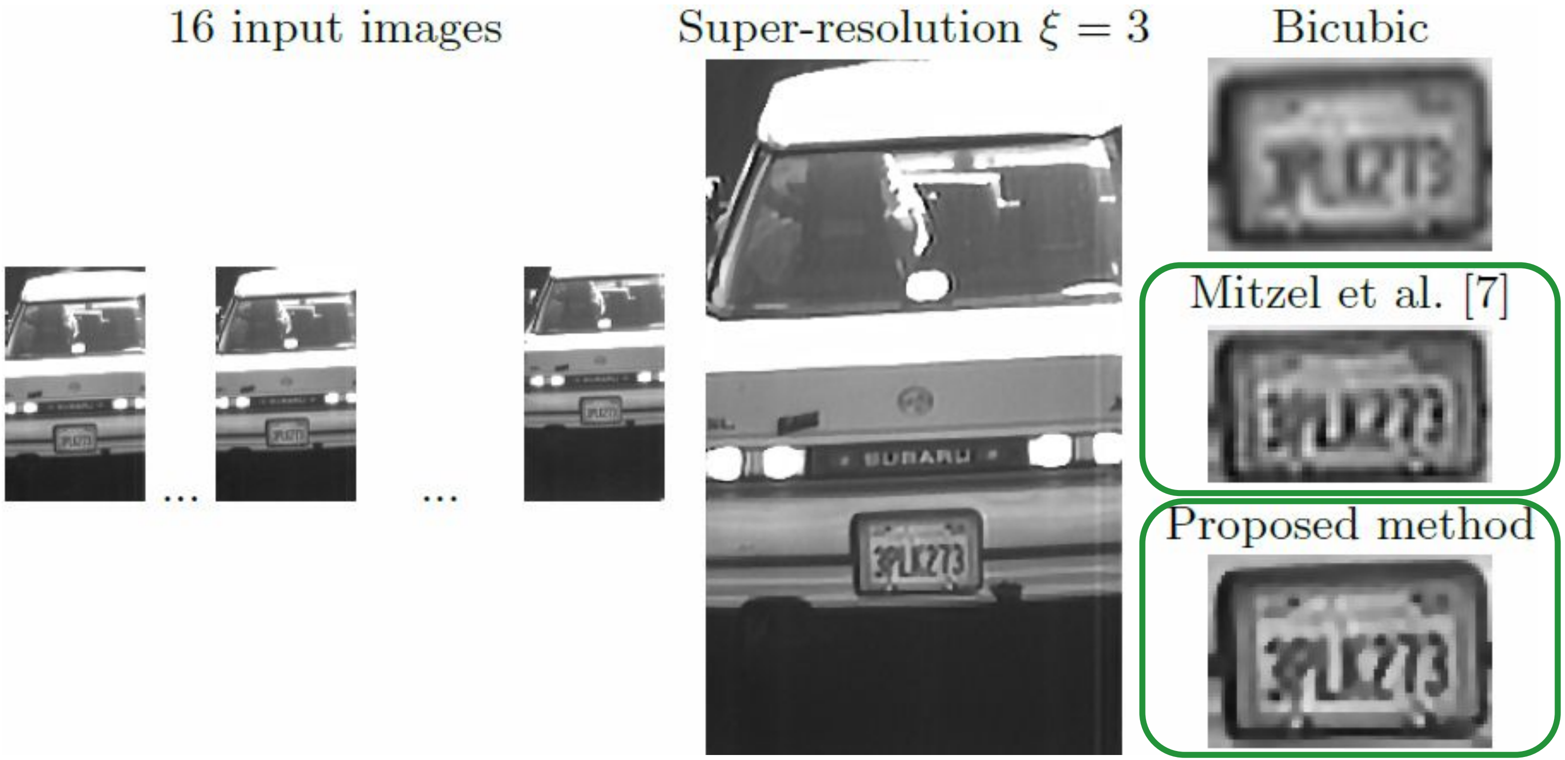
Image super resolution from multiple images (**Huber loss**)



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

Image super resolution from multiple images (**Huber loss**)

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



TV-L1

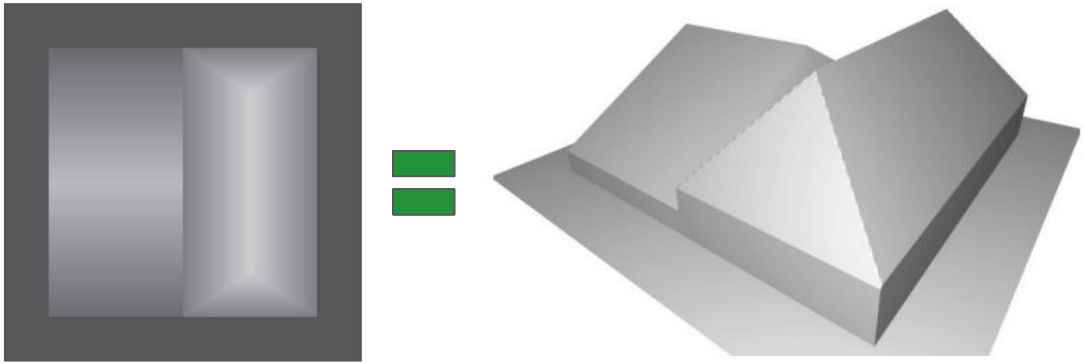
Huber

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

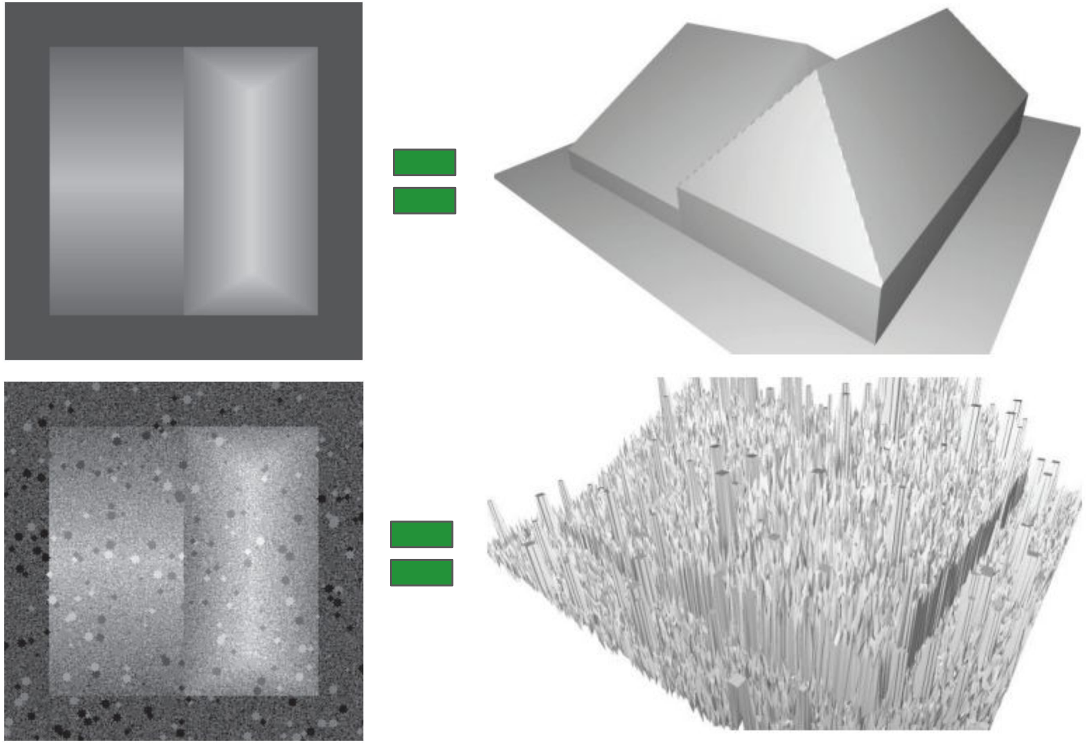
Ссылки: TV-L2 (ROF), TV-L1, Primal-Dual минимизация

- 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)
- Video Super Resolution using Duality Based TV-L1 Optical Flow, Mitzel et al., 2009
- A Convex Approach for Variational Super-Resolution, Unger et al., 2010

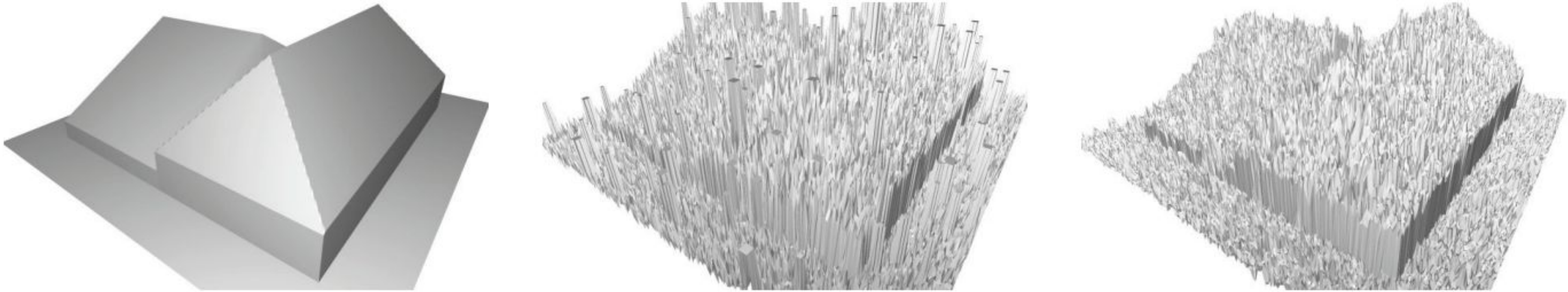
DSM (**D**igital **S**urface **M**odel, 2.5D карта высот)



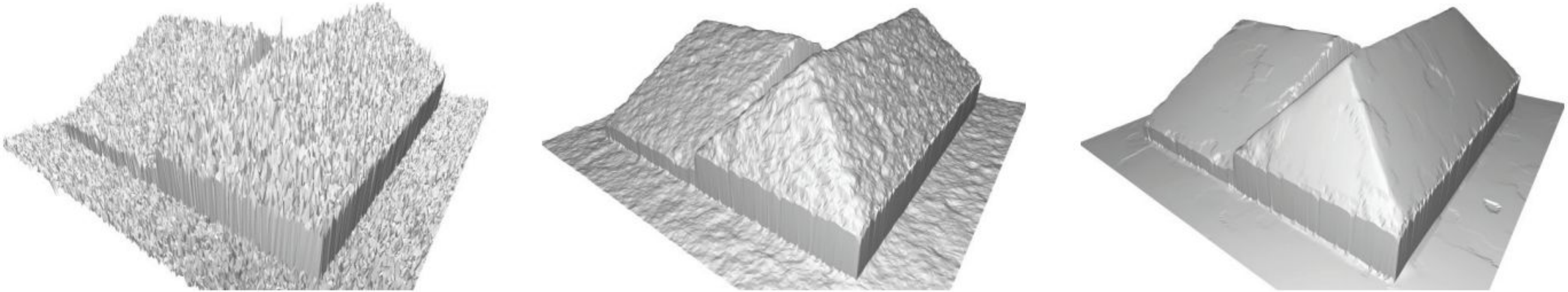
DSM (**D**igital **S**urface **M**odel, 2.5D карта высот)



DSM из 5 шумных наблюдений



(a) Ground truth (b) Input image (c) Average



(d) Median (e) Huber (f) TGV^2

DSM из 5 шумных наблюдений (**TV-L1, Huber, TGV**)

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

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

Полная вариация **Huber loss**

DSM из 5 шумных наблюдений (**TV-L1, Huber, TGV**)

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

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

Huber model:

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

Полная вариация (Huber)

Huber loss

DSM из 5 шумных наблюдений (**TV-L1, Huber, TGV**)

$\boldsymbol{TV\text{-}L^1}$ **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:**

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

Полная вариация v

Huber loss

DSM из 5 шумных наблюдений (**TV-L1, Huber, TGV**)

$\boldsymbol{TV\text{-}L^1}$ **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:**

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

Т.е. регуляризация второго порядка:

$$\alpha_1 \int_\Omega \boxed{|\nabla u - v|} dx \Rightarrow \boxed{v \approx \nabla u}$$

Huber loss

DSM из 5 шумных наблюдений (**TV-L1, Huber, TGV**)

$\boldsymbol{TV\text{-}L^1}$ **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:**

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

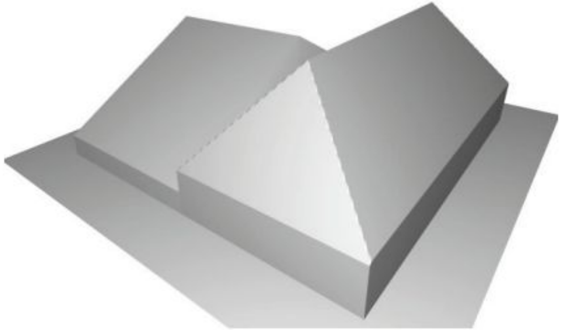
Т.е. регуляризация второго порядка:

$$\alpha_1 \int_\Omega \boxed{|\nabla u - v|} dx \quad \Rightarrow \quad \boxed{v \approx \nabla u}$$

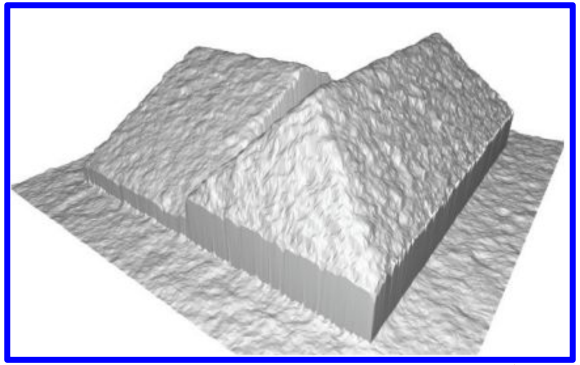
Huber loss

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

DSM из 5 шумных наблюдений (**TV-L1, Huber, TGV**)



(a) Ground truth



(e) Huber

TGV^2 **model:** $\min_{u,v}\left\{\alpha_1\int_\Omega \boxed{|\nabla u - v|}dx + \alpha_0\int_\Omega \boxed{|\mathcal{E}(v)|}dx + \sum_{l=1}^{K}\int_\Omega \boxed{|u - f_l|_\delta} dx\right\}$

Т.е. регуляризация второго порядка:

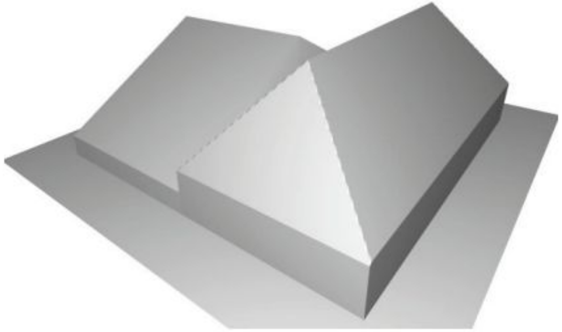
Что нам здесь не нравится?

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

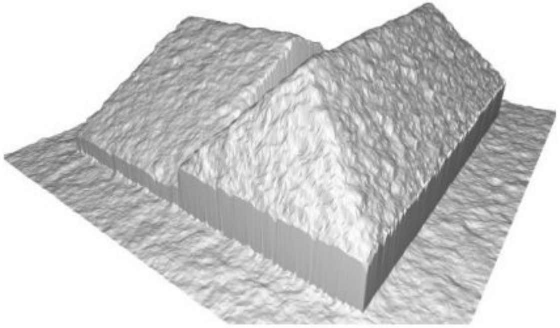
Huber loss

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

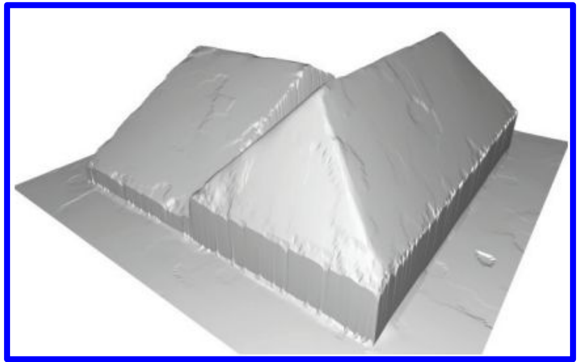
DSM из 5 шумных наблюдений (**TV-L1, Huber, TGV**)



(a) Ground truth



(e) Huber



(f) TGV^2

TGV^2 **model:** $\min_{u,v} \left\{ \alpha_1 \int_\Omega \boxed{|\nabla u - v|} dx + \alpha_0 \int_\Omega \boxed{|\mathcal{E}(v)|} dx + \sum_{l=1}^{K} \int_\Omega \boxed{|u - f_l|_\delta} dx \right\}$

Т.е. регуляризация второго порядка:

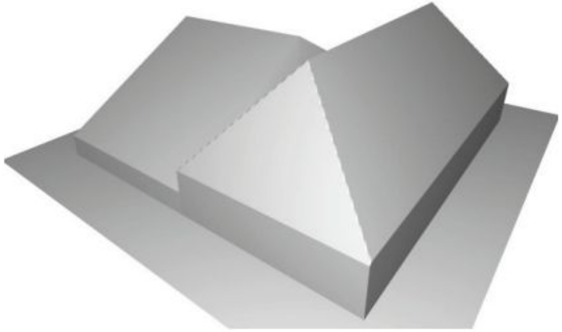
$\alpha_1 \int_\Omega \boxed{|\nabla u - v|} dx$ => $\boxed{v \approx \nabla u}$

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

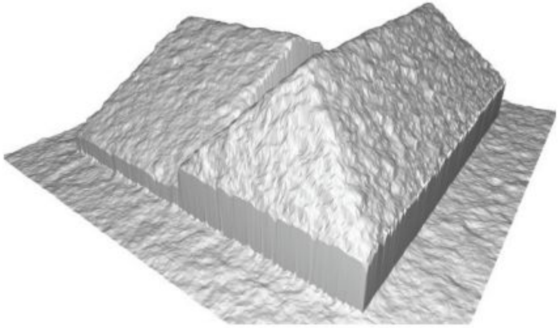
Что же мы хотим минимизировать?
Какое априорное требование?

Huber loss

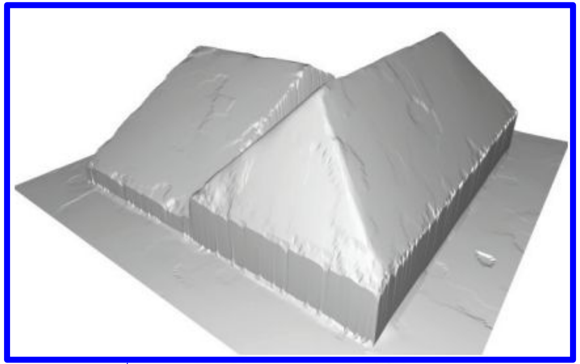
DSM из 5 шумных наблюдений (**TV-L1, Huber, TGV**)



(a) Ground truth



(e) Huber



(f) TGV^2

TGV^2 **model:** $$\min_{u,v}\left\{\alpha_1\int_\Omega \boxed{|\nabla u - v|}\,dx + \alpha_0\int_\Omega \boxed{|\mathcal{E}(v)|}\,dx + \sum_{l=1}^{K}\int_\Omega \boxed{|u - f_l|_\delta}\,dx\right\}$$

Т.е. регуляризация второго порядка:

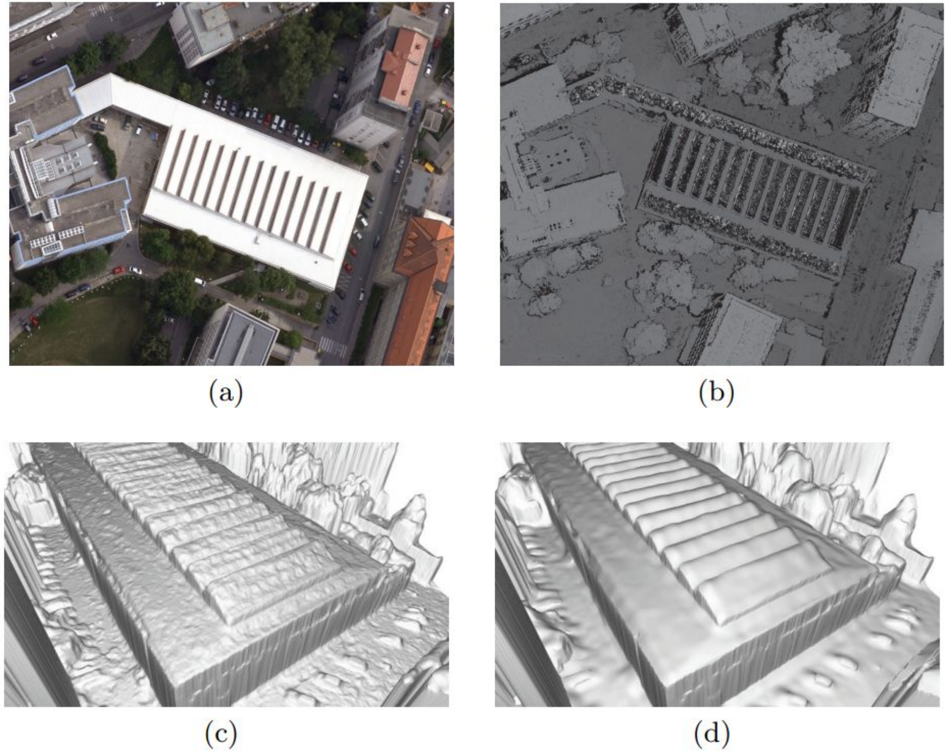
$$\alpha_1\int_\Omega \boxed{|\nabla u - v|}\,dx \;=>\; \boxed{v \approx \nabla u}$$

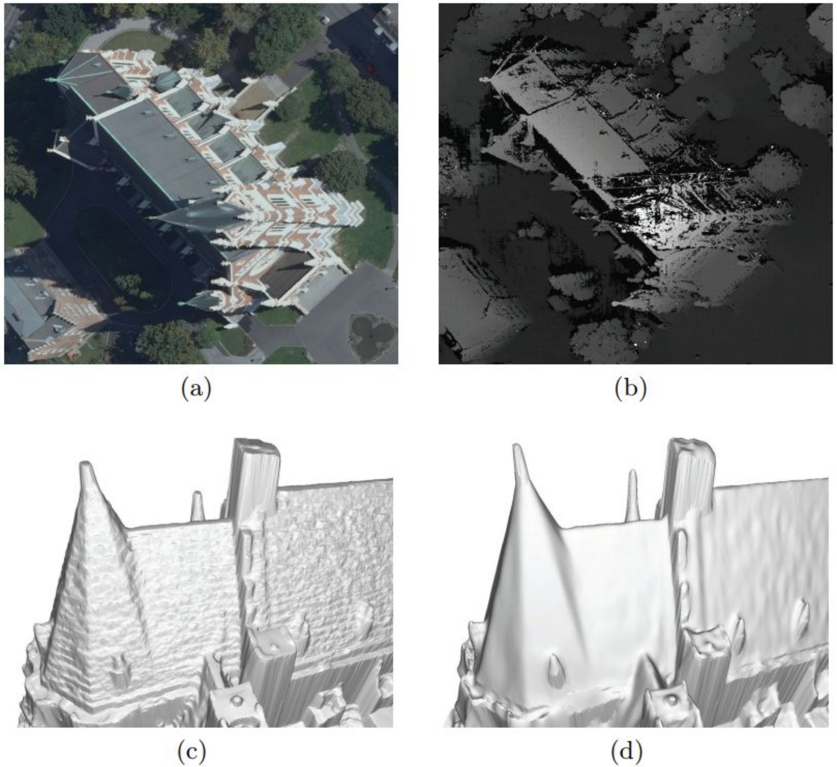
Хотим гладкую поверхность. Поэтому минимизируем перепады нормалей.

Huber loss

$\alpha_0\int_\Omega \boxed{|\mathcal{E}(v)|}\,dx$ - полная вариация v ≈ $\boxed{\text{полная вариация } \nabla u}$

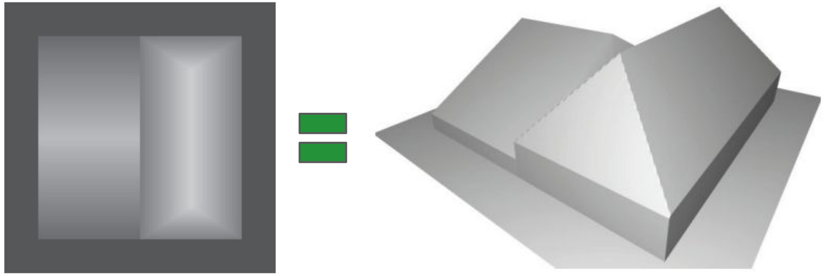
DSM из шумных наблюдений (**TGV-Huber**)





TGV-Fusion, Pock et al., 2011

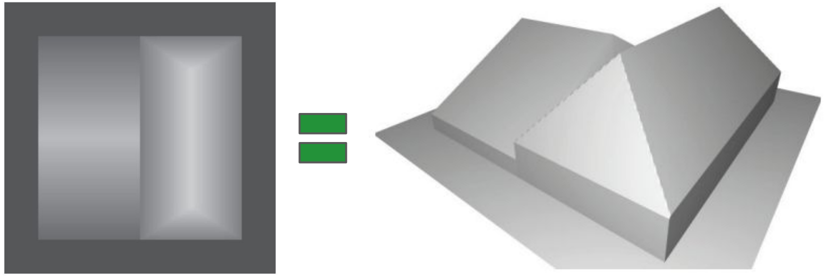
DSM (**2.5D** карта высот)



На вход: ???

На выход: одна карта (картинка высот)

DSM (**2.5D** карта высот)

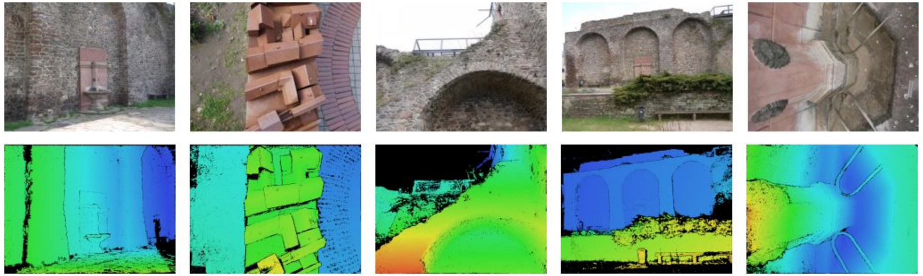


На вход: несколько неполных карт высоты
(каждая получена из карты глубины)

На выход: одна карта (картинка высот)

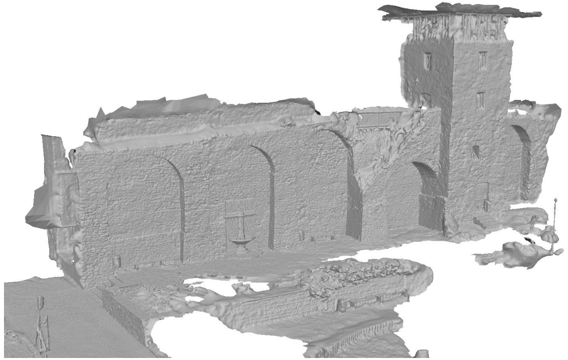
$$\min_u \left\{ \alpha \int_\Omega |\nabla u| dx + \boxed{\sum_{l=1}^{K} \int_\Omega |u - f_l|} dx \right\}$$

3D модель

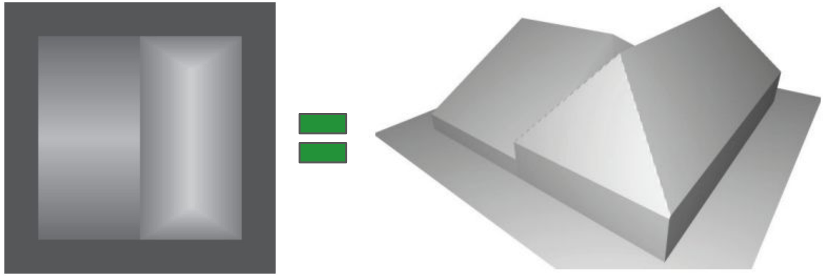


На вход: карты глубины**???**

На выход: полигональная модель**???**



DSM (2.5D карта высот)



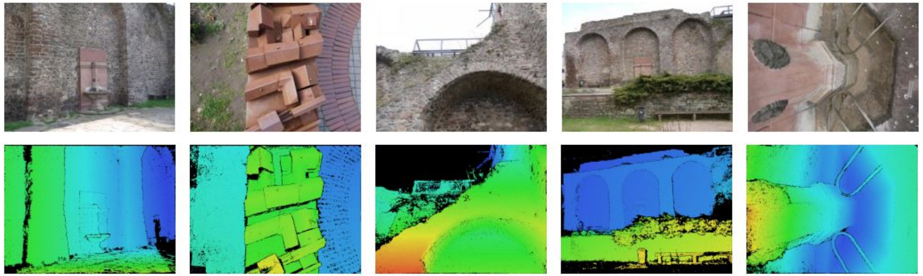
На вход: несколько неполных карт высоты
(каждая получена из карты глубины)

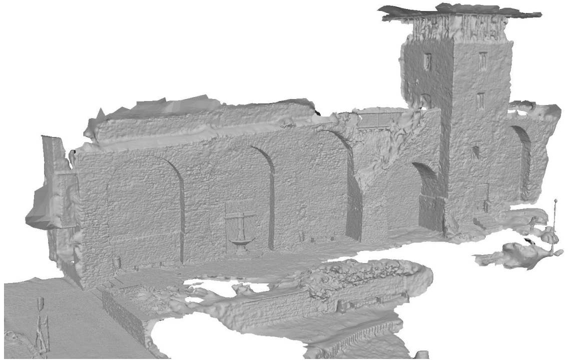
На выход: одна карта (картинка высот)

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

3D модель

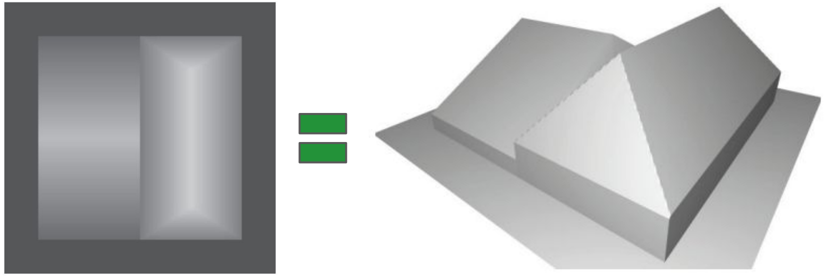
DSM (2.5D карта высот)





На вход: карты глубины**???**

На выход: полигональная модель**???**



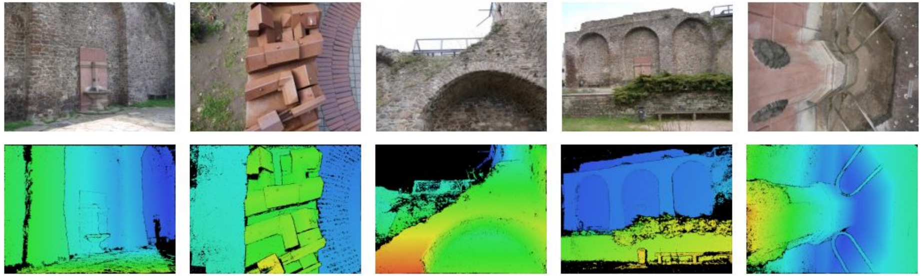
На вход: несколько неполных карт высоты
(каждая получена из карты глубины)

На выход: одна карта (картинка высот)

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

Какое 3D представление позволит легко сверить текущий ответ с входными данными?

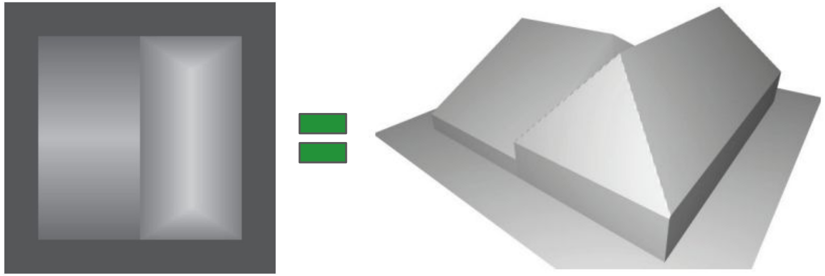
3D модель



На вход: карты глубины???

На выход: индикаторное скалярное поле

DSM (2.5D карта высот)

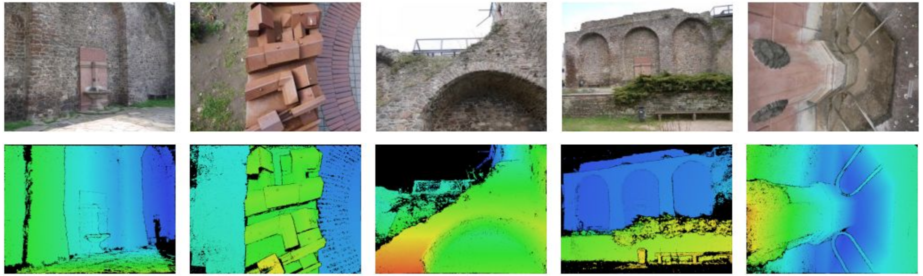


На вход: несколько неполных карт высоты
(каждая получена из карты глубины)

На выход: одна карта (картинка высот)

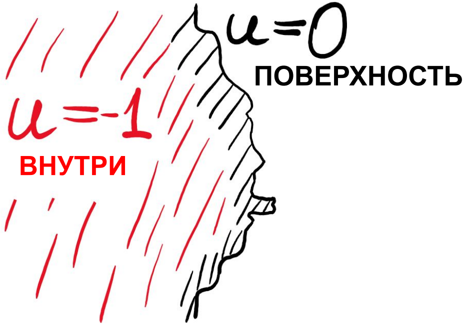
$$\min_u \left\{ \alpha \int_\Omega |\nabla u| dx + \boxed{\sum_{l=1}^{K} \int_\Omega |u - f_l|} dx \right\}$$

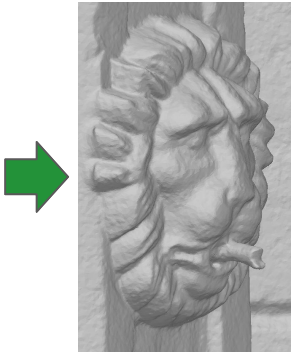
3D модель



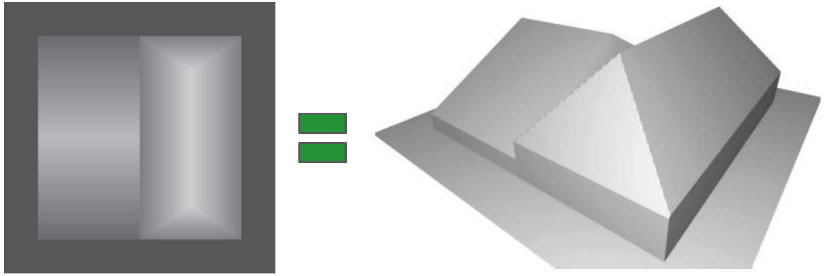
На вход: карты глубины???

На выход: индикаторное скалярное поле





DSM (2.5D карта высот)

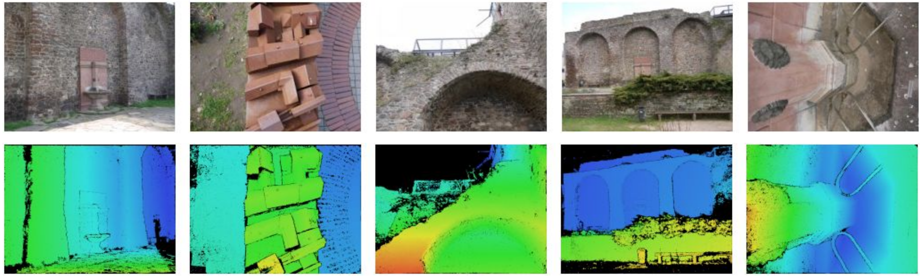


На вход: несколько неполных карт высоты
(каждая получена из карты глубины)

На выход: одна карта (картинка высот)

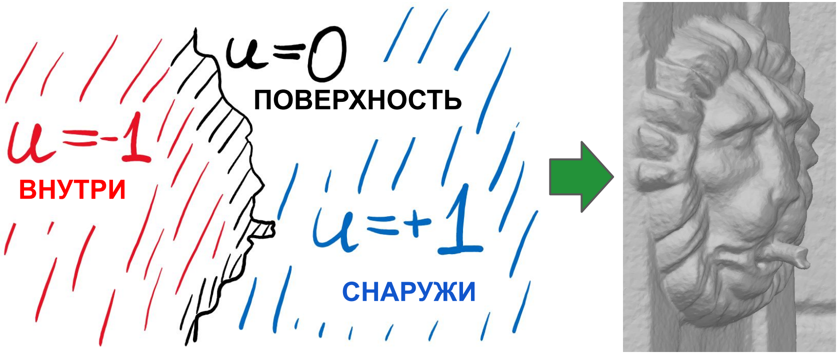
$$\min_u \left\{ \alpha \int_\Omega |\nabla u| dx + \boxed{\sum_{l=1}^{K} \int_\Omega |u - f_l|} dx \right\}$$

3D модель

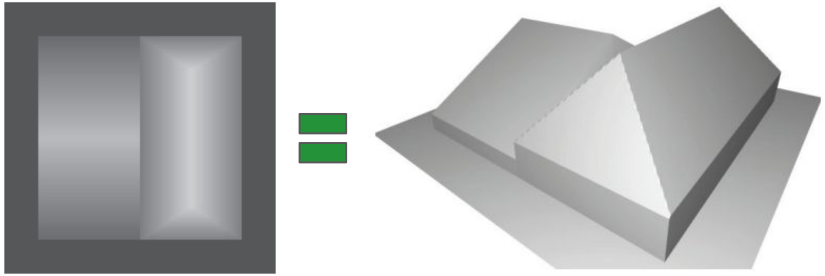


На вход: карты глубины???

На выход: индикаторное скалярное поле



DSM (2.5D карта высот)

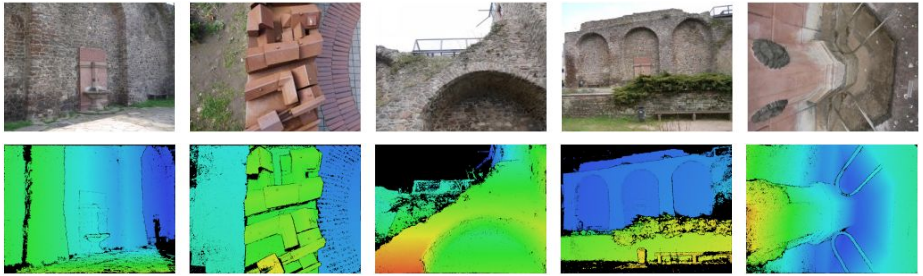


На вход: несколько неполных карт высоты
(каждая получена из карты глубины)

На выход: одна карта (картинка высот)

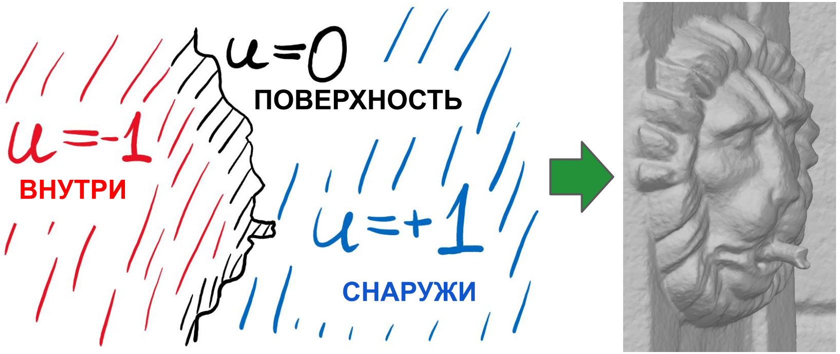
$$\min_u \left\{ \alpha \int_\Omega |\nabla u| dx + \boxed{\sum_{l=1}^{K} \int_\Omega |u - f_l|} dx \right\}$$

3D модель

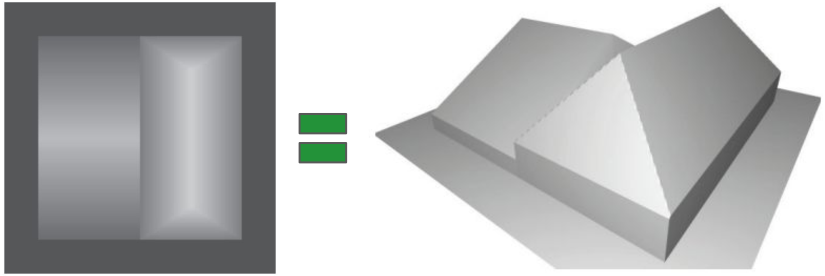


На вход: **карты глубины???**

На выход: индикаторное скалярное поле



DSM (2.5D карта высот)

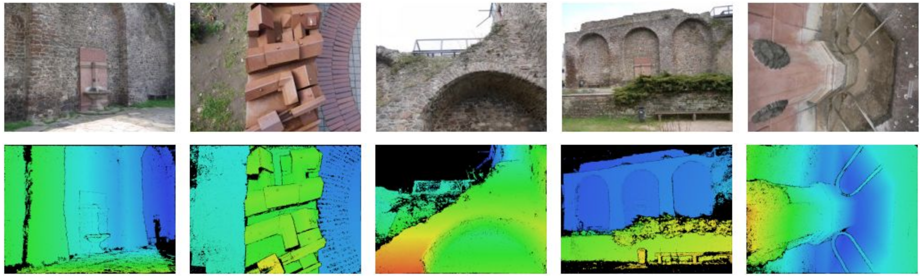


На вход: несколько неполных карт высоты (каждая получена из карты глубины)

На выход: одна карта (картинка высот)

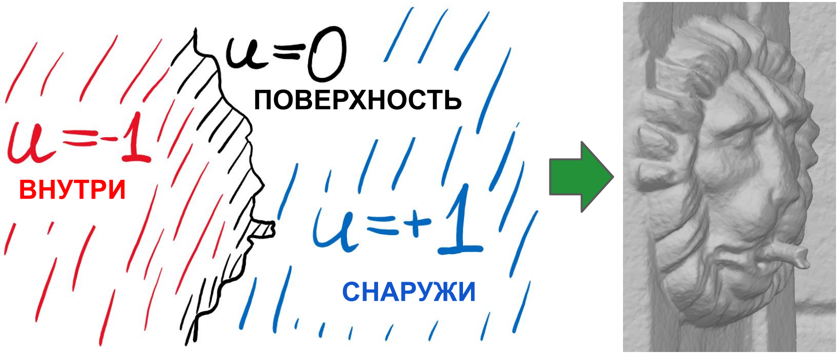
$$\min_u \left\{ \alpha \int_\Omega |\nabla u| dx + \boxed{\sum_{l=1}^{K} \int_\Omega |u - f_l|} dx \right\}$$

3D модель

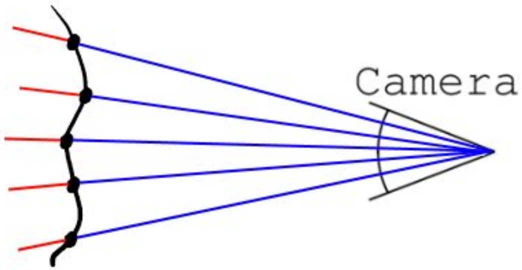


На вход: **карты глубины???**

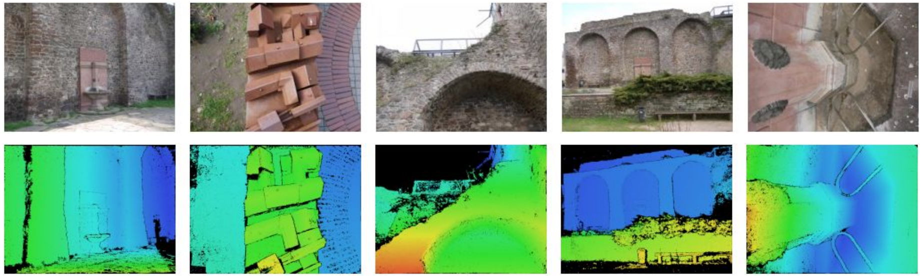
На выход: индикаторное скалярное поле



Карты глубины

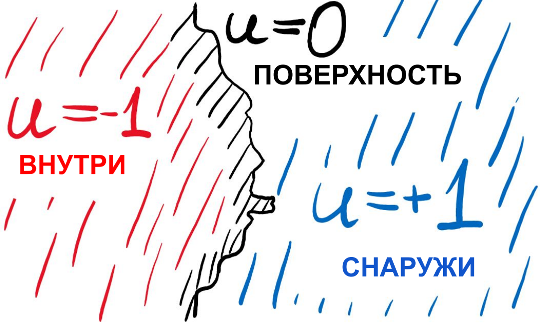


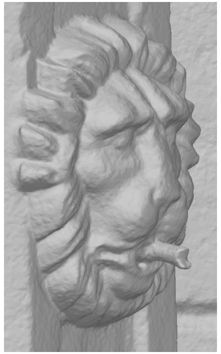
3D модель



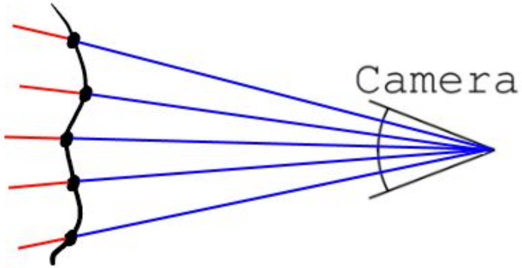
На вход: **карты глубины???**

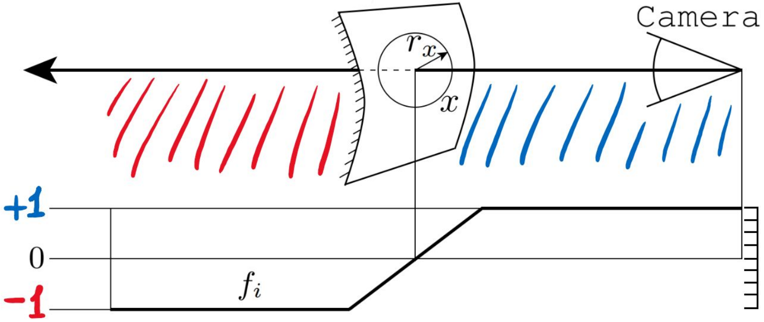
На выход: индикаторное скалярное поле





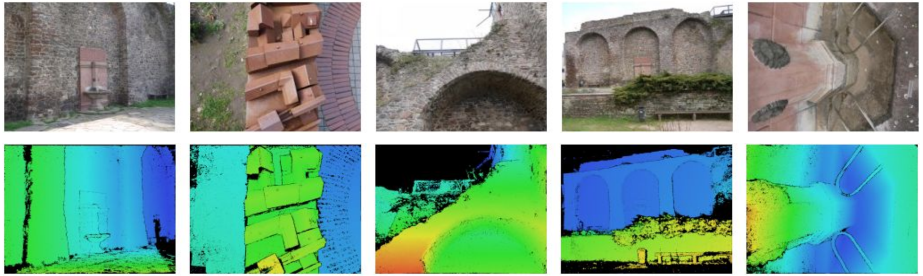
Карты глубины

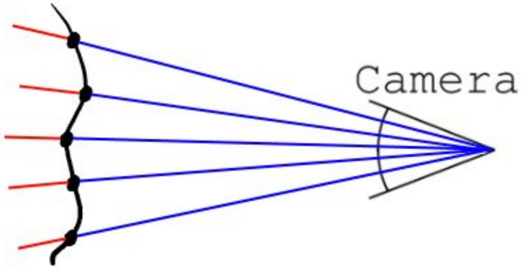




3D модель

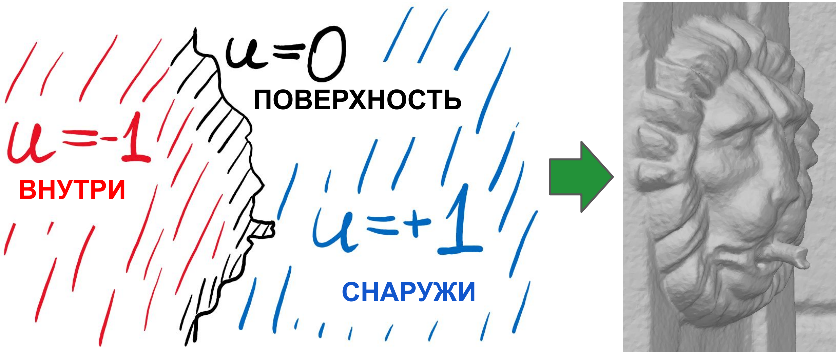
Карты глубины

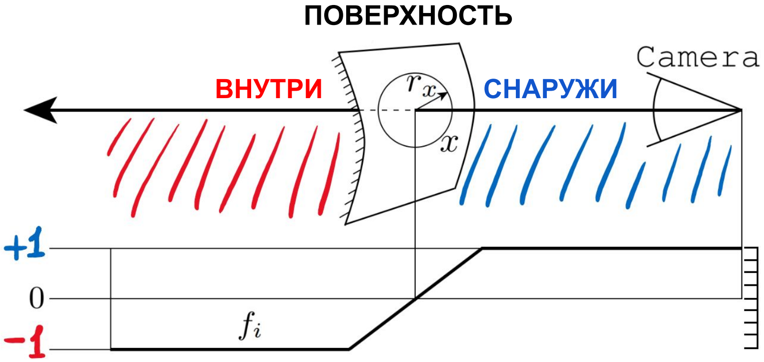




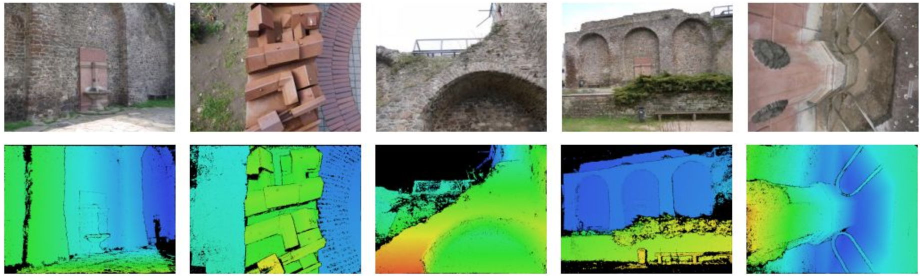
На вход: **карты глубины???**

На выход: индикаторное скалярное поле



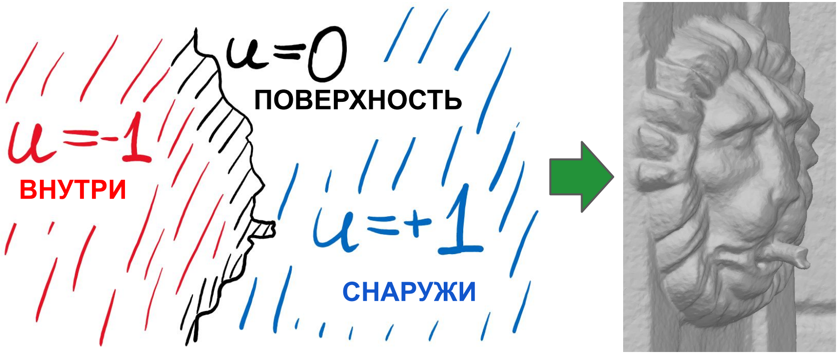


3D модель

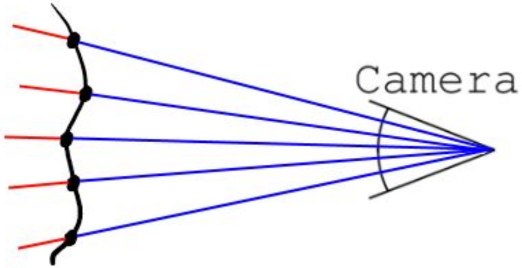


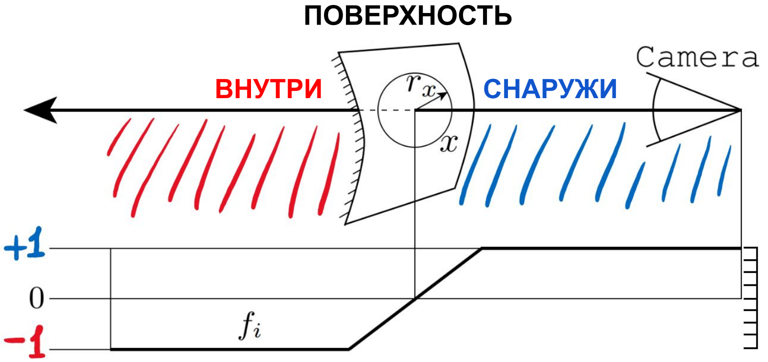
На вход: индикаторное скалярное поле

На выход: индикаторное скалярное поле

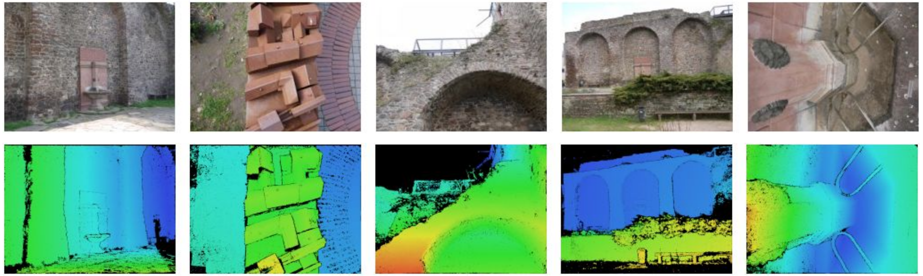


Карты глубины



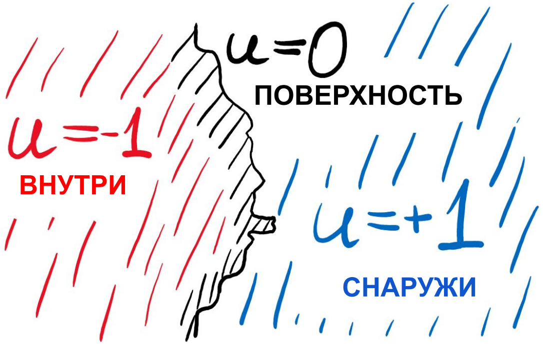


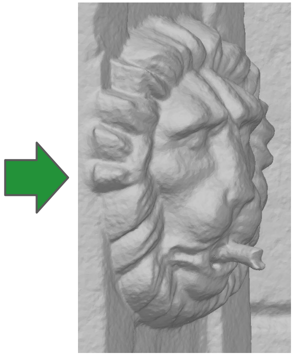
3D модель (TV-L1)



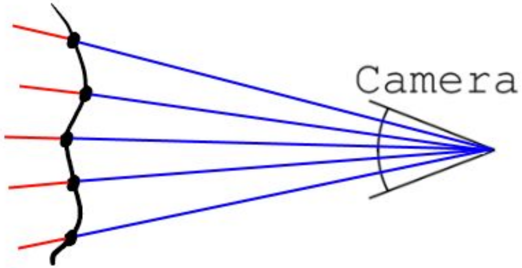
На вход: индикаторное скалярное поле

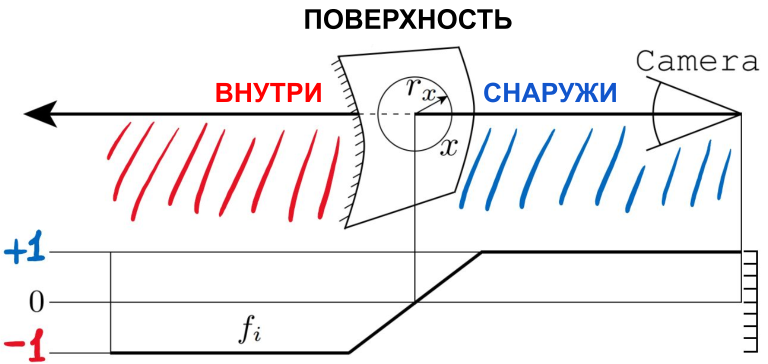
На выход: индикаторное скалярное поле





Карты глубины

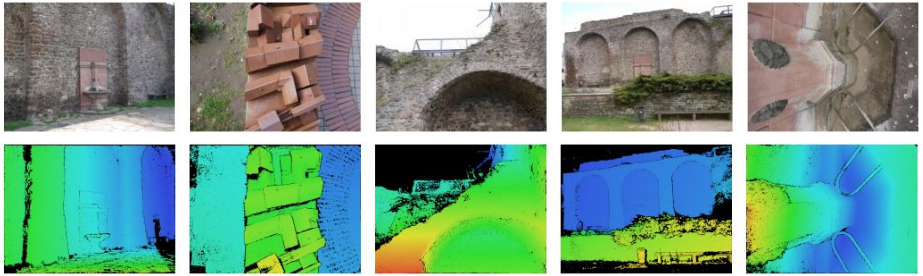




TV-L^1:

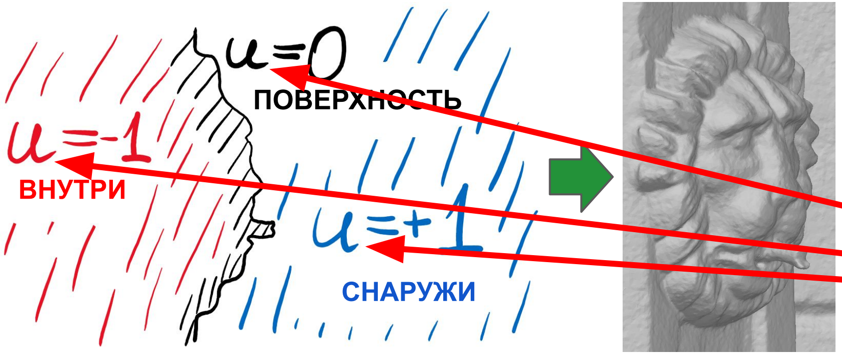
$$E = \int_\Omega \left\{ |\nabla u| + \lambda \sum_{i \in \mathcal{I}(\vec{x})} w_i(\vec{x}) |u - f_i| \right\} d\vec{x}$$

3D модель (TV-L1)

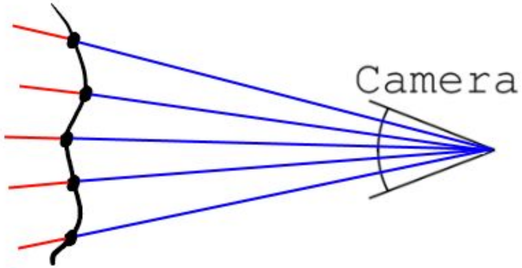


На вход: индикаторное скалярное поле

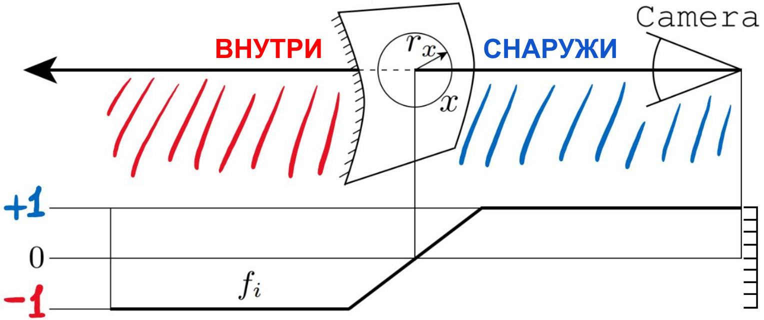
На выход: индикаторное скалярное поле



Карты глубины



ПОВЕРХНОСТЬ

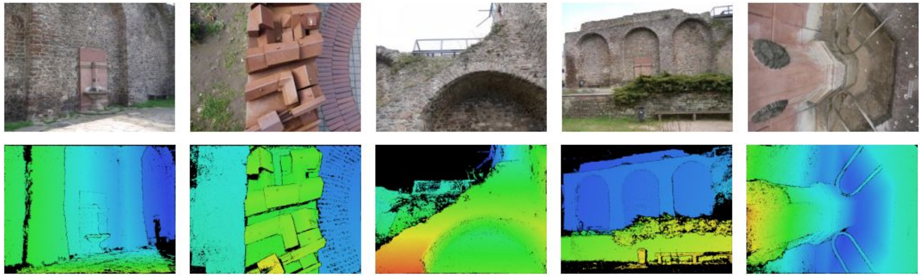


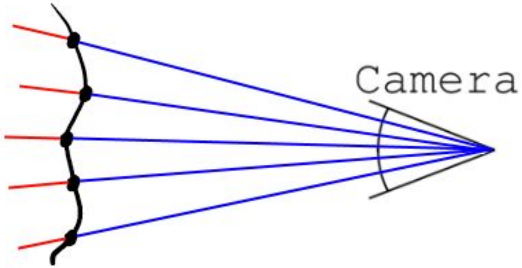
TV-L^1:

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

3D модель (TV-L1)

Карты глубины

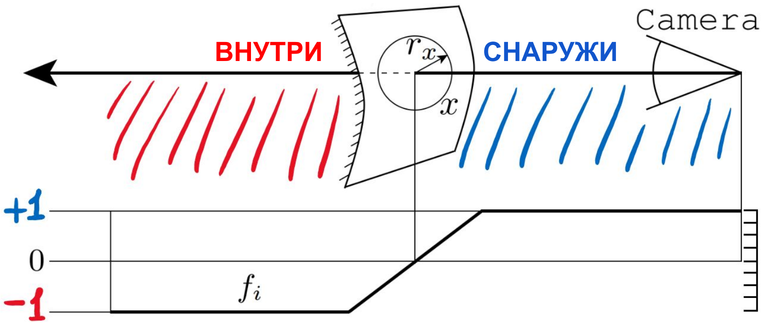


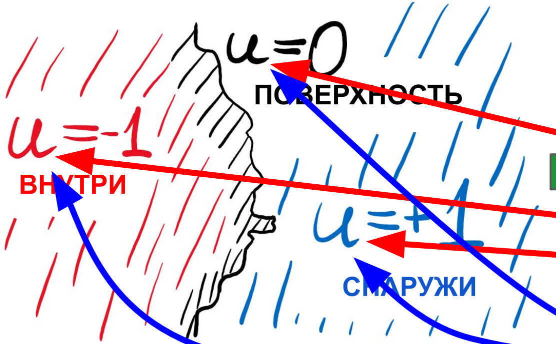


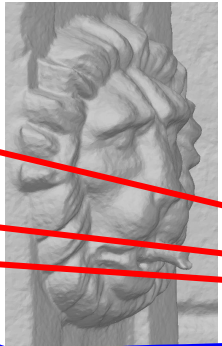
На вход: индикаторное скалярное поле

На выход: индикаторное скалярное поле

ПОВЕРХНОСТЬ





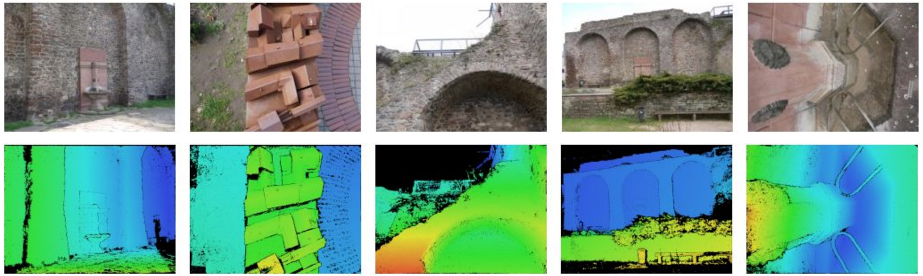


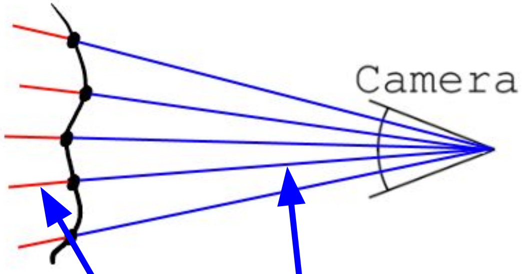
TV-L^1:

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

3D модель (TV-L1)

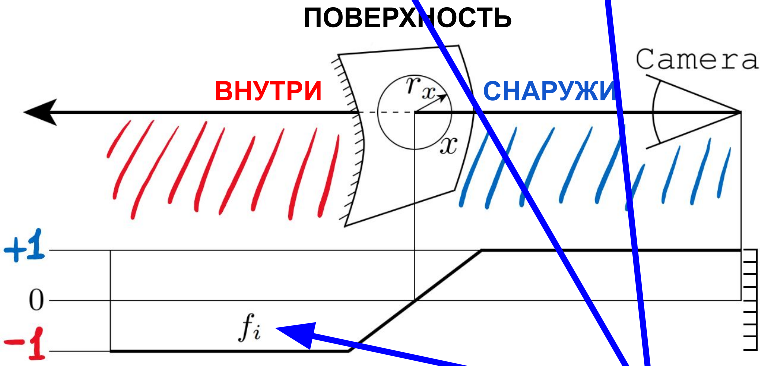
Карты глубины

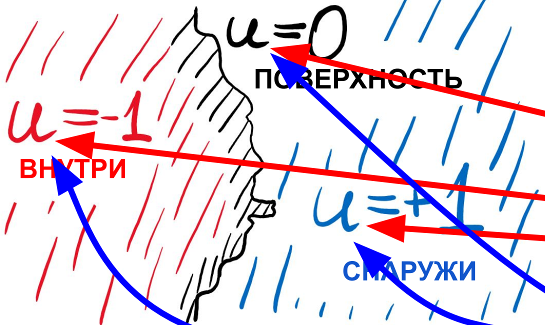


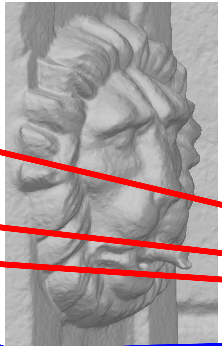


На вход: индикаторное скалярное поле

На выход: индикаторное скалярное поле





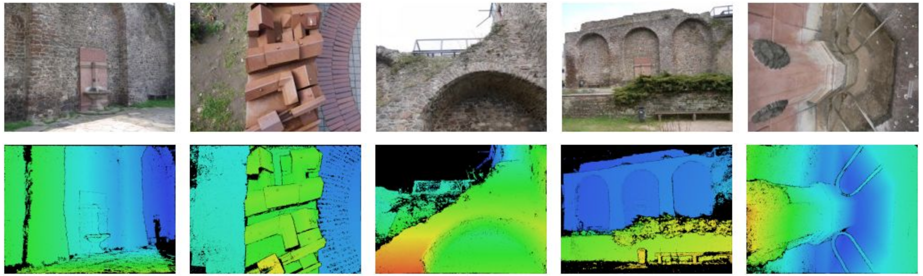


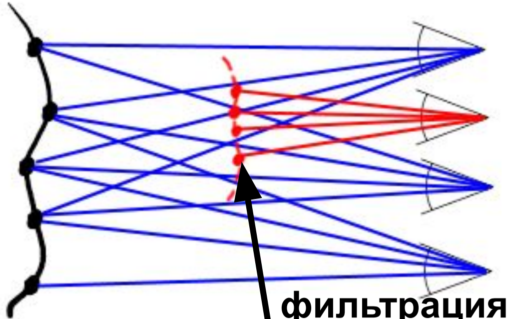
TV-L^1:

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

3D модель (TV-L1)

Карты глубины

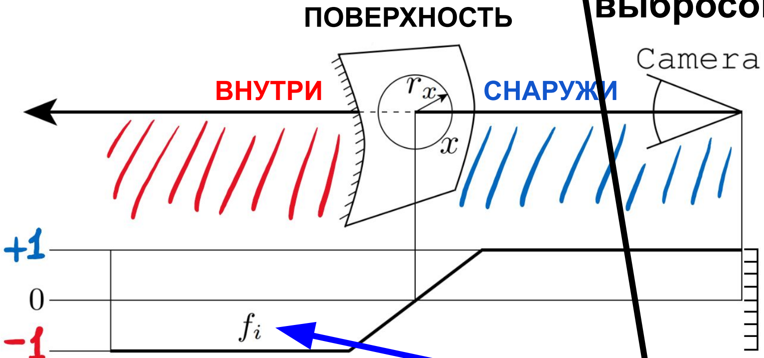


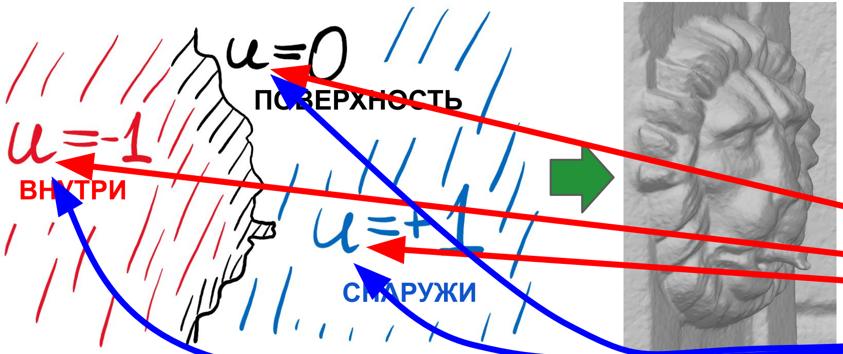


фильтрация выбросов

На вход: индикаторное скалярное поле

На выход: индикаторное скалярное поле

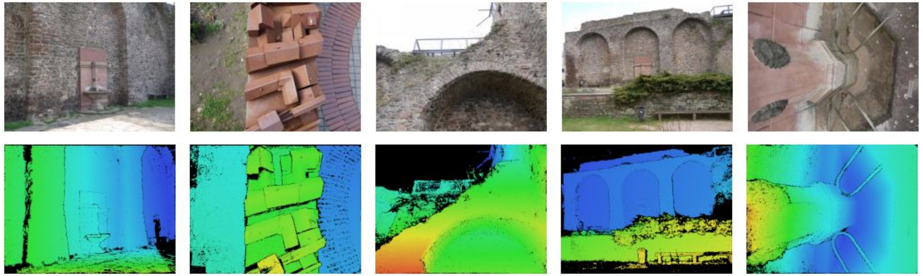




TV-L^1:

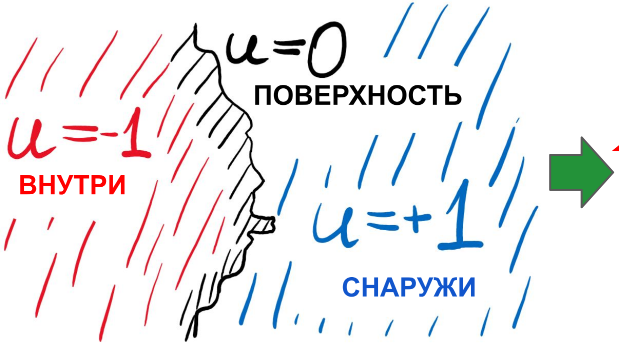
$$E = \int_\Omega \left\{ |\nabla u| + \lambda \sum_{i \in \mathcal{I}(\vec{x})} w_i(\vec{x}) |u - f_i| \right\} d\vec{x}$$

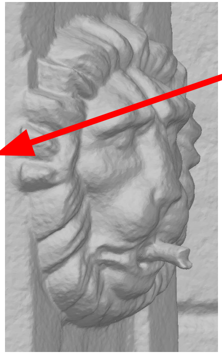
3D модель (TV-L1)



На вход: индикаторное скалярное поле

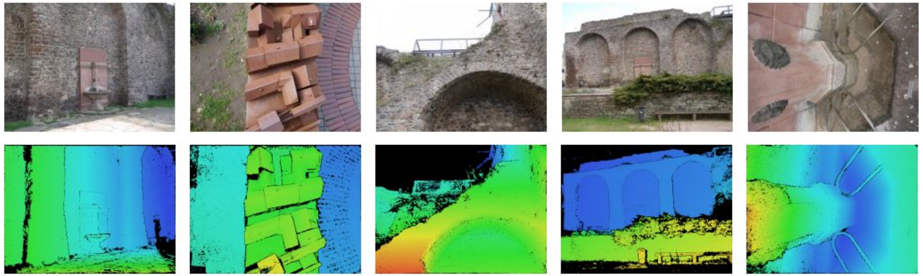
На выход: индикаторное скалярное поле





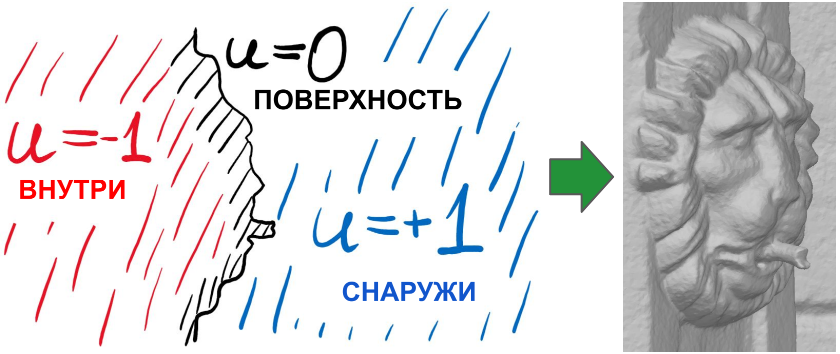
Как по найденному скалярному полю найти полигональную поверхность?

3D модель (TV-L1)

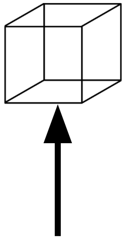


На вход: индикаторное скалярное поле

На выход: индикаторное скалярное поле

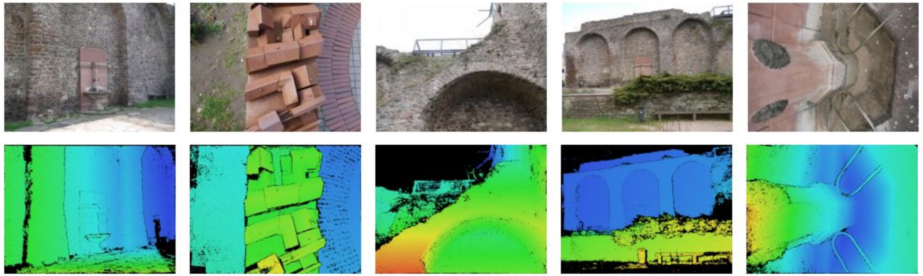


Маршировка кубов



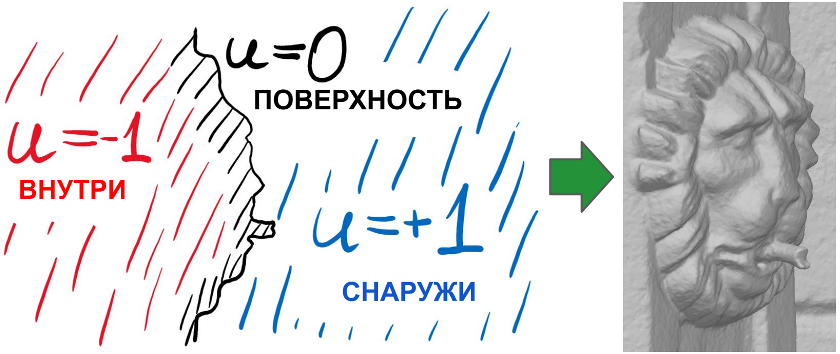
рассмотрим воксель нашего пространства

3D модель (TV-L1)

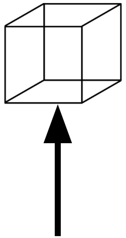


На вход: индикаторное скалярное поле

На выход: индикаторное скалярное поле



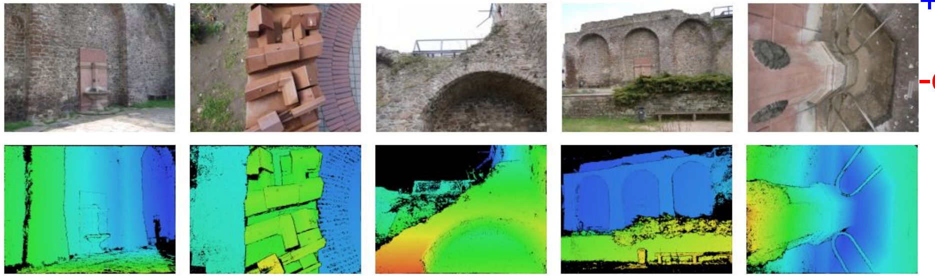
Маршировка кубов



рассмотрим воксель нашего пространства

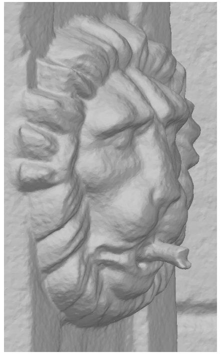
как зная индикаторное поле на его 8 углах понять где проходит поверхность?

3D модель (TV-L1)

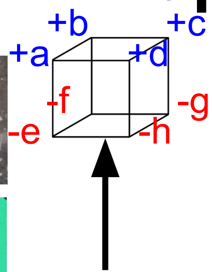


На вход: индикаторное скалярное поле

На выход: индикаторное скалярное поле



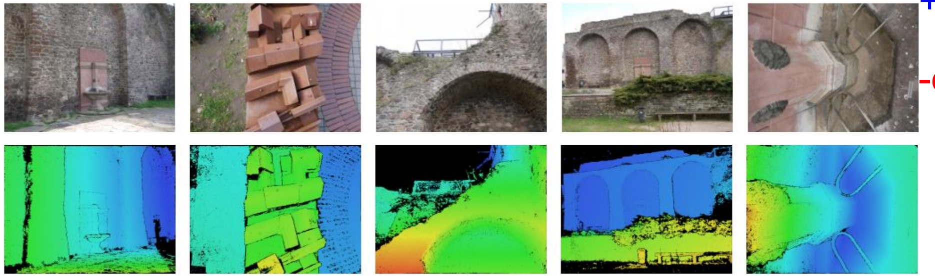
Маршировка кубов



рассмотрим воксель нашего пространства

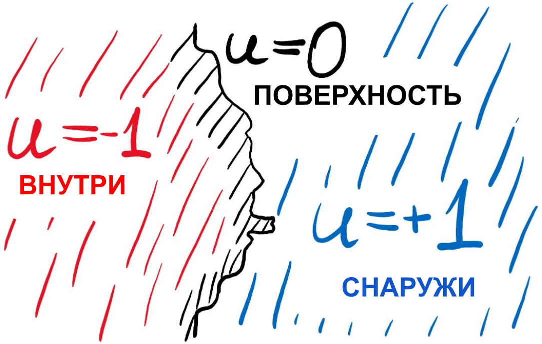
как зная индикаторное поле на его 8 углах понять где проходит поверхность?

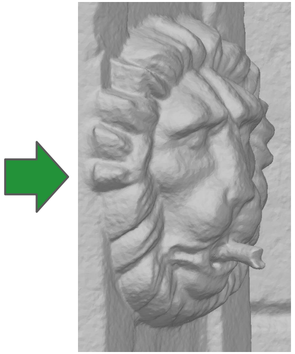
3D модель (TV-L1)



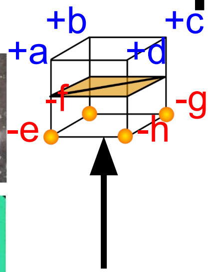
На вход: индикаторное скалярное поле

На выход: индикаторное скалярное поле





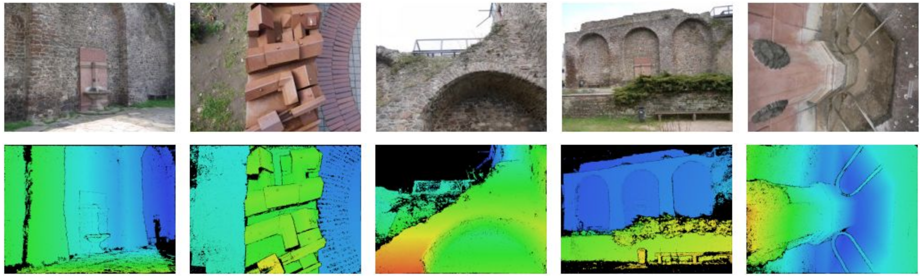
Маршировка кубов



рассмотрим воксель нашего пространства

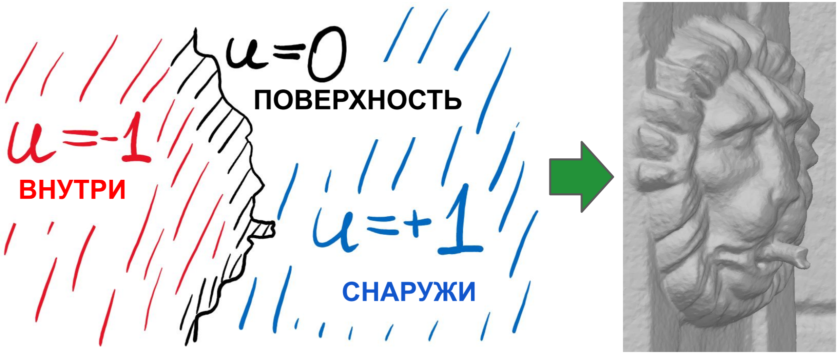
как зная индикаторное поле на его 8 углах понять где проходит поверхность?

3D модель (TV-L1)

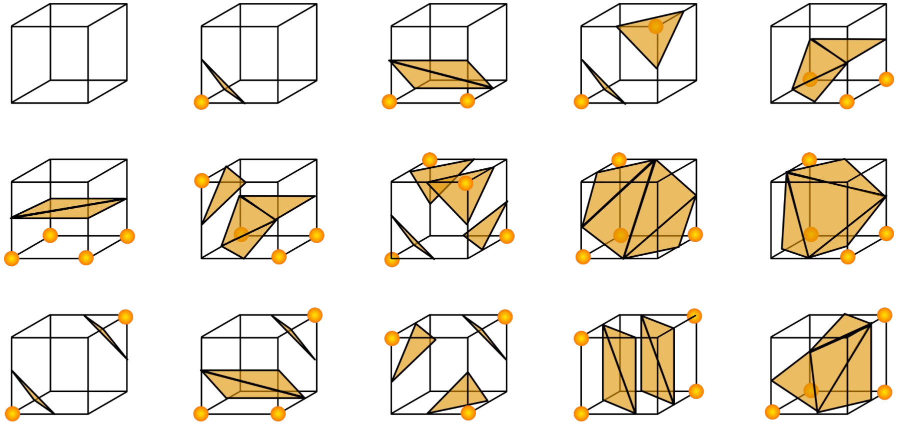


На вход: индикаторное скалярное поле

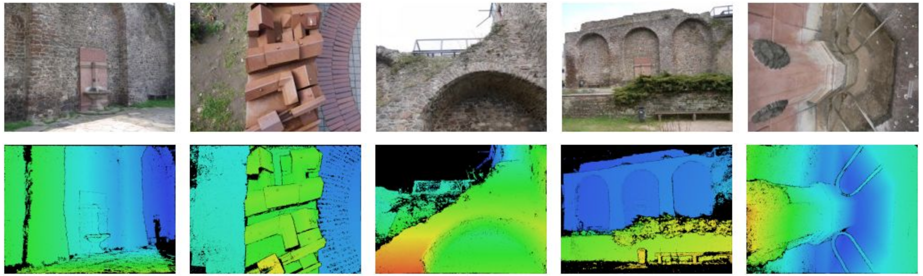
На выход: индикаторное скалярное поле



Маршировка кубов

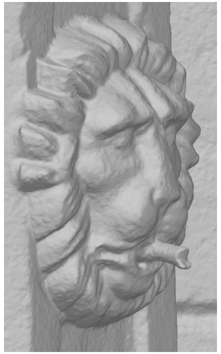


3D модель (TV-L1)

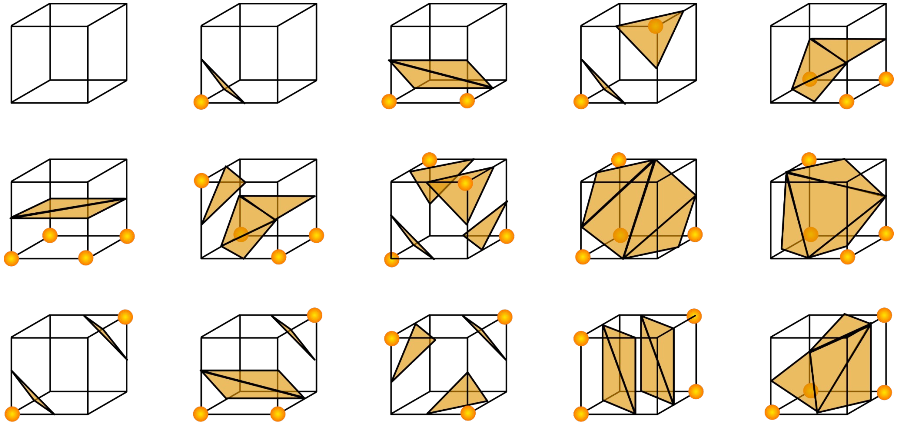


На вход: индикаторное скалярное поле

На выход: индикаторное скалярное поле

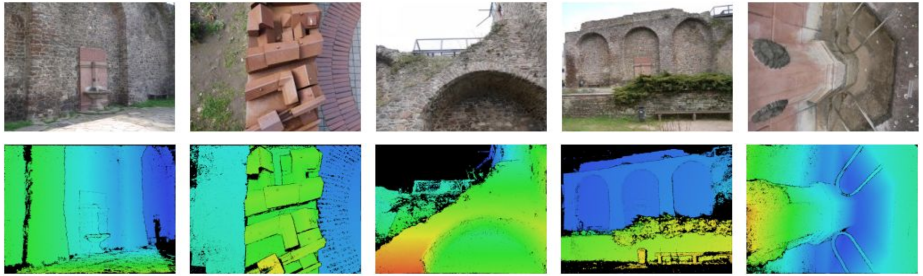


Маршировка кубов



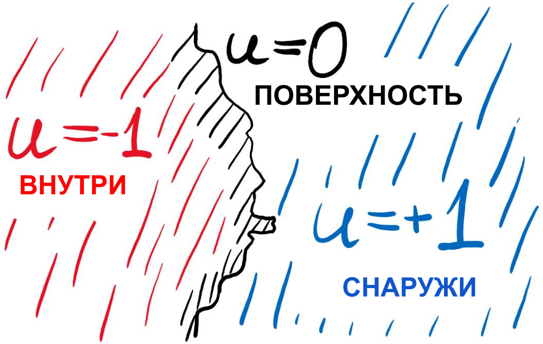
А сколько всего случаев?

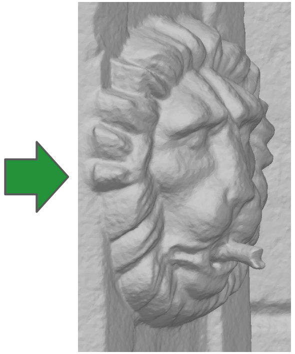
3D модель (TV-L1)



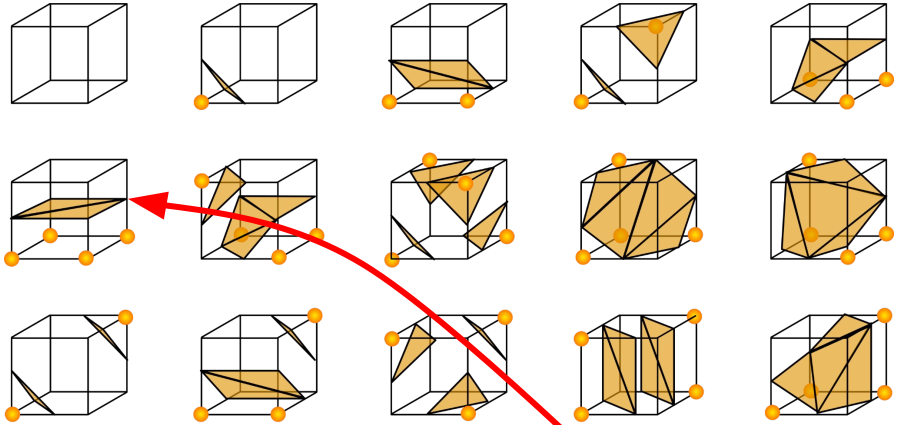
На вход: индикаторное скалярное поле

На выход: индикаторное скалярное поле





Маршировка кубов

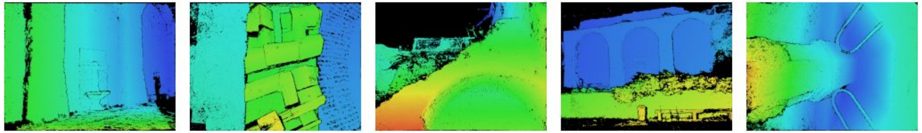


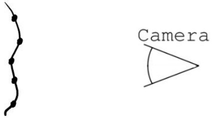
А сколько всего случаев?

А где между углами вокселя ставить вершину?

3D модель (TV-L1)

1) Есть множество карт глубины

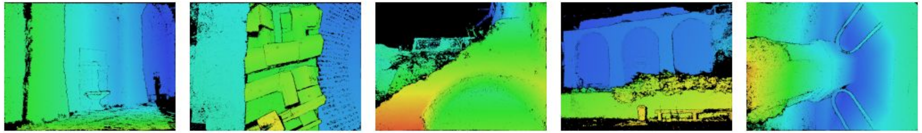


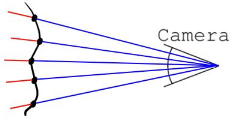


A Globally Optimal Algorithm for Robust TV-L1 Range Image Integration, **Zach et al., 2007**

3D модель (TV-L1)

1) Есть множество карт глубины

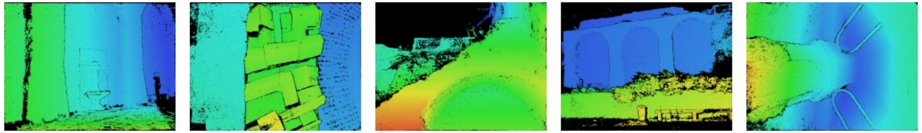


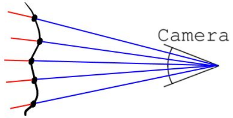


[A Globally Optimal Algorithm for Robust TV-L1 Range Image Integration, **Zach et al., 2007**]

3D модель (TV-L1)

1) Есть множество карт глубины



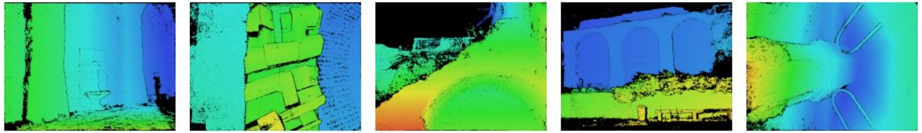


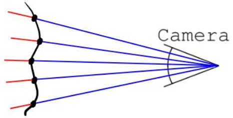
2) Множество точек всех карт глубины - порождает адаптивное октодерево (дискретизация пространства)

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

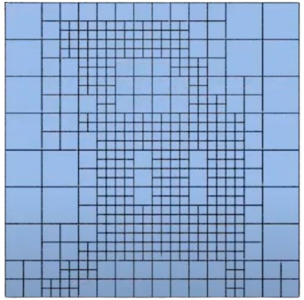
3D модель (TV-L1)

1) Есть множество карт глубины





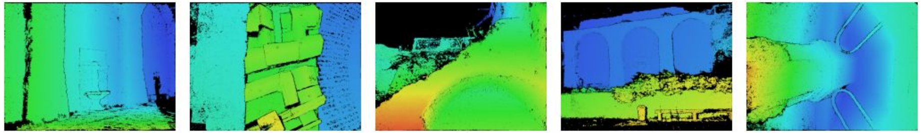
2) Множество точек всех карт глубины - порождает адаптивное октодерево (дискретизация пространства)

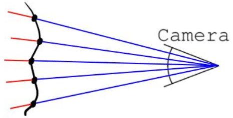


[A Globally Optimal Algorithm for Robust TV-L1 Range Image Integration, **Zach et al., 2007**]

3D модель (**TV-L1**)

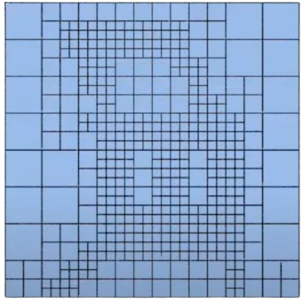
1) Есть множество карт глубины





2) Множество точек всех карт глубины - порождает **адаптивное** октодерево (дискретизация пространства)

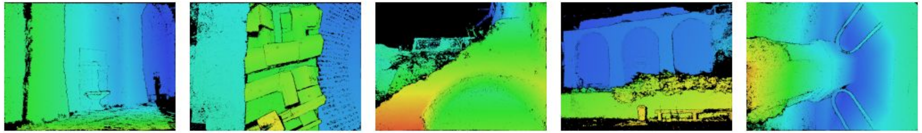
   как по точке решить какого размера нужен воксель?

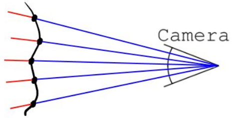


[A Globally Optimal Algorithm for Robust TV-L1 Range Image Integration, **Zach et al., 2007**]

3D модель (**TV-L1**)

1) Есть множество карт глубины

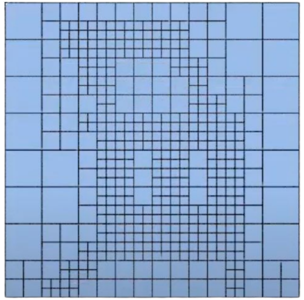




2) Множество точек всех карт глубины - порождает **адаптивное** октодерево (дискретизация пространства)

как по точке решить какого размера нужен воксель?

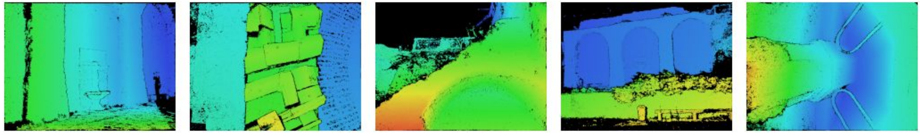
у каждой точки есть разрешение (в миллиметрах)

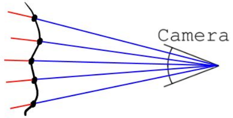


[A Globally Optimal Algorithm for Robust TV-L1 Range Image Integration, **Zach et al., 2007**]

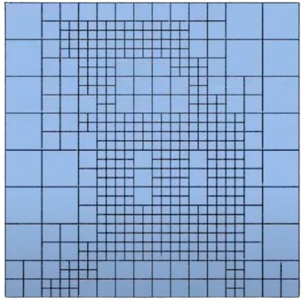
3D модель (**TV-L1**)

1) Есть множество карт глубины





2) Множество точек всех карт глубины - порождает **адаптивное** октодерево (дискретизация пространства)



3) Оптимизировали индикаторное поле (много итераций)

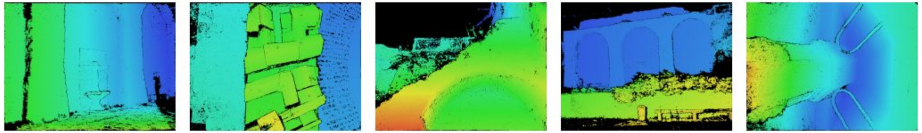
TV-L^1:

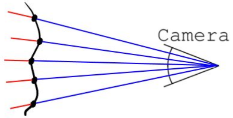
$$E = \int_{\Omega} \left\{ |\nabla u| + \lambda \sum_{i \in \mathcal{I}(\vec{x})} w_i(\vec{x}) |u - f_i| \right\} d\vec{x}$$

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

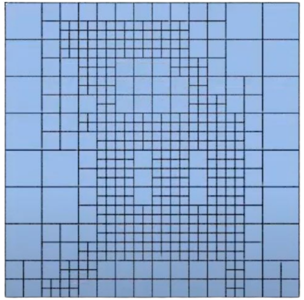
3D модель (**TV-L1**)

1) Есть множество карт глубины





2) Множество точек всех карт глубины - порождает **адаптивное** октодерево (дискретизация пространства)

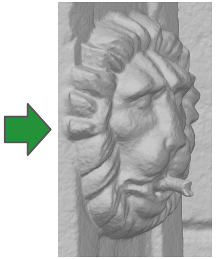


3) Оптимизировали индикаторное поле (много итераций)

TV-L^1:

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

4) Извлекли поверхность маршировкой кубов

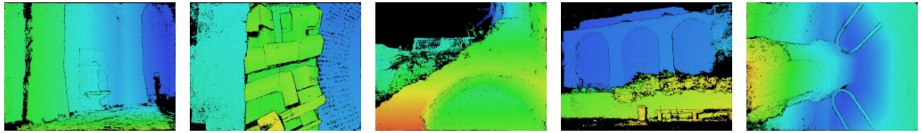


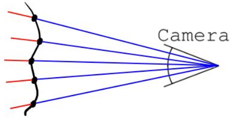
[A Globally Optimal Algorithm for Robust TV-L1 Range Image Integration, **Zach et al., 2007**]

Как ускорить?

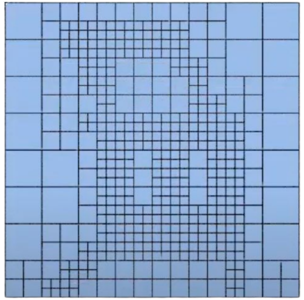
3D модель (**TV-L1**)

1) Есть множество карт глубины





2) Множество точек всех карт глубины - порождает **адаптивное** октодерево (дискретизация пространства)

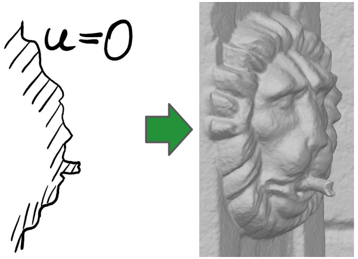


3) Оптимизировали индикаторное поле (**много итераций**)

TV-L^1:

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

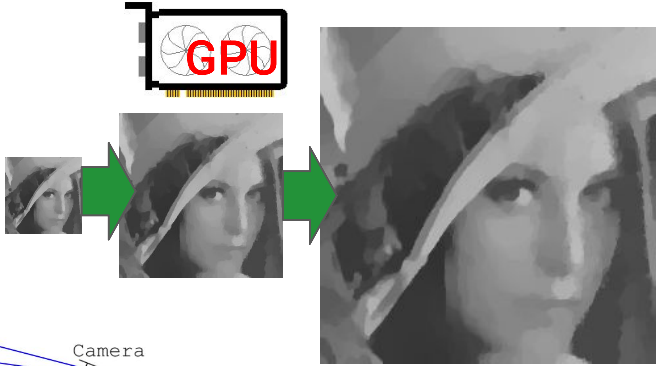
4) Извлекли поверхность маршировкой кубов



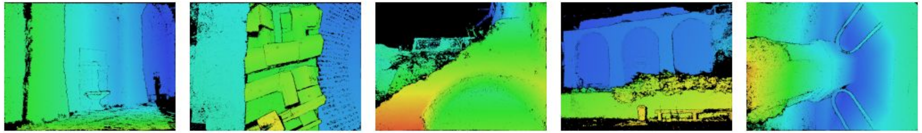
[A Globally Optimal Algorithm for Robust TV-L1 Range Image Integration, **Zach et al., 2007**]

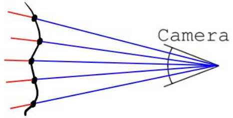
3D модель (**TV-L1**)

Как ускорить?

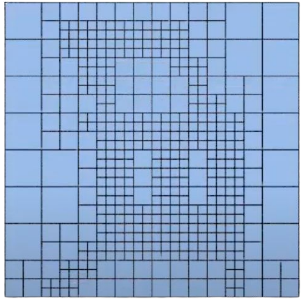


1) Есть множество карт глубины





2) Множество точек всех карт глубины - порождает **адаптивное** октодерево (дискретизация пространства)

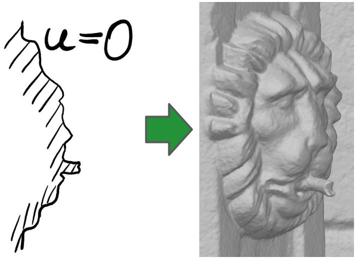


3) Оптимизировали индикаторное поле (**многo итераций**)

TV-L^1:

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

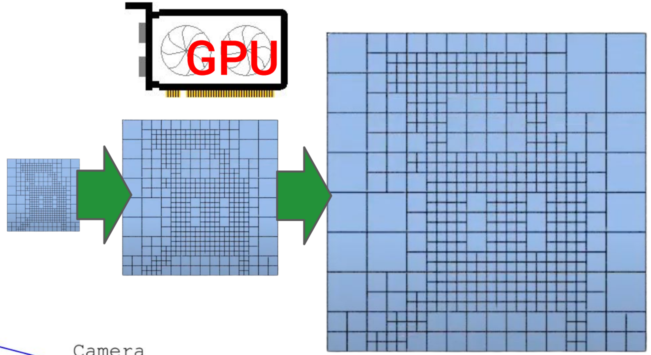
4) Извлекли поверхность маршировкой кубов



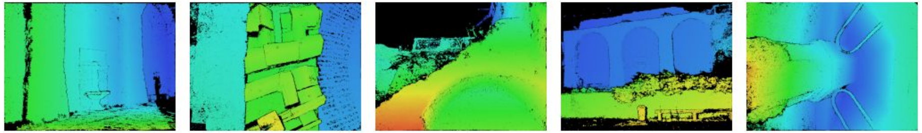
A Globally Optimal Algorithm for Robust TV-L1 Range Image Integration, **Zach et al., 2007**

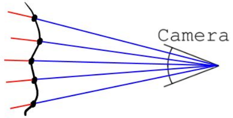
3D модель (**TV-L1**)

Ускоряем: GPU + Coarse-to-Fine

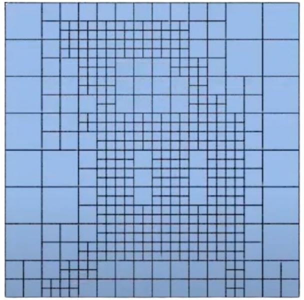


1) Есть множество карт глубины





2) Множество точек всех карт глубины - порождает **адаптивное** октодерево (дискретизация пространства)

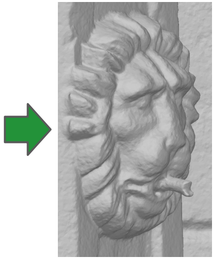


3) Оптимизировали индикаторное поле (**много итераций**)

TV-L^1:

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

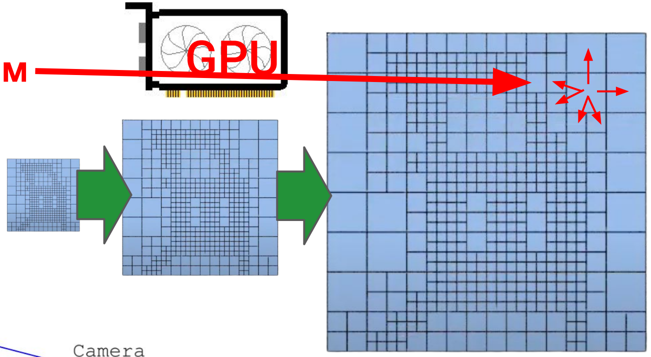
4) Извлекли поверхность маршировкой кубов



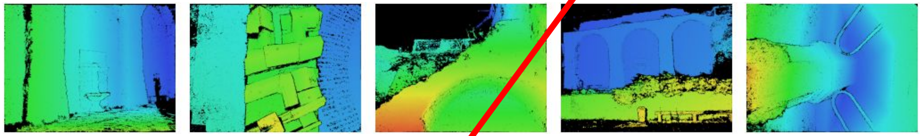
A Globally Optimal Algorithm for Robust TV-L1 Range Image Integration, **Zach et al., 2007**

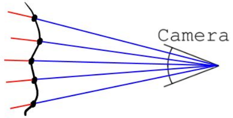
Нужно обращаться к соседним вокселям

3D модель (TV-L1)

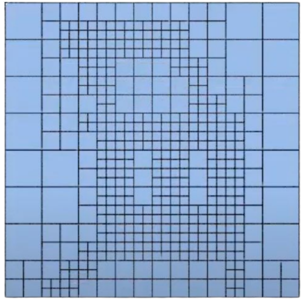


1) Есть множество карт глубины





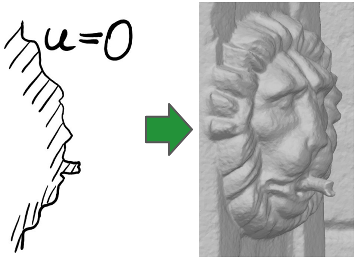
2) Множество точек всех карт глубины - порождает **адаптивное** октодерево (дискретизация пространства)



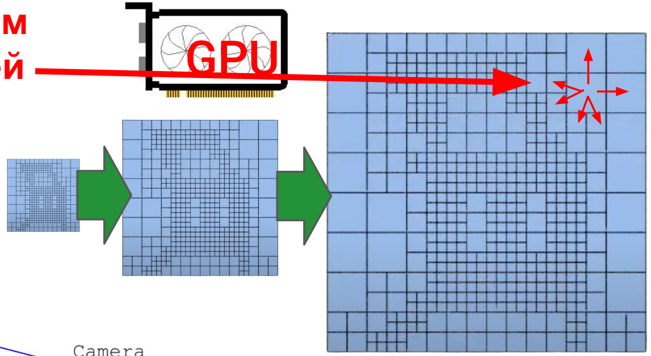
3) Оптимизировали индикаторное поле (**много итераций**)

TV-L^1:
$$E = \int_\Omega \left\{ |\nabla u| + \lambda \sum_{i \in \mathcal{I}(\vec{x})} w_i(\vec{x}) |u - f_i| \right\} d\vec{x}$$

4) Извлекли поверхность маршировкой кубов

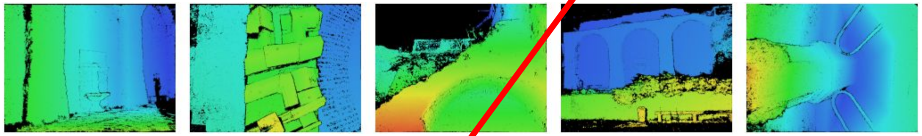


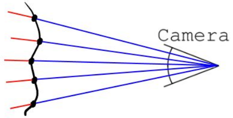
[A Globally Optimal Algorithm for Robust TV-L1 Range Image Integration, **Zach et al., 2007**]



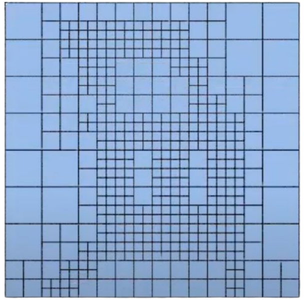
3D модель (**TV-L1**)

1) Есть множество карт глубины





2) Множество точек всех карт глубины - порождает **адаптивное** октодерево (дискретизация пространства)

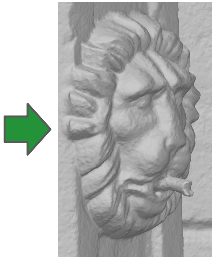


3) Оптимизировали индикаторное поле (**много итераций**)

TV-L^1:

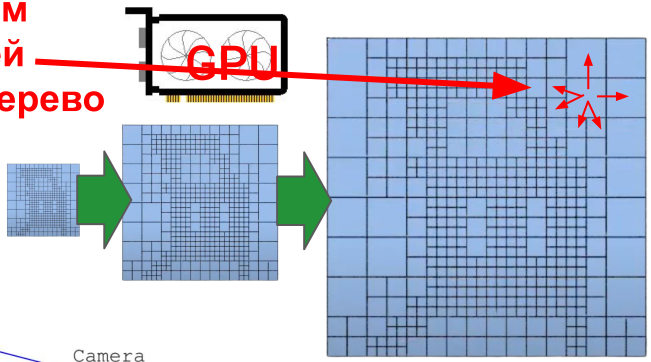
$$E = \int_\Omega \left\{ |\nabla u| + \lambda \sum_{i \in \mathcal{I}(\vec{x})} w_i(\vec{x}) |u - f_i| \right\} d\vec{x}$$

4) Извлекли поверхность маршировкой кубов



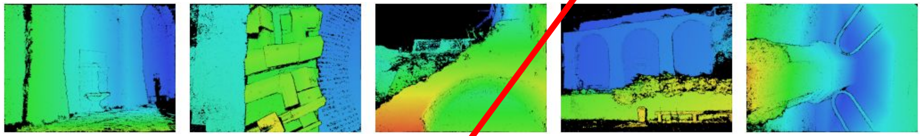
[A Globally Optimal Algorithm for Robust TV-L1 Range Image Integration, **Zach et al., 2007**]

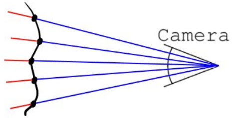
Нужно обращаться к соседним вокселям
Значит нужно ограничить число соседей
Значит нужно сбалансированное октодерево



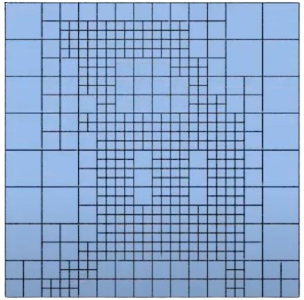
3D модель (**TV-L1**)

1) Есть множество карт глубины





2) Множество точек всех карт глубины - порождает **адаптивное** октодерево (дискретизация пространства)

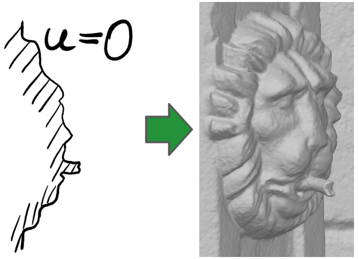


3) Оптимизировали индикаторное поле (**много итераций**)

TV-L^1:

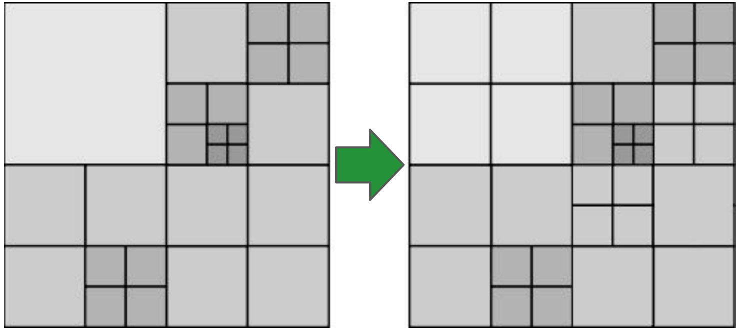
$$E = \int_\Omega \left\{ |\nabla u| + \lambda \sum_{i \in \mathcal{I}(\vec{x})} w_i(\vec{x}) |u - f_i| \right\} d\vec{x}$$

4) Извлекли поверхность маршировкой кубов

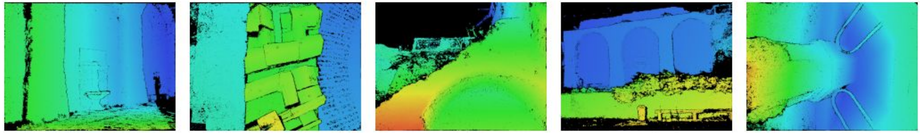


A Globally Optimal Algorithm for Robust TV-L1 Range Image Integration, **Zach et al., 2007**

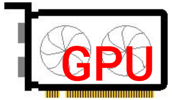
3D модель (**TV-L1**)



1) Есть множество карт глубины



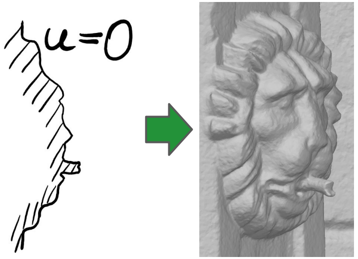
2) Множество точек всех карт глубины - порождает **адаптивное** октодерево (дискретизация пространства)
3) Балансируем октодерево 2:1 (по каждой стороне макс. два соседа)
4) Оптимизировали индикаторное поле (**много итераций + Coarse-to-Fine**)



TV-L^1:

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

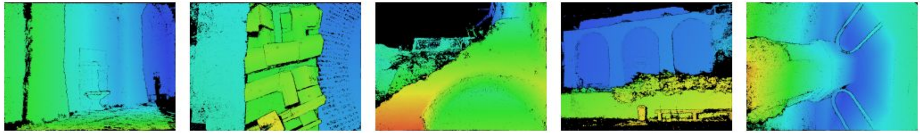
5) Извлекли поверхность маршировкой кубов

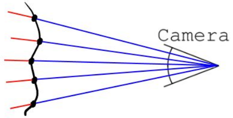


A Globally Optimal Algorithm for Robust TV-L1 Range Image Integration, **Zach et al., 2007**

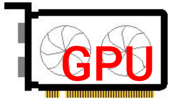
3D модель (**TV-L1**)

1) Есть множество карт глубины





2) Множество точек всех карт глубины - порождает **адаптивное** октодерево (дискретизация пространства)
3) Балансируем октодерево 2:1 (по каждой стороне макс. два соседа)
4) Оптимизировали индикаторное поле (**много итераций + Coarse-to-Fine**)



TV-L^1:

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

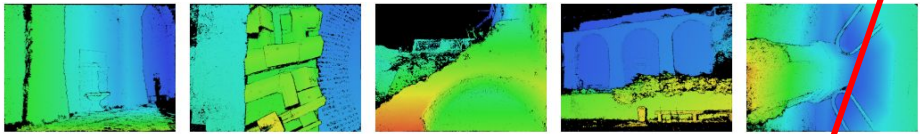
5) Извлекли поверхность маршировкой кубов

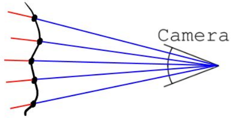
A Globally Optimal Algorithm for Robust TV-L1 Range Image Integration, **Zach et al., 2007**

Как компактно хранить карты глубины?
При этом чтобы быстро с ними сверяться!

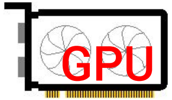
3D модель (**TV-L1**)

1) Есть множество карт глубины





2) Множество точек всех карт глубины - порождает **адаптивное** октодерево (дискретизация пространства)
3) Балансируем октодерево 2:1 (по каждой стороне макс. два соседа)
4) Оптимизировали индикаторное поле (**много итераций + Coarse-to-Fine**)



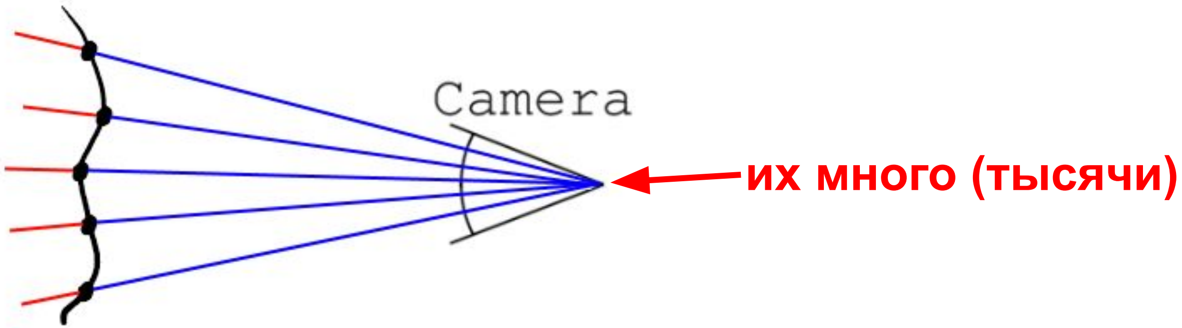
TV-L^1:

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

5) Извлекли поверхность маршировкой кубов

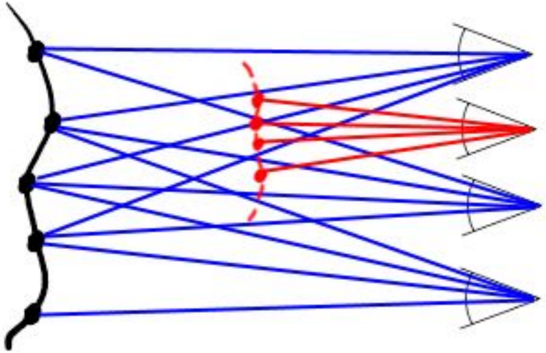
[A Globally Optimal Algorithm for Robust TV-L1 Range Image Integration, **Zach et al., 2007**]

3D модель (**TV-L1**): как компактно хранить f_i карт глубины?



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

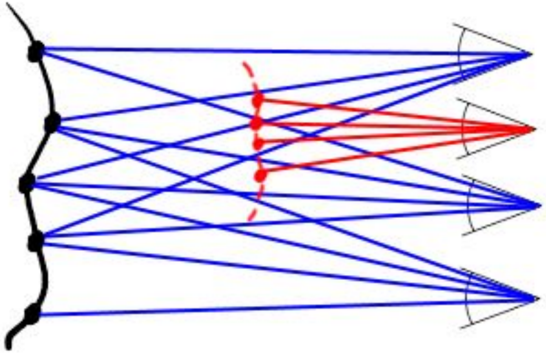
3D модель (**TV-L1**): как компактно хранить f_i карт глубины?



их много (тысячи)

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

3D модель (**TV-L1**): как компактно хранить f_i карт глубины?



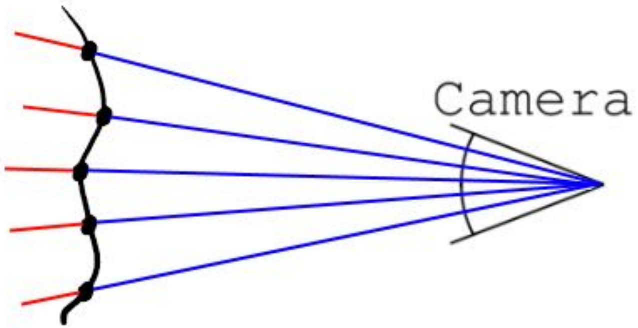
их много (тысячи)

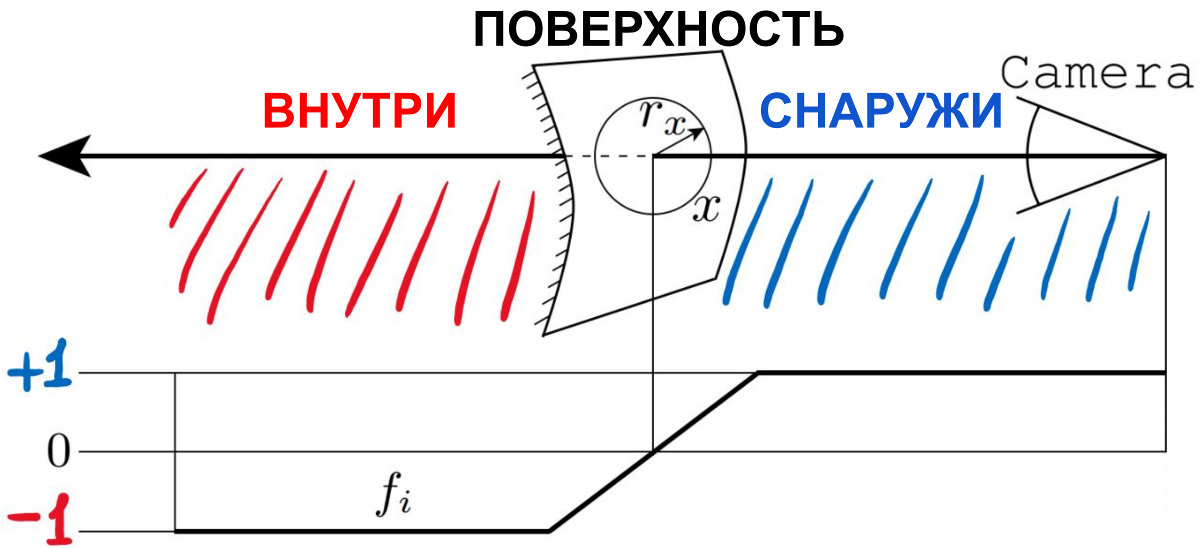
если пространство дискретизировано октодеревом из N вокселей

сколько бы памяти потребовалось чтобы хранить скалярные индикаторные поля?

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

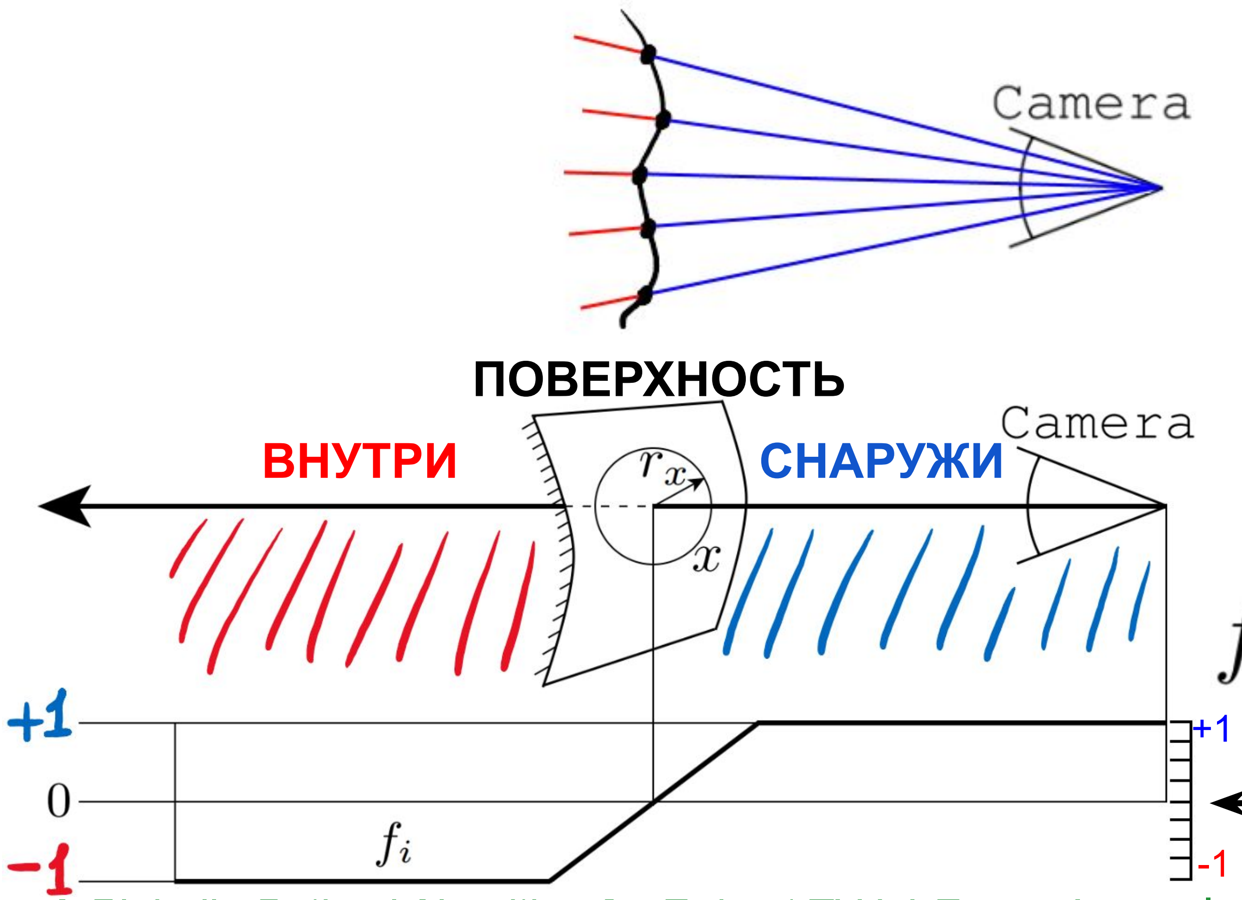
3D модель (**TV-L1**): как компактно хранить f_i карт глубины?





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

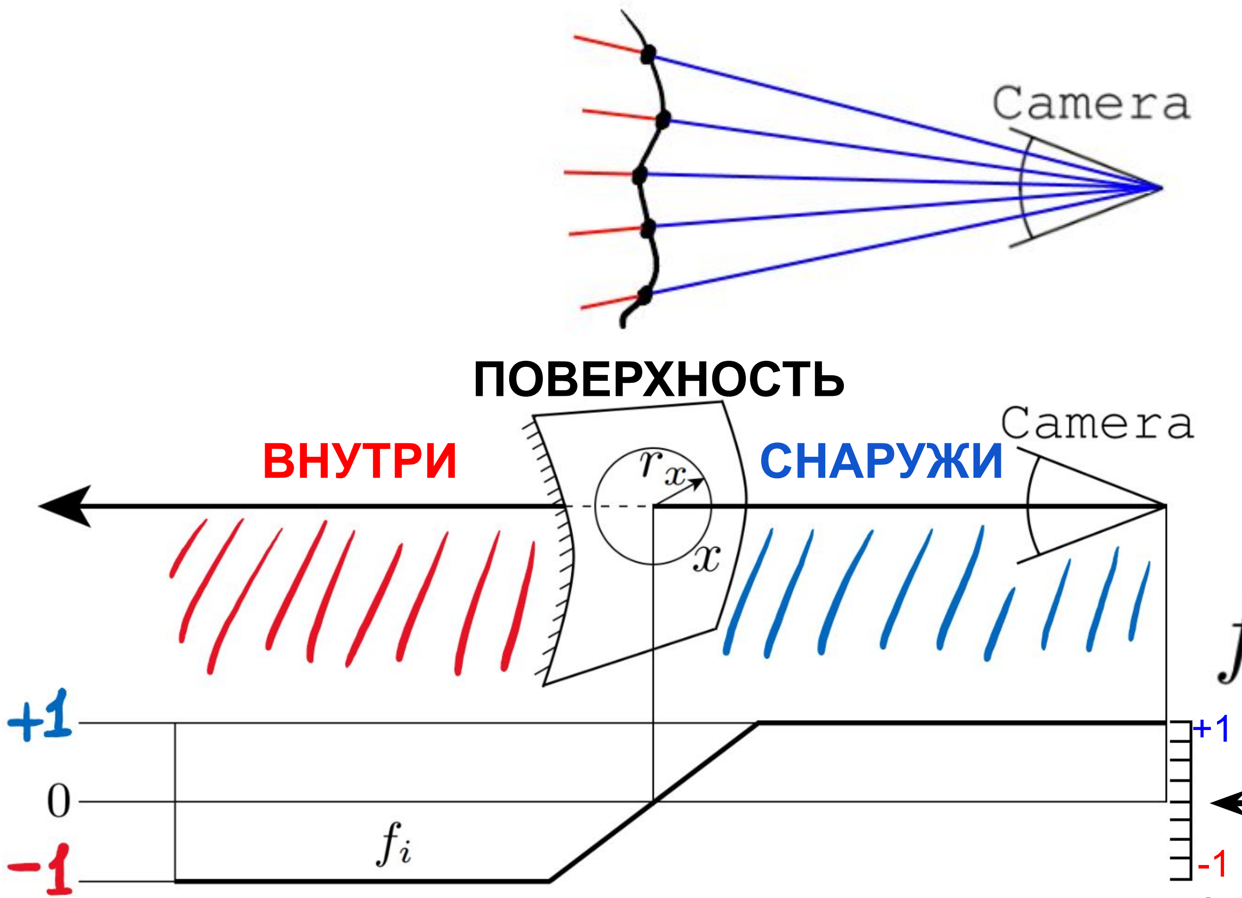
3D модель (**TV-L1**): как компактно хранить f_i карт глубины?

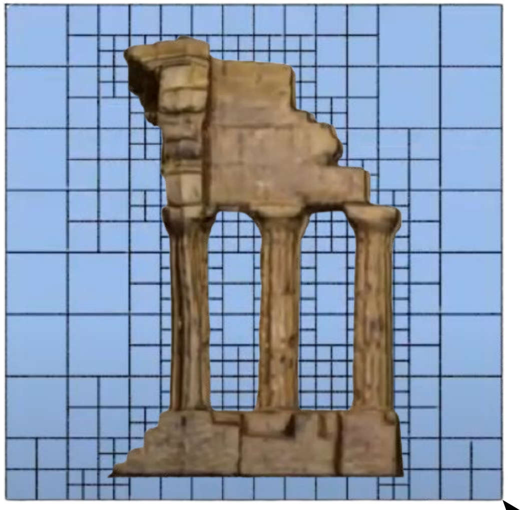


f_i — индикаторное поле соответствующее карте глубины i, хранится в вокселях гистограммами

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

3D модель (**TV-L1**): как компактно хранить f_i карт глубины?

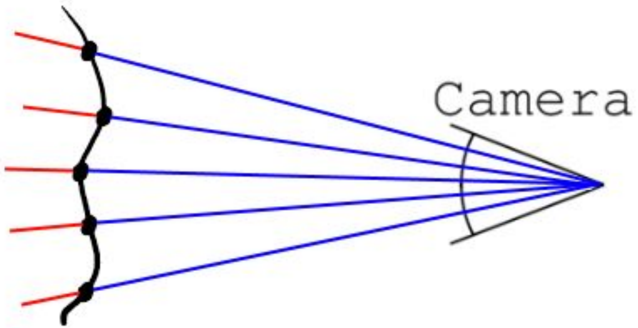


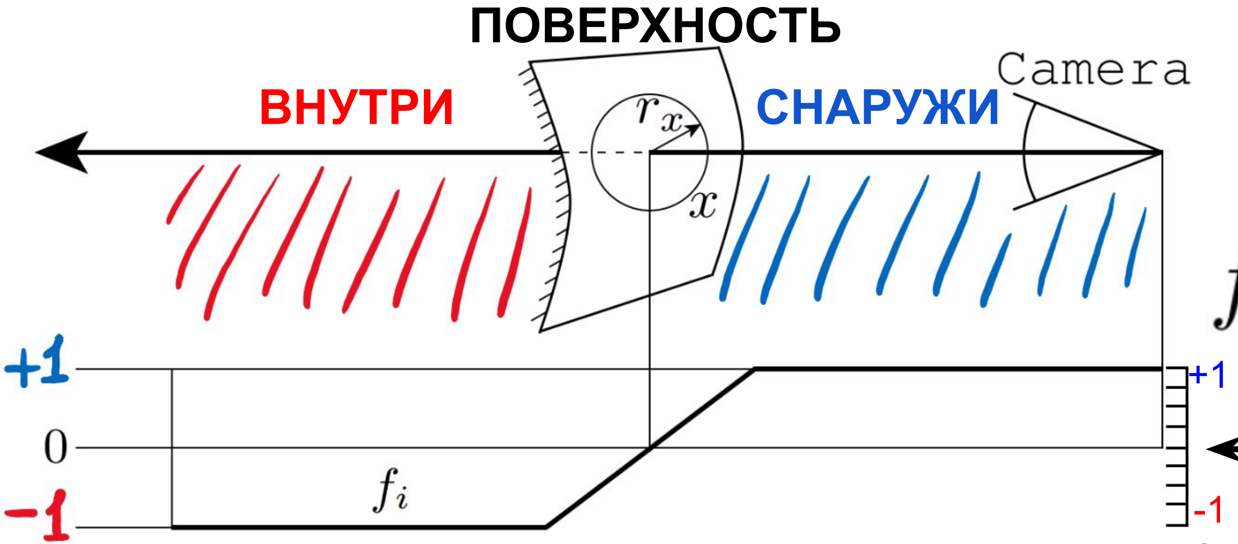


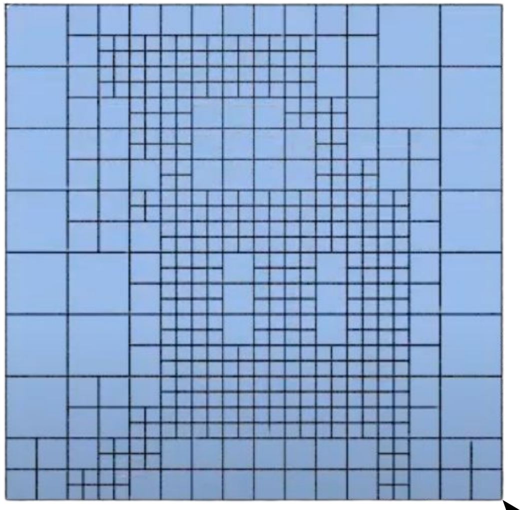
f_i — индикаторное поле соответствующее карте глубины i, хранится в вокселях гистограммами

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

3D модель (**TV-L1**): как компактно хранить f_i карт глубины?





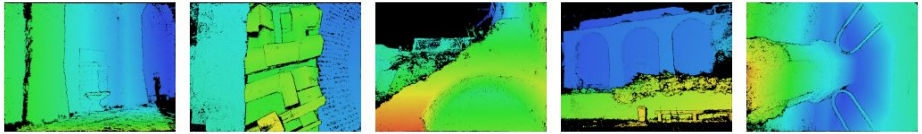


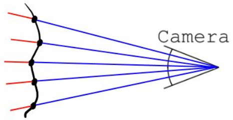
f_i — индикаторное поле соответствующее карте глубины i, хранится в вокселях гистограммами

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

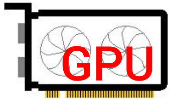
3D модель (**TV-L1**)

1) Есть множество карт глубины

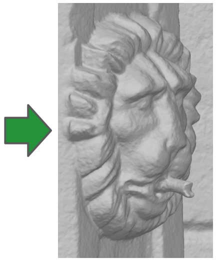




2) Множество точек всех карт глубины - порождает **адаптивное** октодерево (дискретизация пространства)
3) Балансируем октодерево 2:1 (по каждой стороне макс. два соседа)
4) Каждая карта глубины вносит свои индикаторные f_i голоса за воксели
5) Оптимизировали индикаторное поле (**много итераций + Coarse-to-Fine**)



$$\text{TV-}L^1: \quad E = \int_\Omega \left\{ |\nabla u| + \lambda \sum_{i \in \mathcal{I}(\vec{x})} w_i(\vec{x}) |u - f_i| \right\} d\vec{x}$$



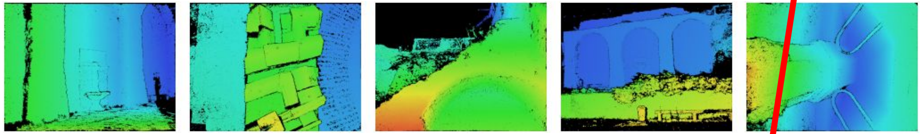
6) Извлекли поверхность маршировкой кубов

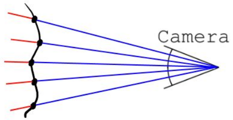
A Globally Optimal Algorithm for Robust TV-L1 Range Image Integration, **Zach et al., 2007**

Как ускорить?

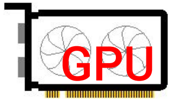
3D модель (**TV-L1**)

1) Есть множество карт глубины



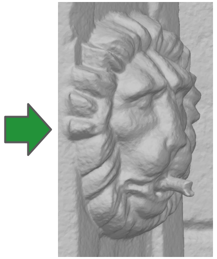


2) Множество точек всех карт глубины - порождает **адаптивное** октодерево (дискретизация пространства)
3) Балансируем октодерево 2:1 (по каждой стороне макс. два соседа)
4) Каждая карта глубины вносит свои индикаторные f_i голоса за воксели
5) Оптимизировали индикаторное поле (**много итераций + Coarse-to-Fine**)



TV-L^1:

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

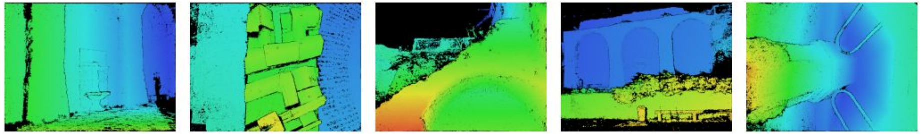


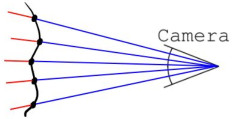
6) Извлекли поверхность маршировкой кубов

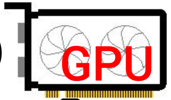
A Globally Optimal Algorithm for Robust TV-L1 Range Image Integration, **Zach et al., 2007**

3D модель (**TV-L1**)

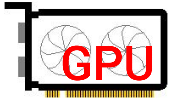
1) Есть множество карт глубины





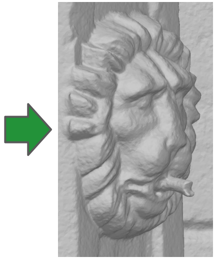
2) Множество точек всех карт глубины - порождает **адаптивное** октодерево (дискретизация пространства)
3) Балансируем октодерево 2:1 (по каждой стороне макс. два соседа)
4) 

Каждая карта глубины вносит индикаторные f_i голоса за воксели
5) Оптимизировали индикаторное поле (**много итераций + Coarse-to-Fine**)



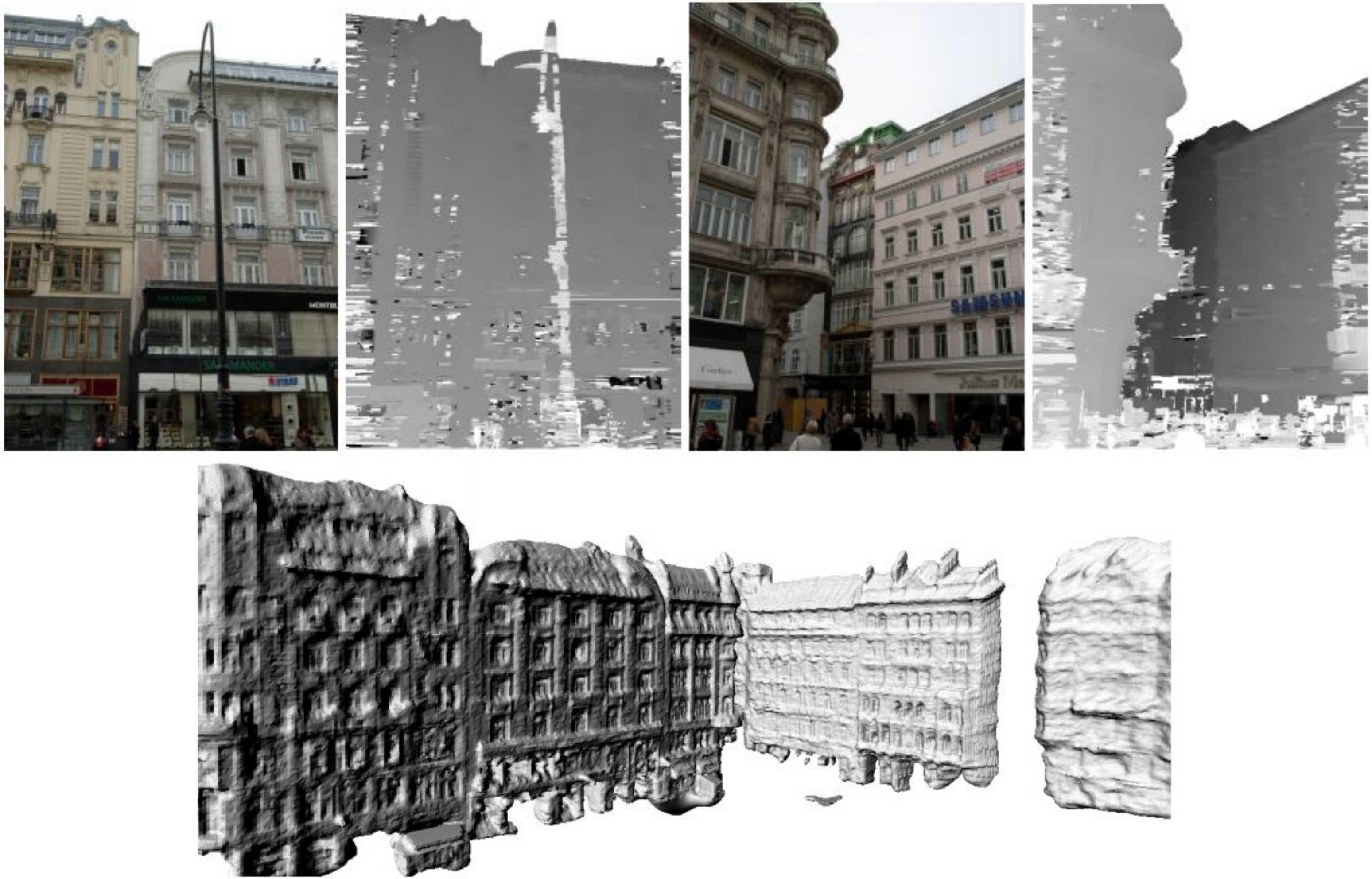
TV-L^1:

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



6) Извлекли поверхность маршировкой кубов

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 модель - **[Ummenhofer 2017]**

1) Адаптивное октодерево

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

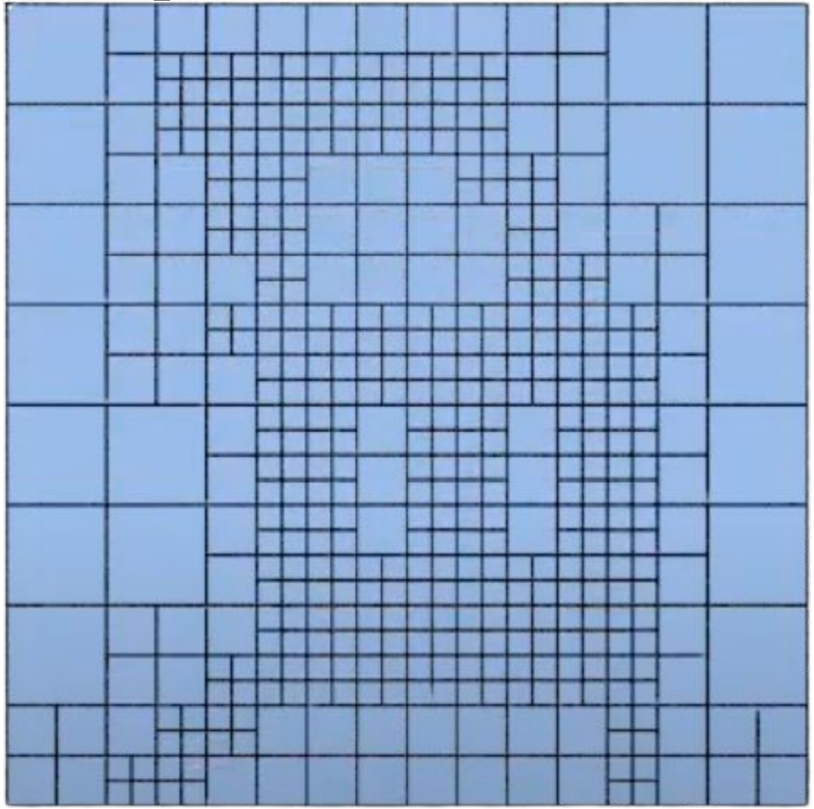
3D модель - **[Ummenhofer 2017]**

1) Адаптивное октодерево

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

3D модель - [Ummenhofer 2017]

1) Адаптивное октодерево

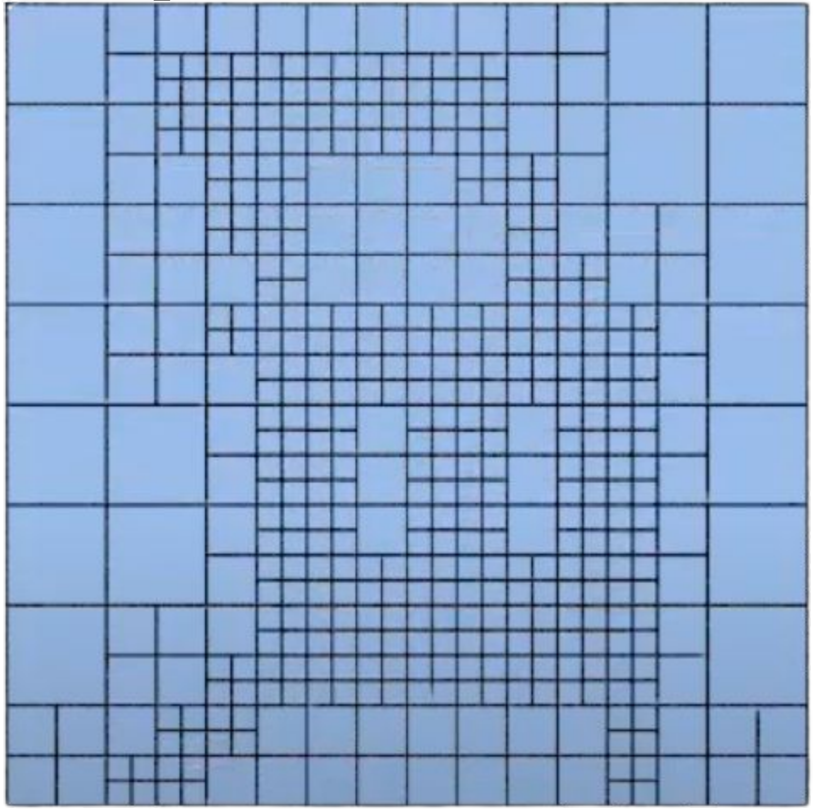


Global, Dense Multiscale Reconstruction for a Billion Points, **Ummenhofer et. al., 2017**

3D модель - [Ummenhofer 2017]

1) Адаптивное октодерево

Важно ли кроме экономии памяти?
Влияет ли на качество?



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

3D модель - [Ummenhofer 2017]

1) Адаптивное октодерево

Важно ли кроме экономии памяти?
Влияет ли на качество?

Позволяет честно обработать плотности:

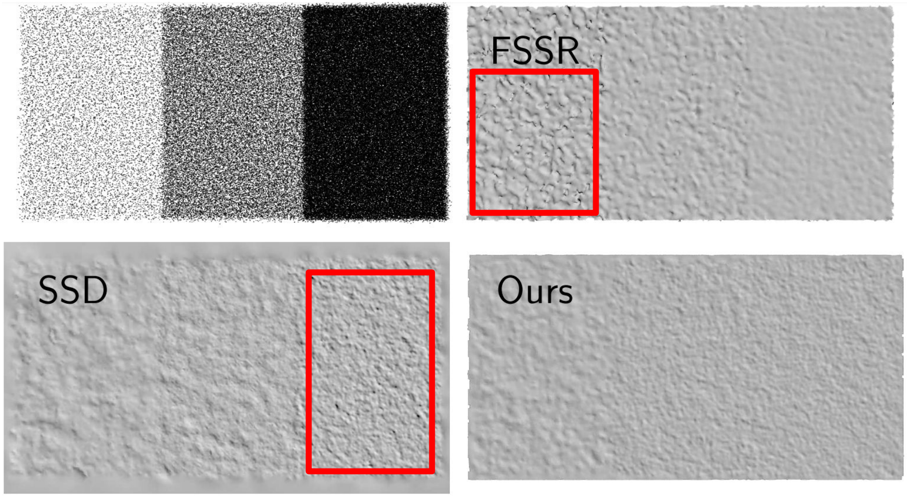
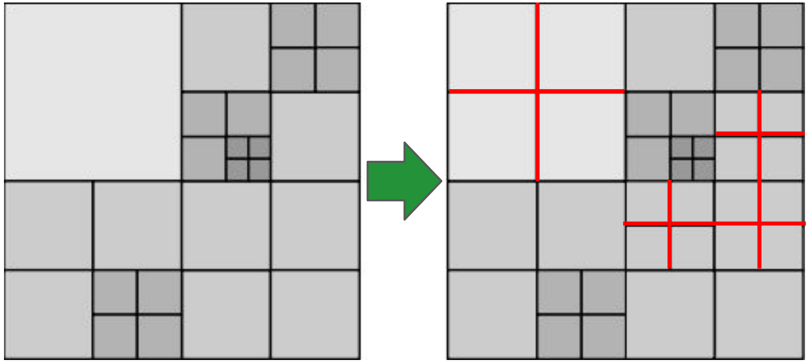


Figure 11. Reconstruction of a plane with three regions, each with a different uniform point density. The scale value assigned to the points corresponds to the sampling density of the central region. Gaussian white noise was added to the points' position and normal. **Top left:** Input point cloud. **FSSR:** 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 модель - **[Ummenhofer 2017]**

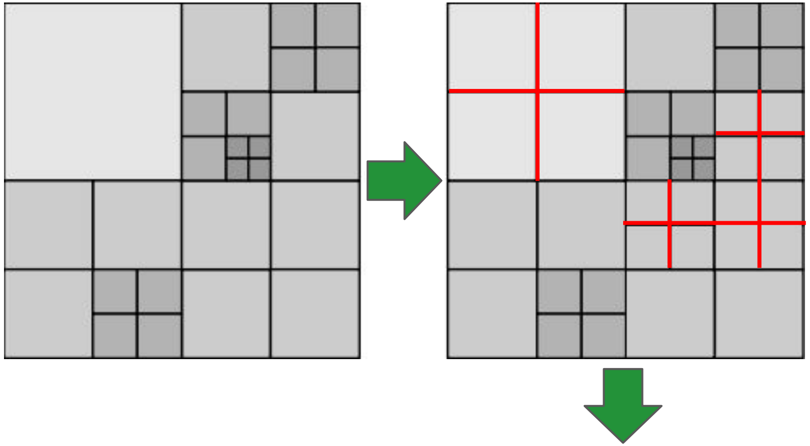
1) Адаптивное октодерево
2) 2:1 балансировка октодерева



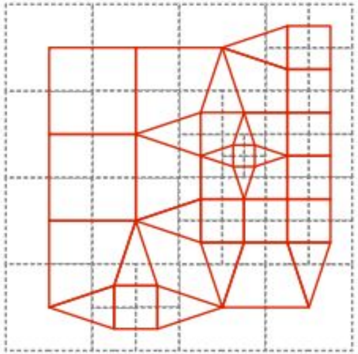
[Global, Dense Multiscale Reconstruction for a Billion Points, **Ummenhofer et. al., 2017**]

3D модель - **[Ummenhofer 2017]**

1) Адаптивное октодерево
2) 2:1 балансировка октодерева



Число соседей ограничено
Поэтому компактные воксели



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

3D модель - [Ummenhofer 2017]

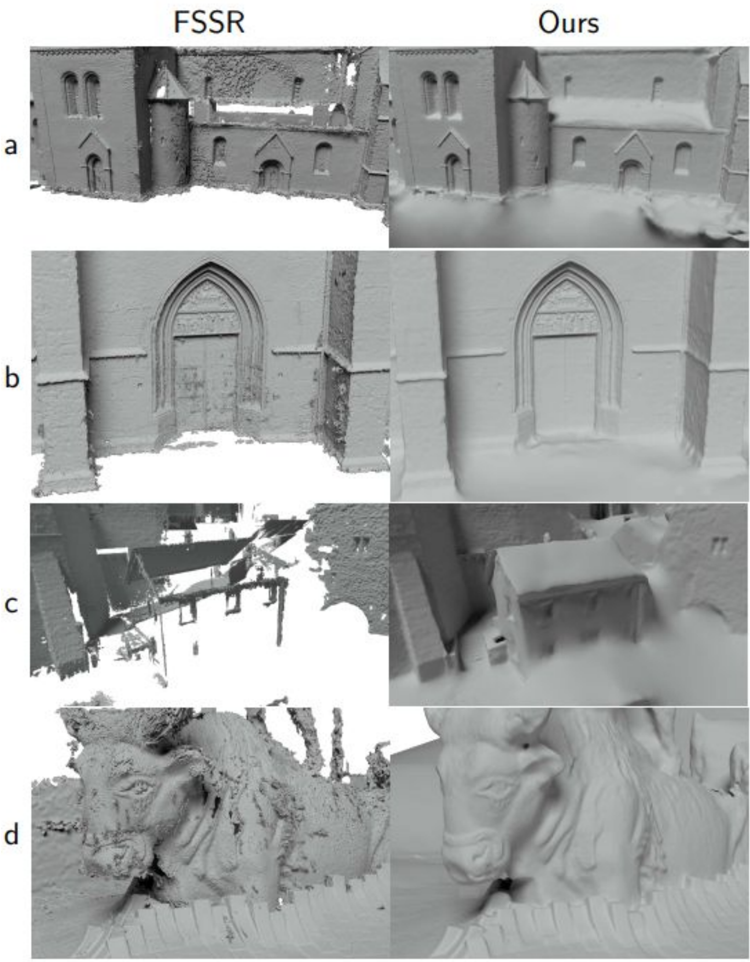
		Duration
Octree generation	Density estimation	74.6 min
	Balancing	7.9 min
	Histograms	782.4 min
Surface comp.	Dual grid generation	19.9 min
	Energy minimization	3678.0 min
	Dual contouring	16.3 min
Other		23.5 min
Total		4602.9 min

17%

80%

Table 1. Runtime breakdown for the Breisach data set.

1 billion points!



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

3D модель - [Ummenhofer 2017]

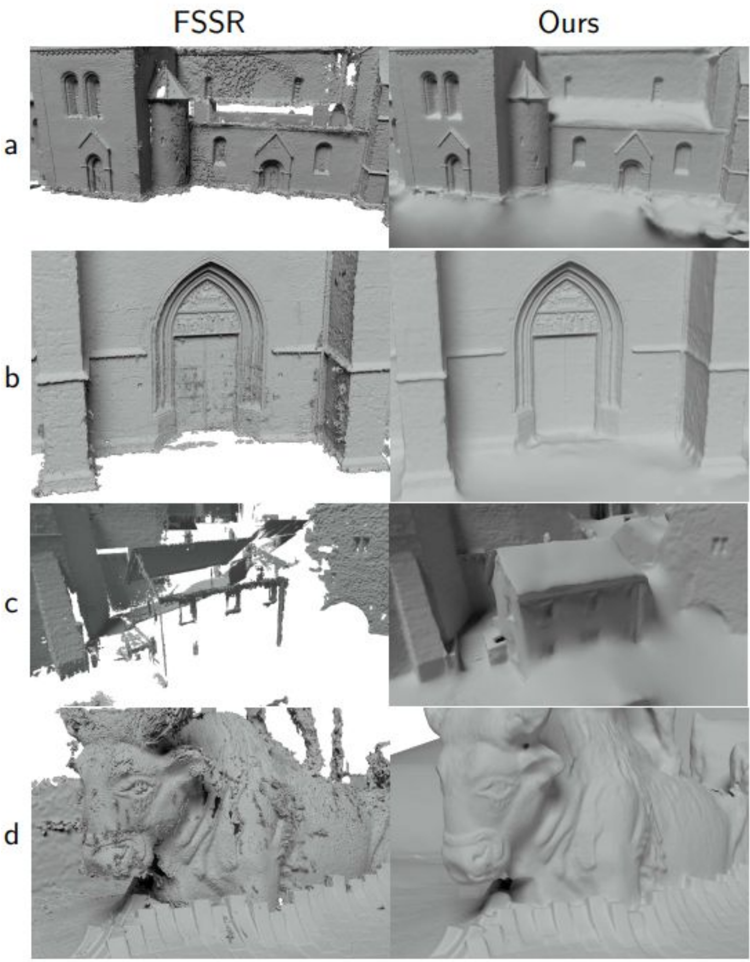
		Duration
Octree generation	Density estimation	74.6 min
	Balancing	7.9 min
	GPU Histograms	782.4 min
Surface co	Dual grid generation	19.9 min
	GPU Energy minimization	3678.0 min
	Dual contouring	16.3 min
Other		23.5 min
Total		4602.9 min

17%

80%

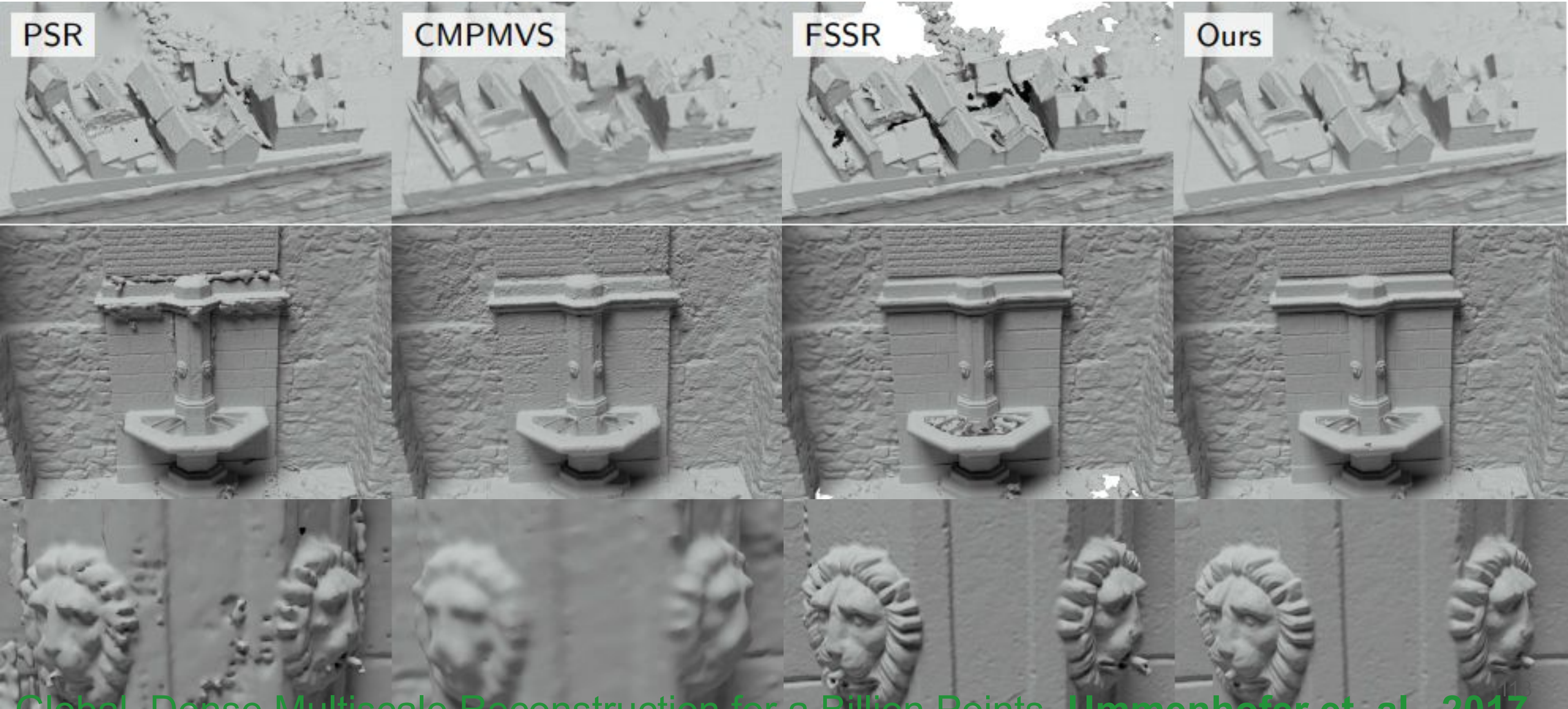
Table 1. Runtime breakdown for the Breisach data set.

1 billion points!



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

3D модель - **[Ummenhofer 2017]**



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

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

Вопросы?

Полярный Николай
polarnick239@gmail.com