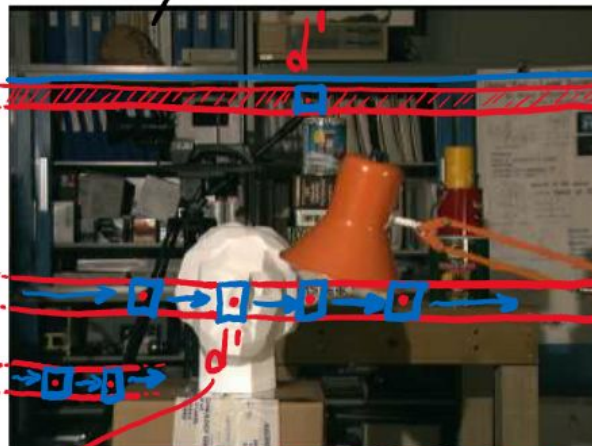
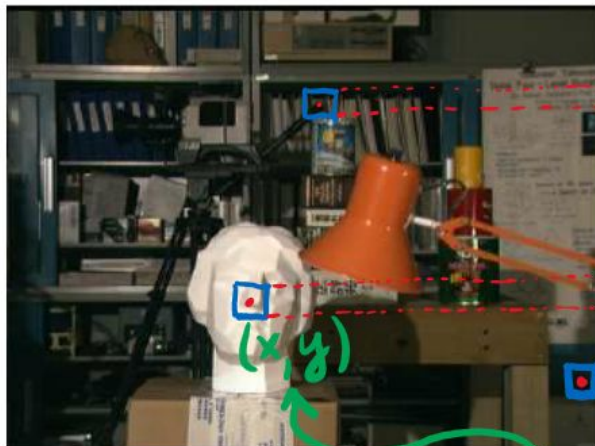




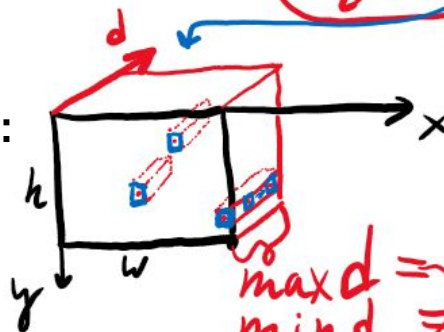
левая

правая



$$d(x, y) = d' = \underset{d}{\operatorname{argmin}} \operatorname{cost}(x, y, d)$$

DSI (disparity space image) volume:



$\max_d$   
 $\min_d$

$\operatorname{cost} = \text{😊}$

какая  
одна из 3D  
каждых точек

Как добиться субпиксельной точности?

Как добиться субпиксельной точности?

- При ректификации делать небольшое увеличение картинки
- Parabola fitting

Как добиться субпиксельной точности?

- При ректификации делать небольшое увеличение картинки
- Parabola fitting

Как пофильтровать ошибки (выбросы) и заслоненности (occlusions)?

Как добиться субпиксельной точности?

- При ректификации делать небольшое увеличение картинки
- Parabola fitting

Как пофильтровать ошибки (выбросы) и заслоненности (occlusions)?

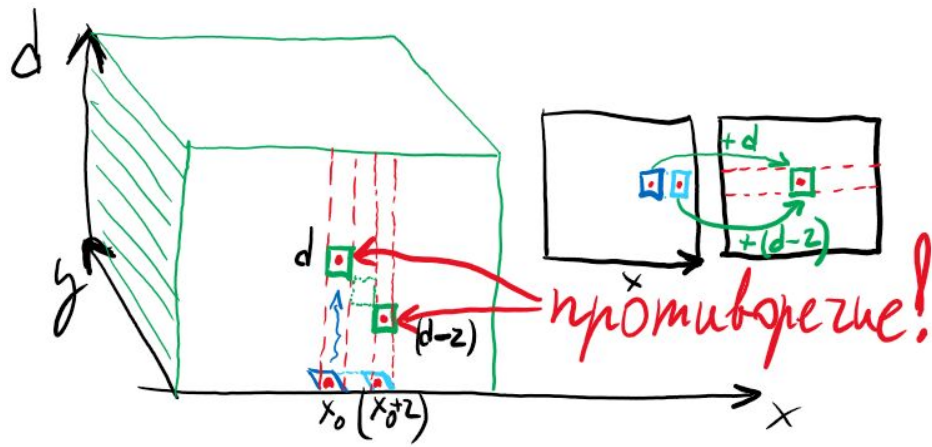
- **Left-right check:** построить обе карты глубины и сверять что переходы симметричны (сумма диспаратета меньше пикселя)

Как добиться субпиксельной точности?

- При ректификации делать небольшое увеличение картинки
- Parabola fitting

Как пофилтровать ошибки (выбросы) и заслоненности (occlusions)?

- **Left-right check:** построить обе карты глубины и сверять что переходы симметричны (сумма диспаритета меньше пикселя)
- Проверяем **one-to-one** mapping через диагональ в DSI volume:

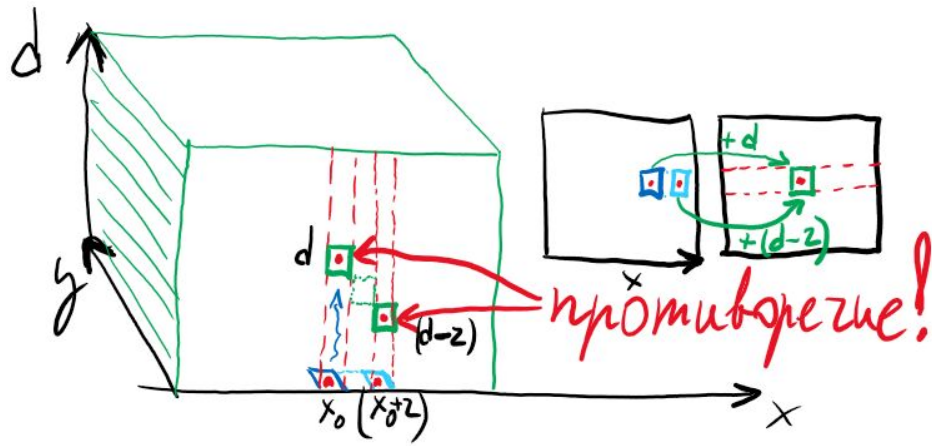


Как добиться субпиксельной точности?

- При ректификации делать небольшое увеличение картинки
- Parabola fitting

Как пофилтровать ошибки (выбросы) и заслоненности (occlusions)?

- **Left-right check**: построить обе карты глубины и сверять что переходы симметричны (сумма диспаритета меньше пикселя)
- Проверяем **one-to-one** mapping через диагональ в DSI volume:



А зачем это нужно когда  
есть Left-right check?



Как добиться субпиксельной точности?

- При ректификации делать небольшое увеличение картинки
- Parabola fitting

Как профильтровать ошибки (выбросы) и заслоненности (occlusions)?

- **Left-right check**: построить обе карты глубины и сверять что переходы симметричны (сумма диспаратета меньше пикселя)
- Проверяем **one-to-one** mapping через диагональ в DSI volume:

Как справляться с неоднозначностями - повторяющимися, регулярными и слабо текстурированными поверхностями?

Как добиться субпиксельной точности?

- При ректификации делать небольшое увеличение картинки
- Parabola fitting

Как пофильтровать ошибки (выбросы) и заслоненности (occlusions)?

- **Left-right check**: построить обе карты глубины и сверять что переходы симметричны (сумма диспаритета меньше пикселя)
- Проверяем **one-to-one** mapping через диагональ в DSI volume:

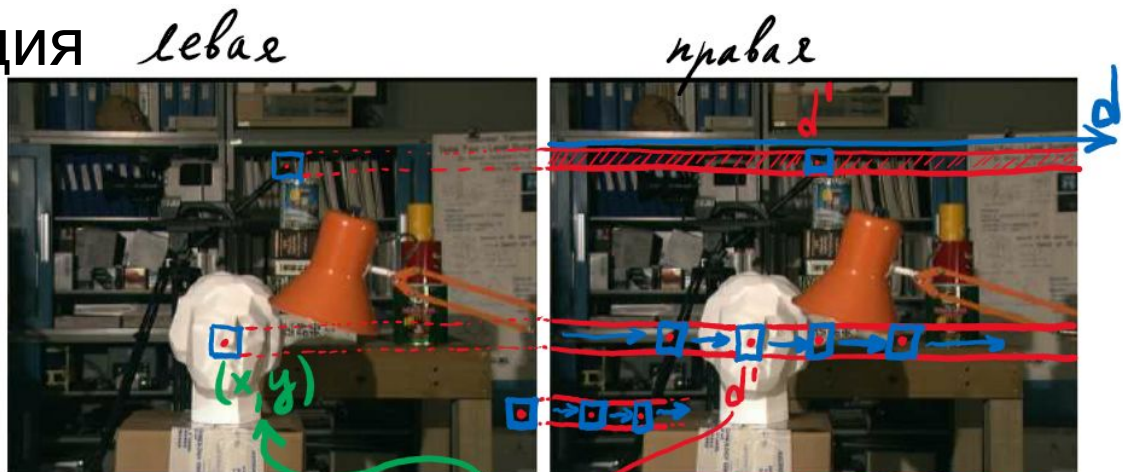
Как справляться с неоднозначностями - повторяющимися, регулярными и слабо текстурированными поверхностями?

- Решать задачу глобальной оптимизации
- Предпочитать “связные” диспаритеты (хотим гладкость)
- Штрафовать за отличающиеся диспаритеты (хотим мало разрывов)

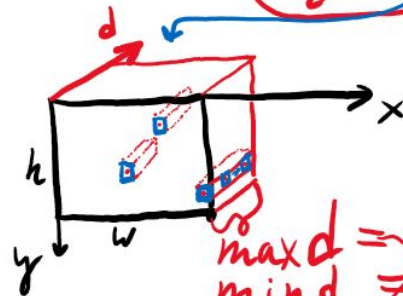
# Глобальная оптимизация *левая*

Минимизируем энергию:

$$E(d) = ???$$



$$d(x, y) = d' = \underset{d}{\operatorname{argmin}} \operatorname{cost}(x, y, d)$$



$\max_d$   
 $\min_d$

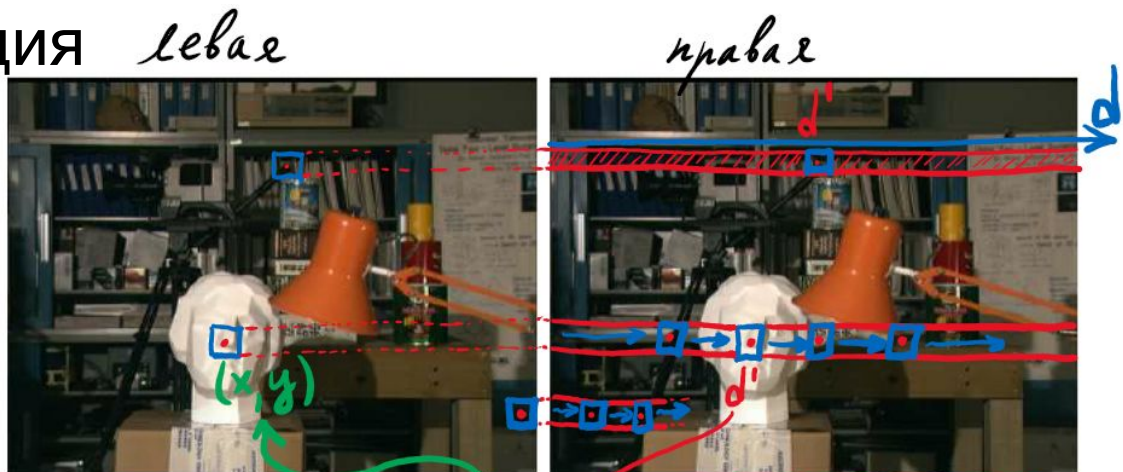
$\operatorname{cost} = \text{😊}$

на базе  
обычных 3D  
ключевых точек

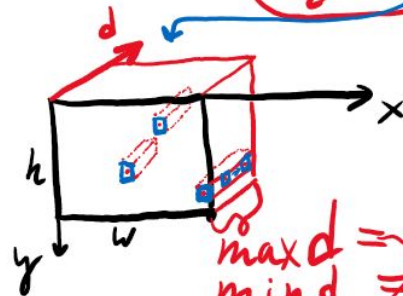
# Глобальная оптимизация *левая*

Минимизируем энергию:

$$E(d) = E_D(d) + \lambda E_S(d)$$



$$d(x, y) = d' = \underset{d}{\operatorname{argmin}} \operatorname{cost}(x, y, d)$$



cost = 😊

на базе  
одних 3D  
ключевых точек

# Глобальная оптимизация

Минимизируем энергию:  $E(d) = E_D(d) + \lambda E_S(d)$

Где **data term**: 
$$E_D(d) = \sum_{(x,y)} C(x, y, d(x, y))$$

# Глобальная оптимизация

Минимизируем энергию:  $E(d) = E_D(d) + \lambda E_S(d)$

Где **data term**:  $E_D(d) = \sum_{(x,y)} C(x, y, d(x, y))$

И **smoothness term**:

$$E_S(d) = \sum_{(x,y)} \rho(d(x, y) - d(x + 1, y)) + \rho(d(x, y) - d(x, y + 1))$$

где  $\rho$  - монотонная функция штрафа

# Глобальная оптимизация

Минимизируем энергию:  $E(d) = E_D(d) + \lambda E_S(d)$

Где **data term**:  $E_D(d) = \sum_{(x,y)} C(x, y, d(x, y))$

И **smoothness term**:

$$E_S(d) = \sum_{(x,y)} \rho(d(x, y) - d(x + 1, y)) + \rho(d(x, y) - d(x, y + 1))$$

где  $\rho$  - монотонная функция штрафа

Как ослабить штраф за разрыв на границе объекта?

# Глобальная оптимизация

Минимизируем энергию:  $E(d) = E_D(d) + \lambda E_S(d)$

Где **data term**:  $E_D(d) = \sum_{(x,y)} C(x, y, d(x, y))$

И **smoothness term**:

$$E_S(d) = \sum_{(x,y)} \rho(d(x, y) - d(x + 1, y)) + \rho(d(x, y) - d(x, y + 1))$$

где  $\rho$  - монотонная функция штрафа

Как ослабить штраф за разрыв на границе объекта?

Домножать с учетом перепада яркости (или силы градиента):

$$\rho_D(d(x, y) - d(x + 1, y)) \cdot \rho_I(\|I(x, y) - I(x + 1, y)\|)$$



# Глобальная оптимизация

Минимизируем энергию:  $E(d) = E_D(d) + \lambda E_S(d)$

Где **data term**:  $E_D(d) = \sum_{(x,y)} C(x, y, d(x, y))$

И **smoothness term**:

$$E_S(d) = \sum_{(x,y)} \rho(d(x, y) - d(x + 1, y)) + \rho(d(x, y) - d(x, y + 1))$$

где  $\rho$  - монотонная функция штрафа

Это **NP**-полная задача, приближенное решение можно искать с помощью разных методов, например через **Markov Random Fields (MRF)** или минимальный разрез графа.

# Глобальная оптимизация

Минимизируем энергию:  $E(d) = E_D(d) + \lambda E_S(d)$

Где **data term**:  $E_D(d) = \sum_{(x,y)} C(x, y, d(x, y))$

И **smoothness term**:

$$E_S(d) = \sum_{(x,y)} \rho(d(x, y) - d(x + 1, y)) + \rho(d(x, y) - d(x, y + 1))$$

где  $\rho$  - монотонная функция штрафа

Это **NP**-полная задача, приближенное решение можно искать с помощью разных методов, например через **Markov Random Fields (MRF)** или минимальный разрез графа. **Но это медленно!**

# Semi-Global Matching (**SGM**)

$$E(D) = \sum_{\mathbf{p}} \left[ C(\mathbf{p}, D_{\mathbf{p}}) + \sum_{\mathbf{q} \in N_{\mathbf{p}}} P_1 T[|D_{\mathbf{p}} - D_{\mathbf{q}}| = 1] + \sum_{\mathbf{q} \in N_{\mathbf{p}}} P_2 T[|D_{\mathbf{p}} - D_{\mathbf{q}}| > 1] \right]$$

Annotations:

- $\mathbf{p}$ : все пиксели
- $C(\mathbf{p}, D_{\mathbf{p}})$ : локальные costs
- $\mathbf{q} \in N_{\mathbf{p}}$ : соседние пиксели  $\mathbf{p}$
- $P_1 T[|D_{\mathbf{p}} - D_{\mathbf{q}}| = 1]$ : плата за отклонение disparity на 1
- $P_2 T[|D_{\mathbf{p}} - D_{\mathbf{q}}| > 1]$ : штраф за разрыв (отклонение  $\geq 1$ )

# Semi-Global Matching (**SGM**)

$$E(D) = \sum_{\mathbf{p}} [C(\mathbf{p}, D_{\mathbf{p}}) + \sum_{\mathbf{q} \in N_{\mathbf{p}}} P_1 T[|D_{\mathbf{p}} - D_{\mathbf{q}}| = 1] + \sum_{\mathbf{q} \in N_{\mathbf{p}}} P_2 T[|D_{\mathbf{p}} - D_{\mathbf{q}}| > 1]]$$

Annotations:

- $\mathbf{p}$ : все пиксели
- $C(\mathbf{p}, D_{\mathbf{p}})$ : локальные costs
- $\mathbf{q} \in N_{\mathbf{p}}$ : соседние пиксели
- $P_1 T[|D_{\mathbf{p}} - D_{\mathbf{q}}| = 1]$ : плата за отличие disparity на 1
- $P_2 T[|D_{\mathbf{p}} - D_{\mathbf{q}}| > 1]$ : штраф за разрыв (отличие  $\geq 1$ )

# Semi-Global Matching (**SGM**)

$$E(D) = \sum_{\mathbf{p}} \left[ C(\mathbf{p}, D_{\mathbf{p}}) + \sum_{\mathbf{q} \in N_{\mathbf{p}}} P_1 T[|D_{\mathbf{p}} - D_{\mathbf{q}}| = 1] + \sum_{\mathbf{q} \in N_{\mathbf{p}}} P_2 T[|D_{\mathbf{p}} - D_{\mathbf{q}}| > 1] \right]$$

Annotations:

- $\mathbf{p}$ : все пиксели
- $C(\mathbf{p}, D_{\mathbf{p}})$ : локальные costs
- $\mathbf{q} \in N_{\mathbf{p}}$ : соседние пиксели  $\mathbf{p}$
- $P_1 T[|D_{\mathbf{p}} - D_{\mathbf{q}}| = 1]$ : плата за отклонение disparity на 1
- $P_2 T[|D_{\mathbf{p}} - D_{\mathbf{q}}| > 1]$ : штраф за разрыв (отклонение  $\geq 1$ )

# Semi-Global Matching (**SGM**)

$$E(D) = \sum_{\mathbf{p}} \left[ C(\mathbf{p}, D_{\mathbf{p}}) + \sum_{\mathbf{q} \in N_{\mathbf{p}}} P_1 T[|D_{\mathbf{p}} - D_{\mathbf{q}}| = 1] + \sum_{\mathbf{q} \in N_{\mathbf{p}}} P_2 T[|D_{\mathbf{p}} - D_{\mathbf{q}}| > 1] \right]$$

Handwritten annotations in blue:

- Все пиксели  $\rightarrow \mathbf{p}$
- локальные costs  $\rightarrow C(\mathbf{p}, D_{\mathbf{p}})$
- соседние пиксели  $\mathbf{p} \rightarrow \mathbf{q} \in N_{\mathbf{p}}$

Handwritten annotations in red:

- штраф за отличие disparity на 1  $\rightarrow P_1 T[|D_{\mathbf{p}} - D_{\mathbf{q}}| = 1]$
- штраф за разрыв (отличие  $\geq 1$ )  $\rightarrow P_2 T[|D_{\mathbf{p}} - D_{\mathbf{q}}| > 1]$

Как соотносятся штрафы  $P_1$  и  $P_2$ ? Кто больше?

# Semi-Global Matching (**SGM**)

$$E(D) = \sum_{\mathbf{p}} \left[ \underbrace{C(\mathbf{p}, D_{\mathbf{p}})}_{\text{локальные costs}} + \sum_{\mathbf{q} \in N_{\mathbf{p}}} \underbrace{P_1 T[|D_{\mathbf{p}} - D_{\mathbf{q}}| = 1]}_{\text{штраф за отличие disparity на 1}} + \sum_{\mathbf{q} \in N_{\mathbf{p}}} \underbrace{P_2 T[|D_{\mathbf{p}} - D_{\mathbf{q}}| > 1]}_{\text{штраф за разрыв (отличие } \geq 1)} \right]$$

все пиксели  $\rightarrow \mathbf{p}$

соседние пиксели  $\mathbf{p}$

$P_2 \geq P_1$

Как добавить послабление на границах объектов?  
Как снизить там штрафы за разрывы?

# Semi-Global Matching (**SGM**)

$$E(D) = \sum_{\mathbf{p}} \left[ \underbrace{C(\mathbf{p}, D_{\mathbf{p}})}_{\text{локальные costs}} + \sum_{\mathbf{q} \in N_{\mathbf{p}}} \underbrace{P_1 T[|D_{\mathbf{p}} - D_{\mathbf{q}}| = 1]}_{\text{штраф за отличие disparity на 1}} + \sum_{\mathbf{q} \in N_{\mathbf{p}}} \underbrace{P_2 T[|D_{\mathbf{p}} - D_{\mathbf{q}}| > 1]}_{\text{штраф за разрыв (отличие } \geq 1)} \right]$$

все пиксели  $\rightarrow \mathbf{p}$   
 локальные costs  
 соседние пиксели  $\mathbf{p}$   
 штраф за отличие disparity на 1  
 штраф за разрыв (отличие  $\geq 1$ )

$$P_2 \geq P_1$$

можно задать последнее на границах

$$P_2 = \frac{P_2'}{|I_{bp} - I_{bq}|}$$



## Semi-Global Matching (**SGM**)

$$E(D) = \sum_{\mathbf{p}} C(\mathbf{p}, D_{\mathbf{p}}) + \sum_{\mathbf{q} \in N_{\mathbf{p}}} P_1 T[|D_{\mathbf{p}} - D_{\mathbf{q}}| = 1] + \sum_{\mathbf{q} \in N_{\mathbf{p}}} P_2 T[|D_{\mathbf{p}} - D_{\mathbf{q}}| > 1]$$

У пикселя 8 соседей, из-за взаимного влияния - вычислительно тяжело.

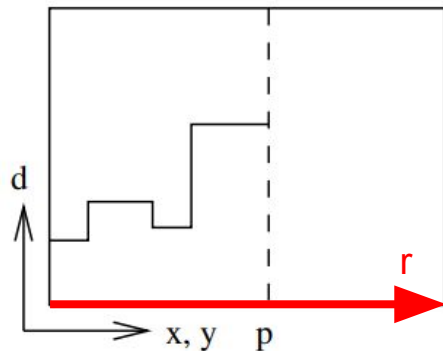
# Semi-Global Matching (**SGM**)

$$E(D) = \sum_{\mathbf{p}} C(\mathbf{p}, D_{\mathbf{p}}) + \sum_{\mathbf{q} \in N_{\mathbf{p}}} P_1 T[|D_{\mathbf{p}} - D_{\mathbf{q}}| = 1] + \sum_{\mathbf{q} \in N_{\mathbf{p}}} P_2 T[|D_{\mathbf{p}} - D_{\mathbf{q}}| > 1]$$

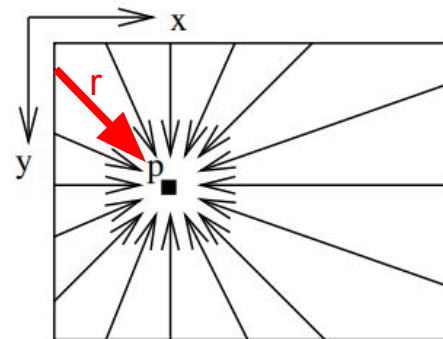
У пикселя 8 соседей, из-за взаимного влияния - вычислительно тяжело.

Давайте фиксируем одно направление и будем учитывать только соседа по этому направлению  $\mathbf{r}$ :

(a) Minimum Cost Path  $L_t(\mathbf{p}, d)$



(b) 16 Paths from all Directions  $\mathbf{r}$



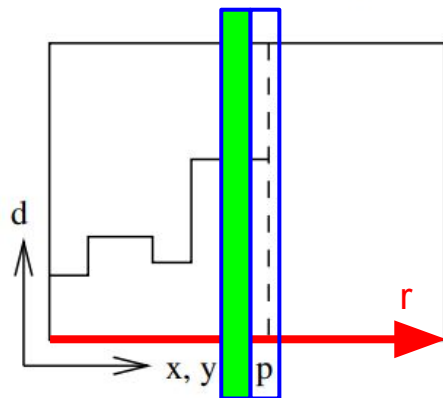
# Semi-Global Matching (**SGM**)

$$E(D) = \sum_{\mathbf{p}} C(\mathbf{p}, D_{\mathbf{p}}) + \sum_{\mathbf{q} \in N_{\mathbf{p}}} P_1 T[|D_{\mathbf{p}} - D_{\mathbf{q}}| = 1] + \sum_{\mathbf{q} \in N_{\mathbf{p}}} P_2 T[|D_{\mathbf{p}} - D_{\mathbf{q}}| > 1]$$

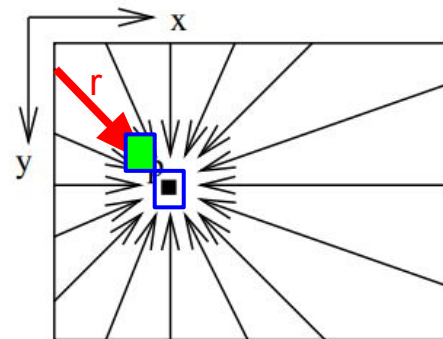
У пикселя 8 соседей, из-за взаимного влияния - вычислительно тяжело.

Давайте фиксируем одно направление и будем учитывать только соседа по этому направлению  $\mathbf{r}$ :

(a) Minimum Cost Path  $L_t(\mathbf{p}, d)$



(b) 16 Paths from all Directions  $\mathbf{r}$



# Semi-Global Matching (**SGM**)

$$E(D) = \sum_{\mathbf{p}} C(\mathbf{p}, D_{\mathbf{p}}) + \sum_{\mathbf{q} \in N_{\mathbf{p}}} P_1 T[|D_{\mathbf{p}} - D_{\mathbf{q}}| = 1] + \sum_{\mathbf{q} \in N_{\mathbf{p}}} P_2 T[|D_{\mathbf{p}} - D_{\mathbf{q}}| > 1]$$

У пикселя 8 соседей, из-за взаимного влияния - вычислительно тяжело.

Давайте фиксируем одно направление и будем учитывать только соседа по этому направлению  $\mathbf{r}$ :

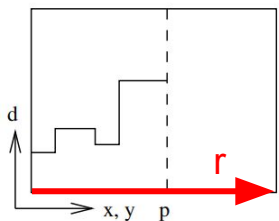
$$L_{\mathbf{r}}(\mathbf{p}, d) = C(\mathbf{p}, d) + \min(L_{\mathbf{r}}(\mathbf{p} - \mathbf{r}, d), L_{\mathbf{r}}(\mathbf{p} - \mathbf{r}, d - 1) + P_1, L_{\mathbf{r}}(\mathbf{p} - \mathbf{r}, d + 1) + P_1, \min_i L_{\mathbf{r}}(\mathbf{p} - \mathbf{r}, i) + P_2) - \min_k L_{\mathbf{r}}(\mathbf{p} - \mathbf{r}, k)$$

*cost* (blue arrow pointing to  $C(\mathbf{p}, d)$ )

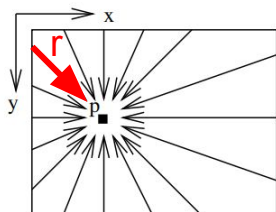
*направление* (red arrow pointing to  $\mathbf{r}$ )

*пиксель шаг  $\mathbf{r}$  назад* (red text below  $\mathbf{p} - \mathbf{r}$ )

(a) Minimum Cost Path  $L_{\mathbf{r}}(\mathbf{p}, d)$



(b) 16 Paths from all Directions  $\mathbf{r}$



# Semi-Global Matching (**SGM**)

$$E(D) = \sum_{\mathbf{p}} C(\mathbf{p}, D_{\mathbf{p}}) + \sum_{\mathbf{q} \in N_{\mathbf{p}}} P_1 T[|D_{\mathbf{p}} - D_{\mathbf{q}}| = 1] + \sum_{\mathbf{q} \in N_{\mathbf{p}}} P_2 T[|D_{\mathbf{p}} - D_{\mathbf{q}}| > 1]$$

У пикселя 8 соседей, из-за взаимного влияния - вычислительно тяжело.

Давайте фиксируем одно направление и будем учитывать только соседа по этому направлению  $\mathbf{r}$ :

$$L_{\mathbf{r}}(\mathbf{p}, d) = C(\mathbf{p}, d) + \min(L_{\mathbf{r}}(\mathbf{p} - \mathbf{r}, d)),$$

*← cost*

$$L_{\mathbf{r}}(\mathbf{p} - \mathbf{r}, d - 1) + P_1, L_{\mathbf{r}}(\mathbf{p} - \mathbf{r}, d + 1) + P_1,$$

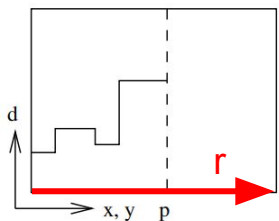
$$\min_i L_{\mathbf{r}}(\mathbf{p} - \mathbf{r}, i) + P_2) - \min_k L_{\mathbf{r}}(\mathbf{p} - \mathbf{r}, k)$$

*← то же смещение*

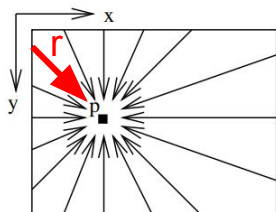
*← направление*

*← пиксель шаг r назад*

(a) Minimum Cost Path  $L_{\mathbf{r}}(\mathbf{p}, d)$



(b) 16 Paths from all Directions  $\mathbf{r}$



# Semi-Global Matching (**SGM**)

$$E(D) = \sum_{\mathbf{p}} C(\mathbf{p}, D_{\mathbf{p}}) + \sum_{\mathbf{q} \in N_{\mathbf{p}}} P_1 T[|D_{\mathbf{p}} - D_{\mathbf{q}}| = 1] + \sum_{\mathbf{q} \in N_{\mathbf{p}}} P_2 T[|D_{\mathbf{p}} - D_{\mathbf{q}}| > 1]$$

У пикселя 8 соседей, из-за взаимного влияния - вычислительно тяжело.

Давайте фиксируем одно направление и будем учитывать только соседа по этому направлению  $\mathbf{r}$ :

$$L_{\mathbf{r}}(\mathbf{p}, d) = C(\mathbf{p}, d) + \min(L_{\mathbf{r}}(\mathbf{p} - \mathbf{r}, d), L_{\mathbf{r}}(\mathbf{p} - \mathbf{r}, d - 1) + P_1, L_{\mathbf{r}}(\mathbf{p} - \mathbf{r}, d + 1) + P_1, \min L_{\mathbf{r}}(\mathbf{p} - \mathbf{r}, i) + P_2), \min L_{\mathbf{r}}(\mathbf{p} - \mathbf{r}, k)$$

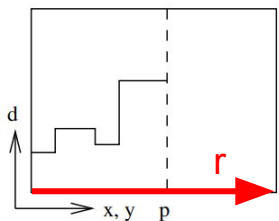
*cost* (pointing to  $C(\mathbf{p}, d)$ )

*направление* (pointing to  $\mathbf{r}$ )

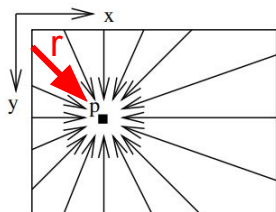
*соседнее смещение* (pointing to  $\mathbf{p} - \mathbf{r}$ )

*пиксель шаг  $\mathbf{r}$  назад* (pointing to  $\mathbf{p} - \mathbf{r}$ )

(a) Minimum Cost Path  $L_{\mathbf{r}}(\mathbf{p}, d)$



(b) 16 Paths from all Directions  $\mathbf{r}$



# Semi-Global Matching (**SGM**)

$$E(D) = \sum_{\mathbf{p}} C(\mathbf{p}, D_{\mathbf{p}}) + \sum_{\mathbf{q} \in N_{\mathbf{p}}} P_1 T[|D_{\mathbf{p}} - D_{\mathbf{q}}| = 1] + \sum_{\mathbf{q} \in N_{\mathbf{p}}} P_2 T[|D_{\mathbf{p}} - D_{\mathbf{q}}| > 1]$$

У пикселя 8 соседей, из-за взаимного влияния - вычислительно тяжело.

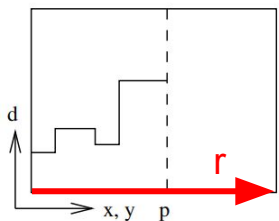
Давайте фиксируем одно направление и будем учитывать только соседа по этому направлению  $\mathbf{r}$ :

$$L_{\mathbf{r}}(\mathbf{p}, d) = C(\mathbf{p}, d) + \min(L_{\mathbf{r}}(\mathbf{p} - \mathbf{r}, d), L_{\mathbf{r}}(\mathbf{p} - \mathbf{r}, d - 1) + P_1, L_{\mathbf{r}}(\mathbf{p} - \mathbf{r}, d + 1) + P_1, \min L_{\mathbf{r}}(\mathbf{p} - \mathbf{r}, i) + P_2) - \min L_{\mathbf{r}}(\mathbf{p} - \mathbf{r}, k)$$

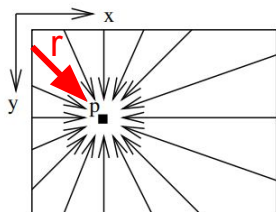
*Handwritten annotations:*

- $\mathbf{r}$ : направление (direction)
- $C(\mathbf{p}, d)$ : cost
- $\min$ : мин
- $i$ : разрыв (break)
- $k$ : против пикселя шаг  $\mathbf{r}$  назад (against pixel step  $\mathbf{r}$  back)
- $P_2$ : переполнения (overflow)

(a) Minimum Cost Path  $L_{\mathbf{r}}(\mathbf{p}, d)$



(b) 16 Paths from all Directions  $\mathbf{r}$



# Semi-Global Matching (**SGM**)

- 1) Ректифицировали стереопару (ради гарантии на горизонтальные сдвиги)



# Semi-Global Matching (**SGM**)

- 1) Ректифицировали стереопару (ради гарантии на горизонтальные сдвиги)
- 2) Преподсчитали **Census** для каждого пикселя у обеих фотографий

# Semi-Global Matching (**SGM**)

- 1) Ректифицировали стереопару (ради гарантии на горизонтальные сдвиги)
- 2) Преподсчитали **Census** для каждого пикселя у обеих фотографий
- 3) **Matching cost computation**: посчитали cost во всем DSI volume

# Semi-Global Matching (**SGM**)

- 1) Ректифицировали стереопару (ради гарантии на горизонтальные сдвиги)
- 2) Преподсчитали **Census** для каждого пикселя у обеих фотографий
- 3) **Matching cost computation**: посчитали cost во всем DSI volume
- 4) **Cost support aggregation**: прошли 16-ю волнами - оценили энергию во всех гипотезах всего объема

# Semi-Global Matching (**SGM**)

- 1) Ректифицировали стереопару (ради гарантии на горизонтальные сдвиги)
- 2) Преподсчитали **Census** для каждого пикселя у обеих фотографий
- 3) **Matching cost computation**: посчитали cost во всем DSI volume
- 4) **Cost support aggregation**: прошли 16-ю волнами - оценили энергию во всех гипотезах всего объема
- 5) **Winner takes all (WTA)**: для каждого пикселя нашли disparity с минимальной энергией - это победитель

**Но как объединять энергию ведь у нас 16 разных направлений?**

# Semi-Global Matching (**SGM**)

- 1) Ректифицировали стереопару (ради гарантии на горизонтальные сдвиги)
- 2) Преподсчитали **Census** для каждого пикселя у обеих фотографий
- 3) **Matching cost computation**: посчитали cost во всем DSI volume
- 4) **Cost support aggregation**: прошли 16-ю волнами - оценили энергию во всех гипотезах всего объема
- 5) **Winner takes all (WTA)**: для каждого пикселя нашли disparity с минимальной энергией - это победитель
- 6) **Refinement**: субпиксельно уточнили disparity-победителя с учетом энергии соседей

# Semi-Global Matching (**SGM**)

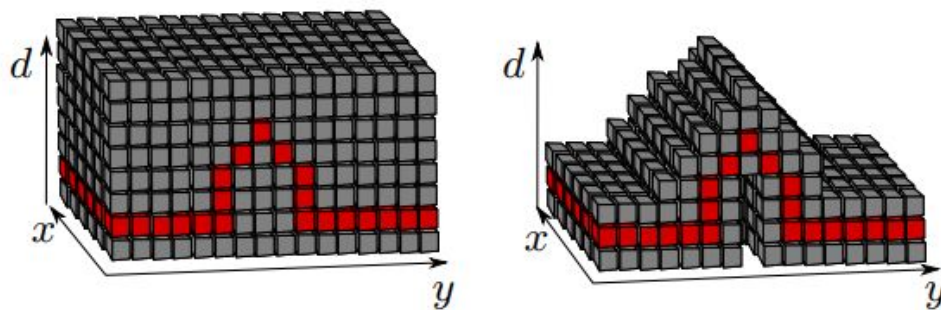
- 1) Ректифицировали стереопару (ради гарантии на горизонтальные сдвиги)
- 2) Преподсчитали **Census** для каждого пикселя у обеих фотографий
- 3) **Matching cost computation**: посчитали cost во всем DSI volume
- 4) **Cost support aggregation**: прошли 16-ю волнами - оценили энергию во всех гипотезах всего объема
- 5) **Winner takes all (WTA)**: для каждого пикселя нашли disparity с минимальной энергией - это победитель
- 6) **Refinement**: субпиксельно уточнили disparity-победителя с учетом энергии соседей
- 7) **Left-right check** или проверка **one-to-one mapping** по диагонали ради удаления ошибок и регионов где не видит один из кадров (**occlusion**)

# Semi-Global Matching (**SGM**)

- 1) Рректифицировали стереопару (ради гарантии на горизонтальные сдвиги)
- 2) Преподсчитали **Census** для каждого пикселя у обеих фотографий
- 3) **Matching cost computation**: посчитали cost во всем DSI volume
- 4) **Cost support aggregation**: прошли 16-ю волнами - оценили энергию во всех гипотезах всего объема
- 5) **Winner takes all (WTA)**: для каждого пикселя нашли disparity с минимальной энергией - это победитель
- 6) **Refinement**: субпиксельно уточнили disparity-победителя с учетом энергии соседей
- 7) **Left-right check** или проверка **one-to-one mapping** по диагонали ради удаления ошибок и регионов где не видит один из кадров (**occlusion**)

Но  $O(W*H*D)$  памяти! (и времени) Как сделать лучше?

# SURE tSGM



Но  $O(W*H*D)$  памяти! (и времени) Как сделать лучше?



# SURE tSGM

Воспользуемся **coarse-to-fine** схемой!

(т.е. будем прогрессировать по пирамиде детализаций стереопары, постепенно уточняя карту диспаритета)

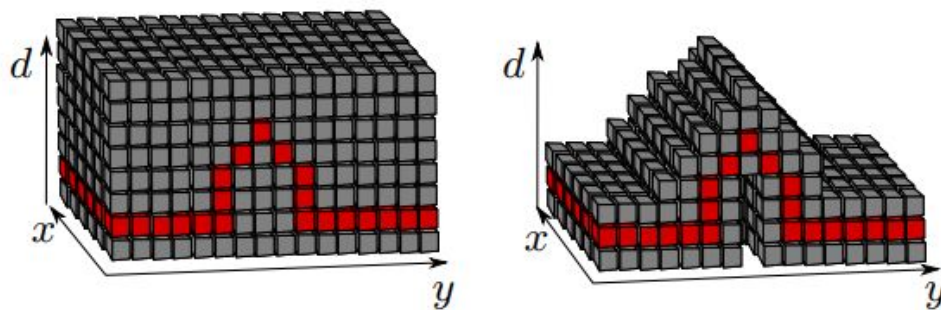
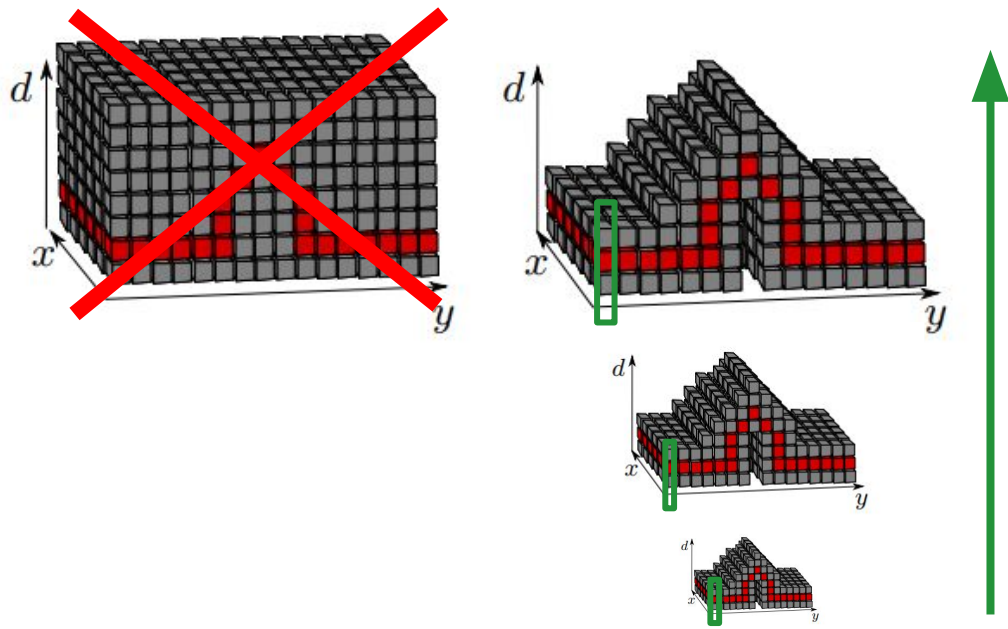


Figure 2: Cost structures of classic SGM (left) and tSGM (right). Red cubes represent costs for the true correspondences. Gray cubes mark the costs of potential correspondences, thus the disparity search ranges.

# SURE tSGM

Как при переходе от менее детальной стереопары и карты глубины к более детальной решить в каком диапазоне требуется выполнять поиск?  
(т.е. агрегацию энергии по 16 направлениям)



# SURE tSGM

Определив карту диспаратета очередного уровня - определяем диапазон дальнейшего поиска:

# SURE tSGM

Определив карту диспаратета очередного уровня - определяем диапазон дальнейшего поиска:

- 1) Если пиксель был успешно сопоставлен - его диапазон это минимум и максимум по диспаратетам в окне  $7 \times 7$

# SURE tSGM

Определив карту диспаратета очередного уровня - определяем диапазон дальнейшего поиска:

- 1) Если пиксель был успешно сопоставлен - его диапазон это минимум и максимум по диспаратетам в окне  $7 \times 7$
- 2) Если пиксель не был сопоставлен - минимум и максимум по диспаратетам в окне  $31 \times 31$ , а в сам пиксель кладем медиану диспаратета по этому же окну  $31 \times 31$

# SURE tSGM

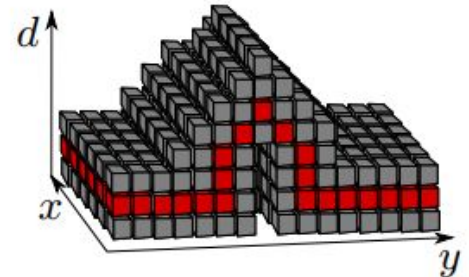
Определив карту диспаритета очередного уровня - определяем диапазон дальнейшего поиска:

- 1) Если пиксель был успешно сопоставлен - его диапазон это минимум и максимум по диспаритетам в окне  $7 \times 7$
- 2) Если пиксель не был сопоставлен - минимум и максимум по диспаритетам в окне  $31 \times 31$ , а в сам пиксель кладем медиану диспаритета по этому же окну  $31 \times 31$

Ограничиваем диапазон до 16 и 32 соответственно. И домножаем на два т.к. переход детальности

**Итого в пространстве поиска диспаритета следующего уровня:**

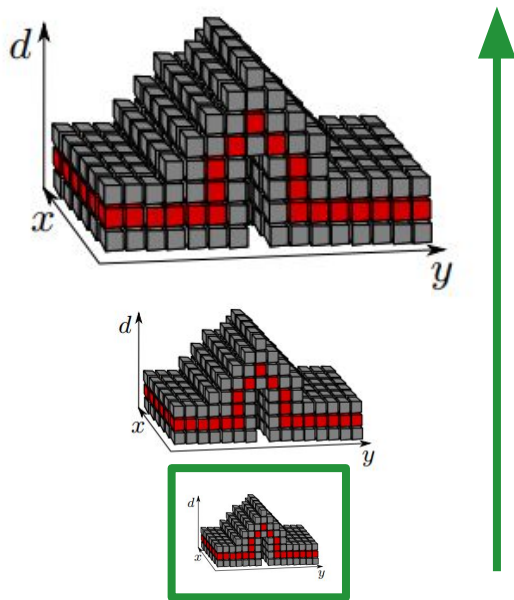
- чаще всего высота столбика - 32 ячейки
- на границах или в сложных местах - 64 ячеек



# SURE tSGM

Сэкономили память и ускорили, теперь  $O(W*H*64)$

**Coarse-to-fine** схема - как **мат. индукция**. Нужна база! **Как найти первый уровень?**



# SURE tSGM

Сэкономили память и ускорили, теперь  $O(W*H*64)$

Как найти **первый уровень**?

- Экстраполяцией смещений по сопоставленным ключевым точкам
- Построить с диапазоном поиска размера  $64 >$  ширина самого недетального первого уровня



# SURE tSGM

Сэкономили память и ускорили, теперь  $O(W*H*64)$

Как найти **первый уровень**?

- Экстраполяцией смещений по сопоставленным ключевым точкам
- Построить с диапазоном поиска размера  $64 >$  ширина самого недетального первого уровня

Бесплатен ли нам по скорости **Left-Right check**?

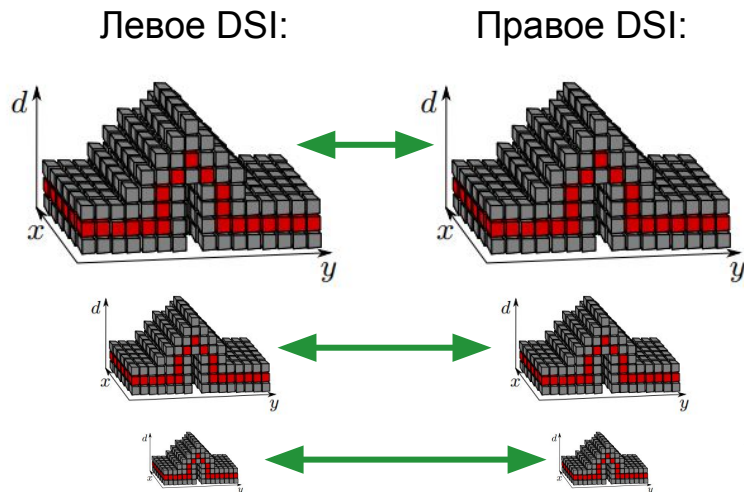
# SURE tSGM

Сэкономили память и ускорили, теперь  $O(W*H*64)$

Как найти **первый уровень**?

- Экстраполяцией смещений по сопоставленным ключевым точкам
- Построить с диапазоном поиска размера  $64 >$  ширина самого недетального первого уровня

Бесплатен ли нам по скорости **Left-Right check**?



# SURE tSGM

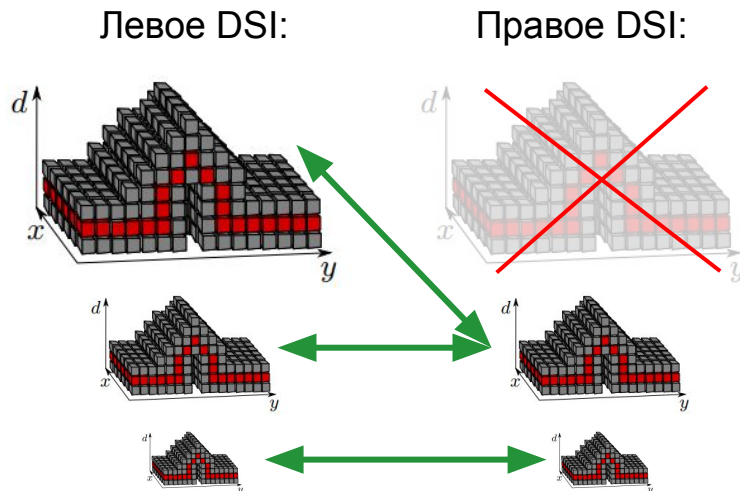
Сэкономили память и ускорили, теперь  $O(W*H*64)$

Как найти **первый уровень**?

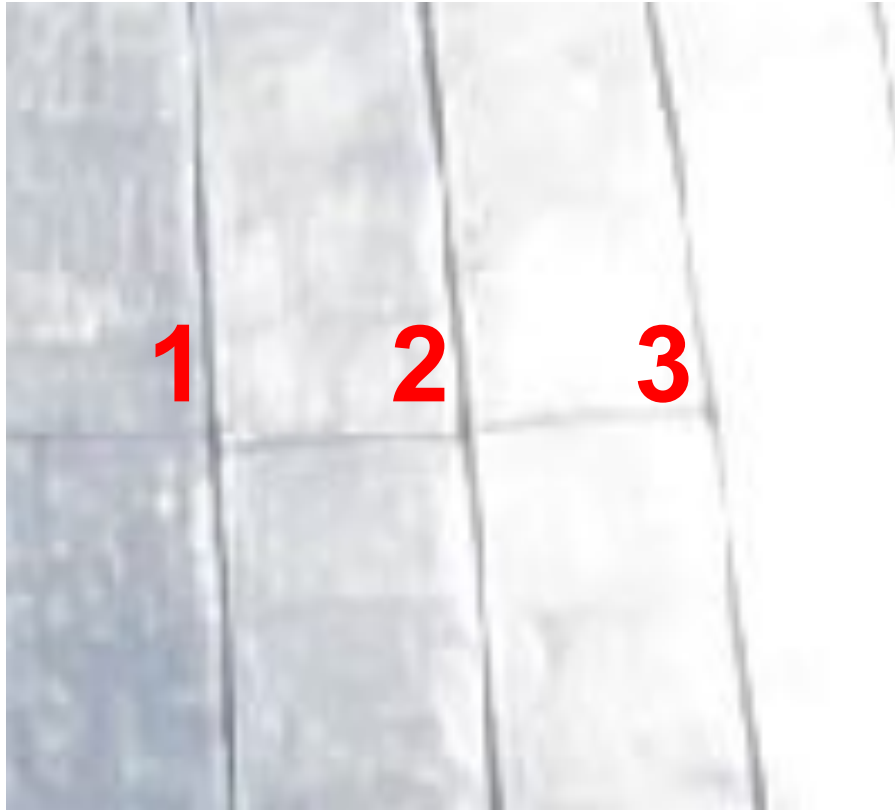
- Экстраполяцией смещений по сопоставленным ключевым точкам
- Построить с диапазоном поиска размера  $64 >$  ширина самого недетального первого уровня

Как ускорить Left-Rigth check?

- Последний уровень диспаратетов строить только слева
- Сверку на симметрию делать с грубой версией справа - т.е. на один уровень меньше



# SURE tSGM



# SURE tSGM



# SURE tSGM



# Итого

- 1) Всё знали про камеры - положение в пространстве + калибровки

## Итого

- 1) Всё знали про камеры - положение в пространстве + калибровки
- 2) Знали грубую геометрию сцены - разреженное 3D облако ключевых точек



# Итого

- 1) Всё знали про камеры - положение в пространстве + калибровки
- 2) Знали грубую геометрию сцены - разреженное 3D облако ключевых точек
- 3) Строим карты глубины для каждой пары камер:

# Итого

- 1) Всё знали про камеры - положение в пространстве + калибровки
- 2) Знали грубую геометрию сцены - разреженное 3D облако ключевых точек
- 3) Строим карты глубины для каждой пары камер:

3.1) **Ректифицировали** - для каждой пары камер построили стереопару:

# Итого

- 1) Всё знали про камеры - положение в пространстве + калибровки
- 2) Знали грубую геометрию сцены - разреженное 3D облако ключевых точек
- 3) Строим карты глубины для каждой пары камер:

3.1) **Ректифицировали** - для каждой пары камер построили стереопару:

3.2) Иерархичным **tSGM** предсказывали диапазон поиска диспаритетов

# Итого

- 1) Всё знали про камеры - положение в пространстве + калибровки
- 2) Знали грубую геометрию сцены - разреженное 3D облако ключевых точек
- 3) Строим карты глубины для каждой пары камер:
  - 3.1) **Ректифицировали** - для каждой пары камер построили стереопару:
  - 3.2) Иерархичным **tSGM** предсказывали диапазон поиска диспаритетов
  - 3.3) Преподсчитывали **Census** описание патча вокруг пикселя

# Итого

- 1) Всё знали про камеры - положение в пространстве + калибровки
- 2) Знали грубую геометрию сцены - разреженное 3D облако ключевых точек
- 3) Строим карты глубины для каждой пары камер:
  - 3.1) **Ректифицировали** - для каждой пары камер построили стереопару:
  - 3.2) Иерархичным **tSGM** предсказывали диапазон поиска диспаритетов
  - 3.3) Преподсчитывали **Census** описание патча вокруг пикселя
  - 3.4) В 16 направлениях считали энергию **Semi-Global Matching (SGM)**

# Итого

- 1) Всё знали про камеры - положение в пространстве + калибровки
- 2) Знали грубую геометрию сцены - разреженное 3D облако ключевых точек
- 3) Строим карты глубины для каждой пары камер:
  - 3.1) **Ректифицировали** - для каждой пары камер построили стереопару:
  - 3.2) Иерархичным **tSGM** предсказывали диапазон поиска диспаритетов
  - 3.3) Преподсчитывали **Census** описание патча вокруг пикселя
  - 3.4) В 16 направлениях считали энергию **Semi-Global Matching (SGM)**
  - 3.5) **WTA** - выбирали диспаритет-победитель + **Left-Right check** для фильтрации

# Итого

- 1) Всё знали про камеры - положение в пространстве + калибровки
- 2) Знали грубую геометрию сцены - разреженное 3D облако ключевых точек
- 3) Строим карты глубины для каждой пары камер:
  - 3.1) **Ректифицировали** - для каждой пары камер построили стереопару:
  - 3.2) Иерархичным **tSGM** предсказывали диапазон поиска диспаритетов
  - 3.3) Преподсчитывали **Census** описание патча вокруг пикселя
  - 3.4) В 16 направлениях считали энергию **Semi-Global Matching (SGM)**
  - 3.5) **WTA** - выбирали диспаритет-победитель + **Left-Right check** для фильтрации
  - 3.6) По значению диспаритета рассчитывали значение глубины

# Итого

- 1) Всё знали про камеры - положение в пространстве + калибровки
- 2) Знали грубую геометрию сцены - разреженное 3D облако ключевых точек
- 3) Строим карты глубины для каждой пары камер:
  - 3.1) **Ректифицировали** - для каждой пары камер построили стереопару:
  - 3.2) Иерархичным **tSGM** предсказывали диапазон поиска диспаритетов
  - 3.3) Преподсчитывали **Census** описание патча вокруг пикселя
  - 3.4) В 16 направлениях считали энергию **Semi-Global Matching (SGM)**
  - 3.5) **WTA** - выбирали диспаритет-победитель + **Left-Right check** для фильтрации
  - 3.6) По значению диспаритета рассчитывали значение глубины



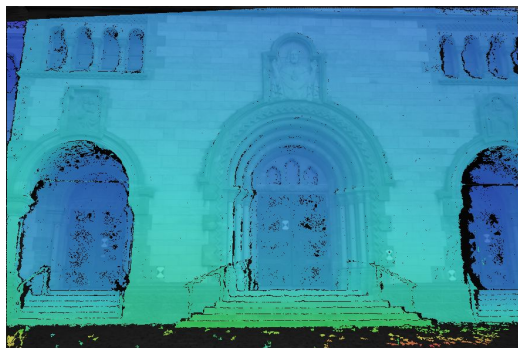
# Итого

- 1) Всё знали про камеры - положение в пространстве + калибровки
- 2) Знали грубую геометрию сцены - разреженное 3D облако ключевых точек
- 3) Строим карты глубины для каждой пары камер:
  - 3.1) **Ректифицировали** - для каждой пары камер построили стереопару:
  - 3.2) Иерархичным **tSGM** предсказывали диапазон поиска диспаритетов
  - 3.3) Преподсчитывали **Census** описание патча вокруг пикселя
  - 3.4) В 16 направлениях считали энергию **Semi-Global Matching (SGM)**
  - 3.5) **WTA** - выбирали диспаритет-победитель + **Left-Right check** для фильтрации
  - 3.6) По значению диспаритета рассчитывали значение глубины

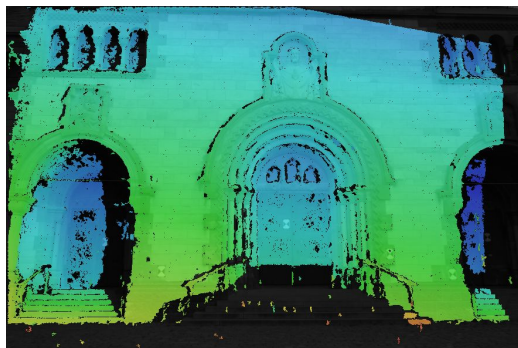
# Итого

- 1) Всё знали про камеры - положение в пространстве + калибровки
- 2) Знали грубую геометрию сцены - разреженное 3D облако ключевых точек
- 3) Строим карты глубины для каждой пары камер:
  - 3.1) **Ректифицировали** - для каждой пары камер построили стереопару:
  - 3.2) Иерархичным **tSGM** предсказывали диапазон поиска диспаритетов
  - 3.3) Преподсчитывали **Census** описание патча вокруг пикселя
  - 3.4) В 16 направлениях считали энергию **Semi-Global Matching (SGM)**
  - 3.5) **WTA** - выбирали диспаритет-победитель + **Left-Right check** для фильтрации
  - 3.6) По значению диспаритета рассчитывали значение глубины

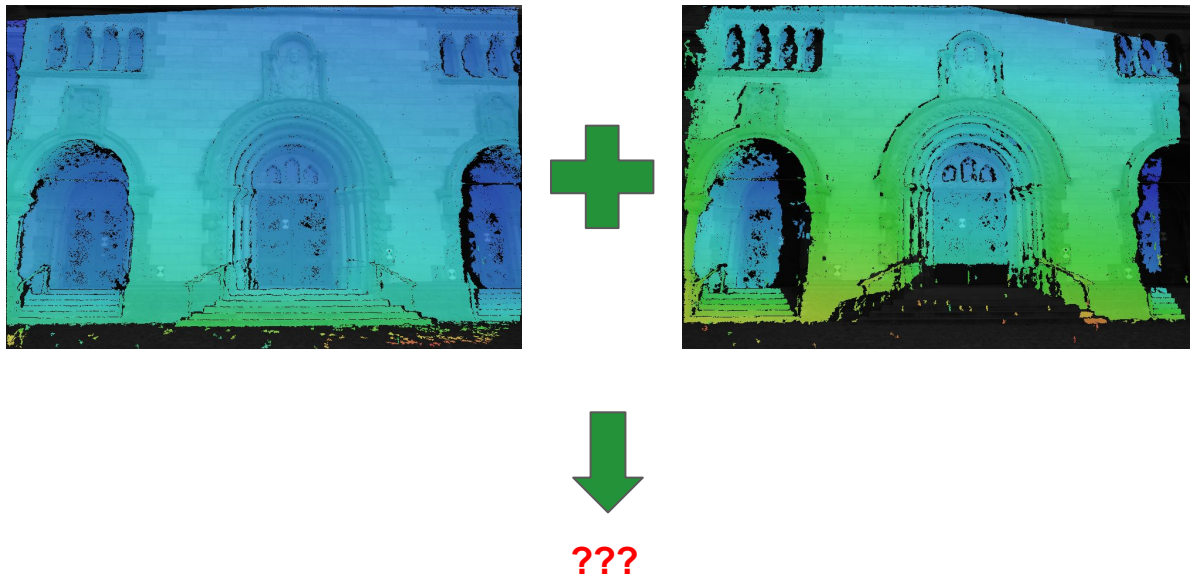
Но ведь мы обрабатывали стереопары!



# Но ведь мы обрабатывали стереопары!



Но ведь мы обрабатывали стереопары! Как объединить?



# Ссылки

Книга (про ректификацию, SGM):

- [Computer Vision: Algorithms and Applications. Richard Szeliski](#)

Cost functions:

- [Evaluation of Stereo Matching Costs on Images with Radiometric Differences, Hirschmuller, 2008](#)

SGM:

- [Accurate and Efficient Stereo Processing by Semi-Global Matching and Mutual Information, Hirschmuller, 2005](#)
- [Stereo Processing by Semi-Global Matching and Mutual Information, Hirschmuller, 2008](#)

SURE tSGM:

- [SURE: PHOTOGRAMMETRIC SURFACE RECONSTRUCTION FROM IMAGERY, Rothmel et. al., 2013](#)

# Вопросы?



Полярный Николай  
[polarnick239@gmail.com](mailto:polarnick239@gmail.com)