

Алгоритм построения панорамы из фотографий 2

- Ключевые точки **SIFT**
- Фильтрация сопоставления ключевых точек:
K-ratio test + Left-Right check + Cluster filtering + RANSAC
- Наложение картинок:
 - оптимизация автоматическим дифференцированием (**Ceres Solver**)
 - компенсация искажений
- Бесшовное смешивание картинок
- Умная прокладка швов

Источник картинки: С. Malin

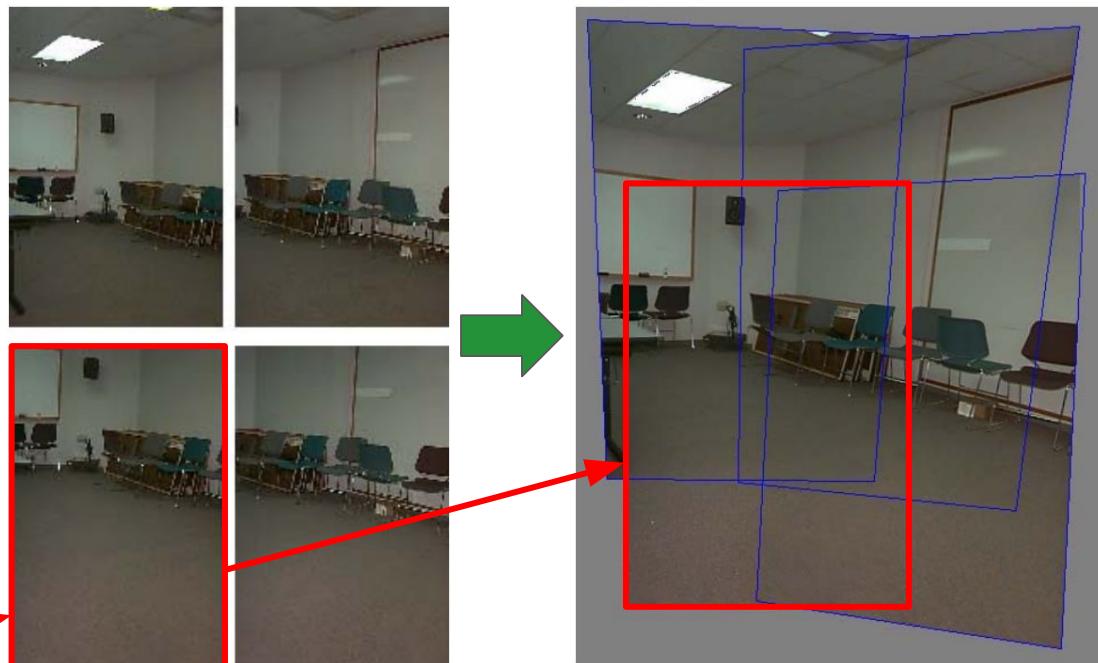
Полярный Николай

polarnick239@gmail.com

Постановка задачи (упрощенная)

Входные данные:

- набор фотографий
- частично накладываются (есть перекрытие)
- все из **одной точки**



Результат:

- проекция фотографий на воображаемую плоскость **первой фотографии**

[Источник картинки](#)

План решения

- 1) Ключевые точки **SIFT** (**Detector: как выбирать ключевые точки?**)
- 2) Фильтрация сопоставления ключевых точек:
K-ratio test + Left-Right check + Cluster filtering + RANSAC
- 3) Наложение картинок:
 - оптимизация автоматическим дифференцированием (**Ceres Solver**)
 - компенсация искажений
- 4) Бесшовное смешивание картинок
- 5) Умная прокладка швов

Наложение картинок: матрица сдвига T

$$\begin{pmatrix} 1 & 0 & a \\ 0 & 1 & b \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} \cdot \\ \cdot \\ \cdot \end{pmatrix}$$

Наложение картинок: матрица сдвига T

$$\begin{pmatrix} 1 & 0 & a \\ 0 & 1 & b \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} \vdots \\ \vdots \\ \vdots \end{pmatrix}$$

Diagram illustrating the matrix multiplication:

- A red arrow above the first column of the first matrix indicates a horizontal shift.
- The element a in the first row of the first matrix is circled in red.
- The elements x , y , and the scalar 1 in the second matrix are circled in red.
- A red arrow points downwards from the scalar 1 in the second matrix to the third row of the result matrix.
- Red dots in the result matrix indicate that the transformation preserves the vertical position of the image.

Наложение картинок: матрица сдвига T

$$\begin{pmatrix} 1 & 0 & a \\ 0 & 1 & b \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \cdot x + 0 \cdot y + a \cdot 1 \\ \cdot \\ \cdot \end{pmatrix}$$

Diagram illustrating the matrix multiplication:

- A red arrow above the first matrix indicates a horizontal shift.
- The first matrix is circled in red, highlighting the column $\begin{pmatrix} 1 & 0 & a \end{pmatrix}$.
- The second matrix is circled in red, highlighting the column $\begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$.
- A red arrow points down from the second matrix to the result, indicating the calculation of the transformed coordinates.
- The result matrix shows the transformed coordinates: $\begin{pmatrix} 1 \cdot x + 0 \cdot y + a \cdot 1 \\ \cdot \\ \cdot \end{pmatrix}$.

Наложение картинок: матрица сдвига T

$$\begin{pmatrix} 1 & 0 & a \\ 0 & 1 & b \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x+a \\ \cdot \\ \cdot \end{pmatrix}$$

Наложение картинок: матрица сдвига T

$$\begin{pmatrix} 1 & 0 & a \\ 0 & 1 & b \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x+a \\ y+b \\ \cdot \end{pmatrix}$$

Наложение картинок: матрица сдвига T

$$\begin{pmatrix} 1 & 0 & a \\ 0 & 1 & b \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x+a \\ y+b \\ 1 \end{pmatrix}$$

Наложение картинок: матрица сдвига T

$$\begin{pmatrix} 1 & 0 & a \\ 0 & 1 & b \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x+a \\ y+b \\ 1 \end{pmatrix}$$

translation matrix T

$$\begin{pmatrix} x \\ y \end{pmatrix} \xrightarrow{(+a, +b)} \begin{pmatrix} x+a \\ y+b \end{pmatrix}$$

Наложение картинок: матрица масштабирования S

$$\begin{pmatrix} ? & ? & ? \\ \cdot & ? & \cdot \\ ? & ? & ? \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} a \cdot x \\ b \cdot y \\ 1 \end{pmatrix}$$

Наложение картинок: матрица масштабирования S

$$\begin{pmatrix} a & & \\ \cdot & \cdot & \cdot \\ \cdot & & \cdot \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} a \cdot x \\ b \cdot y \\ 1 \end{pmatrix}$$

Наложение картинок: матрица масштабирования S

$$\begin{pmatrix} a & \cdot & \cdot \\ \cdot & b & \cdot \\ \cdot & \cdot & \cdot \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} a \cdot x \\ b \cdot y \\ 1 \end{pmatrix}$$

Наложение картинок: матрица масштабирования S

$$\begin{pmatrix} a & 0 & \cdot \\ 0 & b & \cdot \\ \cdot & \cdot & \cdot \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} a \cdot x \\ b \cdot y \\ 1 \end{pmatrix}$$

Наложение картинок: матрица масштабирования S

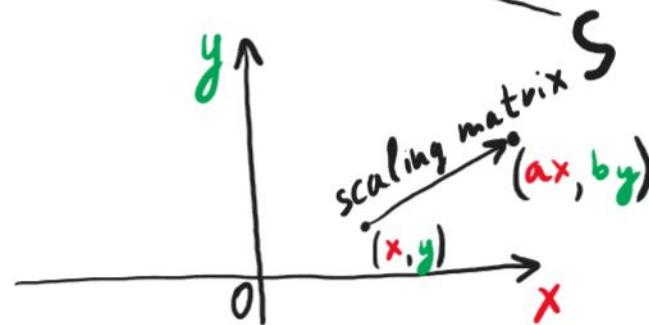
$$\begin{pmatrix} a & 0 & 0 \\ 0 & b & 0 \\ \cdot & \cdot & \cdot \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} a \cdot x \\ b \cdot y \\ 1 \end{pmatrix}$$

Наложение картинок: матрица масштабирования S

$$\begin{pmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} a \cdot x \\ b \cdot y \\ 1 \end{pmatrix}$$

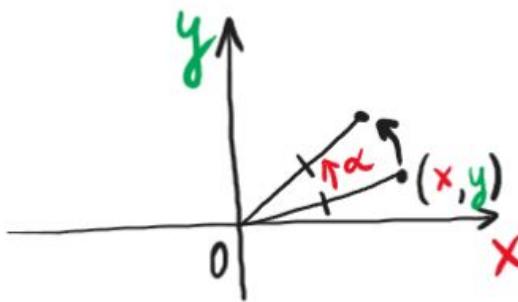
Наложение картинок: матрица масштабирования S

$$\begin{pmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} a \cdot x \\ b \cdot y \\ 1 \end{pmatrix}$$



Наложение картинок: матрица поворота R

rotation matrix

$$R(\alpha) \cdot \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$


Наложение картинок: ассоциативность матриц

$$B \cdot [A \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix}] = (B \cdot A) \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

Наложение картинок: ассоциативность матриц

$$B \cdot [A \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix}] = (B \cdot A) \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

$$C \cdot [B \cdot [A \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix}]] = (C \cdot (B \cdot A)) \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

Наложение картинок: ассоциативность матриц

$$B \cdot [A \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix}] = (B \cdot A) \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

$$C \cdot [B \cdot [A \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix}]] = (C \cdot (B \cdot A)) \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix} = M \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

$\overbrace{\hspace{10em}}^M$

↑
3 2 1

Наложение картинок: ассоциативность матриц

У нас уже есть:

- сдвиг $T(dx, dy)$
- масштабирование $S(sx, sy)$
- поворот $R(\alpha)$

Наложение картинок: ассоциативность матриц

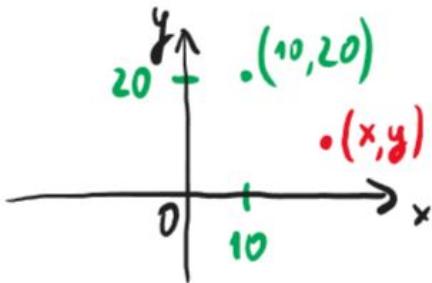
У нас уже есть:

- сдвиг $T(dx, dy)$
- масштабирование $S(sx, sy)$
- поворот $R(\alpha)$

$M =$

Как создать матрицу M которая **поворачивает** вокруг точки $(10, 20)$ на 30 градусов по часовой стрелке и **увеличивает** масштаб вокруг этой же точки $(10, 20)$ в два раза?

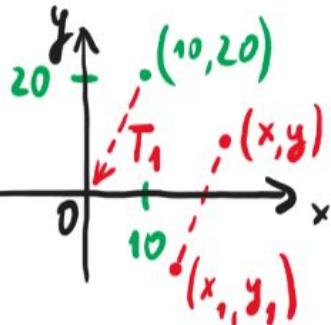
Наложение картинок: ассоциативность матриц



$M =$

Как создать матрицу M которая **поворачивает** вокруг точки $(10, 20)$ на 30 градусов по часовой стрелке и **увеличивает** масштаб вокруг этой же точки $(10, 20)$ в два раза?

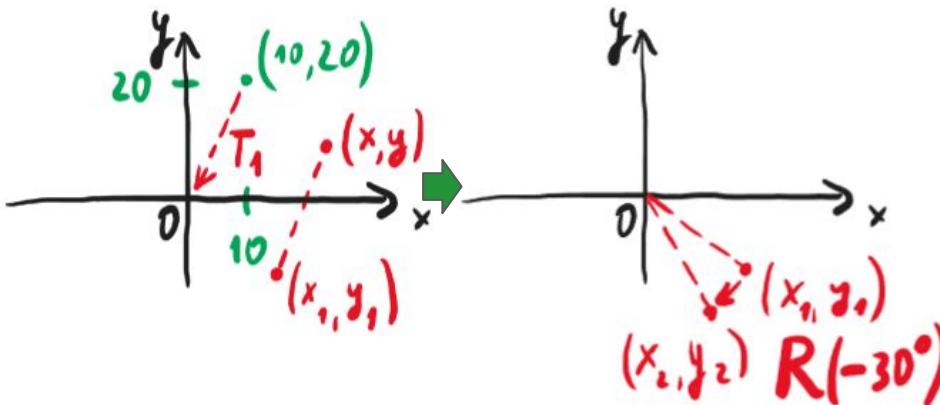
Наложение картинок: ассоциативность матриц



$$M = T_1(-10, -20)$$

Как создать матрицу M которая **поворачивает** вокруг точки $(10, 20)$ на 30 градусов по часовой стрелке и **увеличивает** масштаб вокруг этой же точки $(10, 20)$ в два раза?

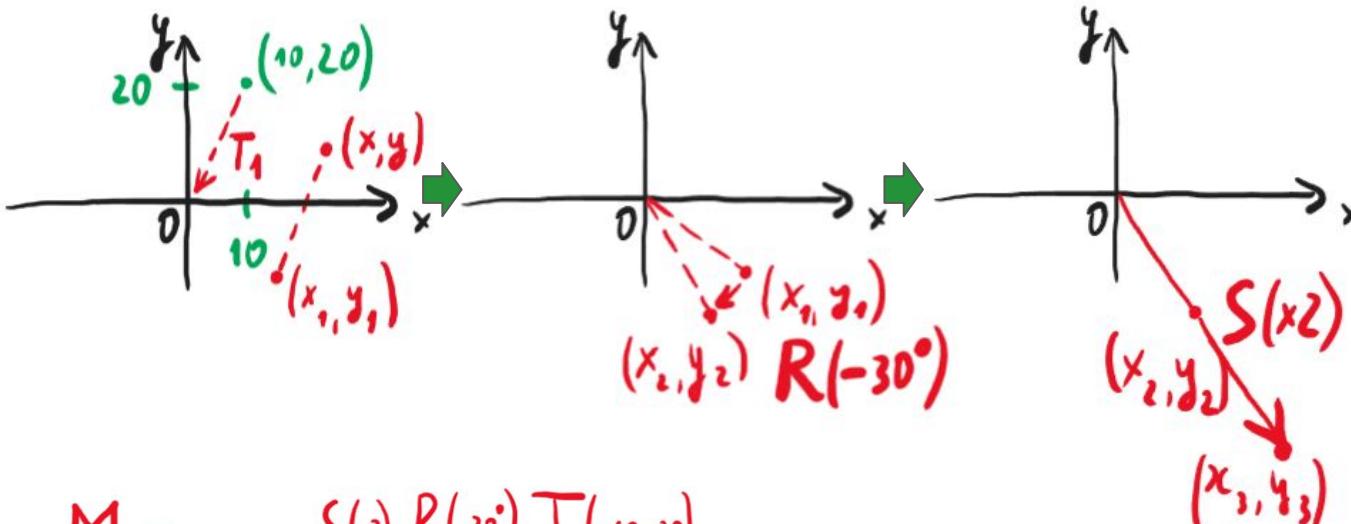
Наложение картинок: ассоциативность матриц



$$M = R(-30^\circ) \cdot T_1(-10, -20)$$

Как создать матрицу M которая **поворачивает** вокруг точки $(10, 20)$ на 30 градусов по часовой стрелке и **увеличивает** масштаб вокруг этой же точки $(10, 20)$ в два раза?

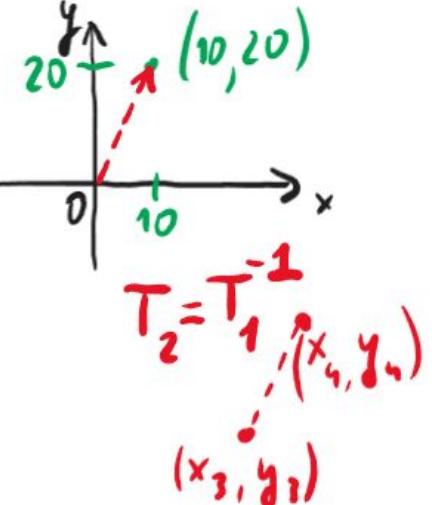
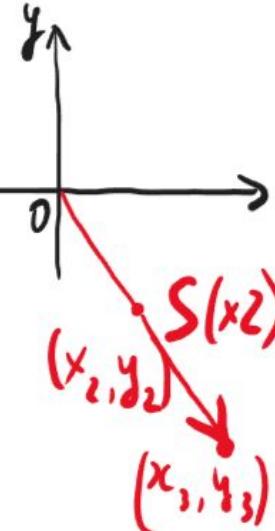
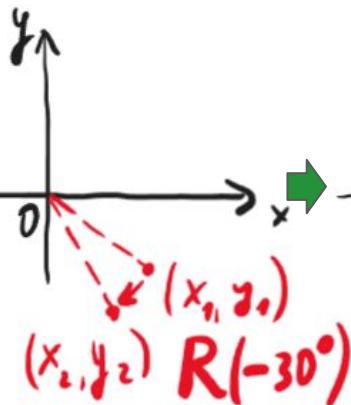
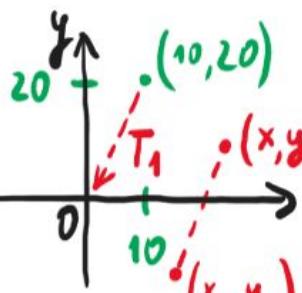
Наложение картинок: ассоциативность матриц



$$M = S(x_2) \cdot R(-30^\circ) \cdot T_1(10, 20)$$

Как создать матрицу M которая **поворачивает** вокруг точки $(10, 20)$ на 30 градусов по часовой стрелке и **увеличивает** масштаб вокруг этой же точки $(10, 20)$ в два раза?

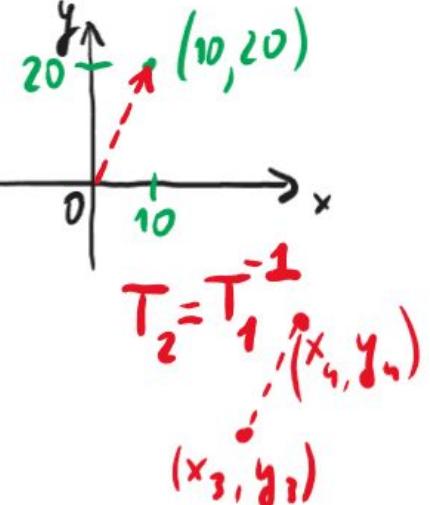
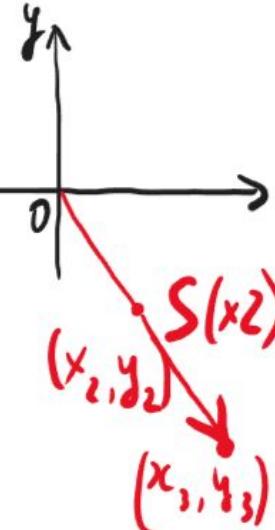
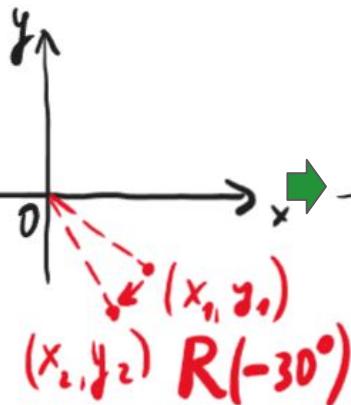
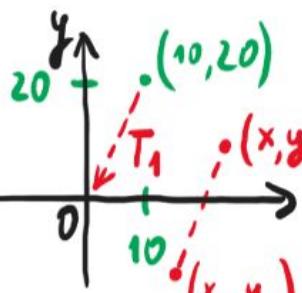
Наложение картинок: ассоциативность матриц



$$M = T_2(10, 20) \cdot S(x_2) \cdot R(-30^\circ) \cdot T_1(-10, -20)$$

Как создать матрицу M которая **поворачивает** вокруг точки $(10, 20)$ на 30 градусов по часовой стрелке и **увеличивает** масштаб вокруг этой же точки $(10, 20)$ в два раза?

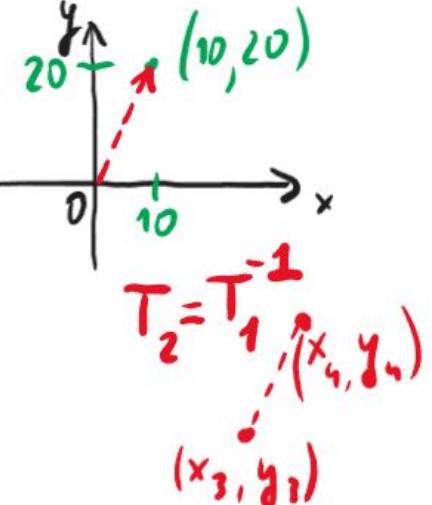
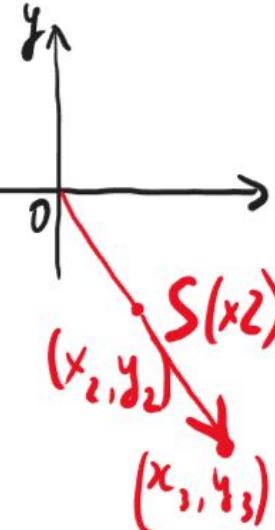
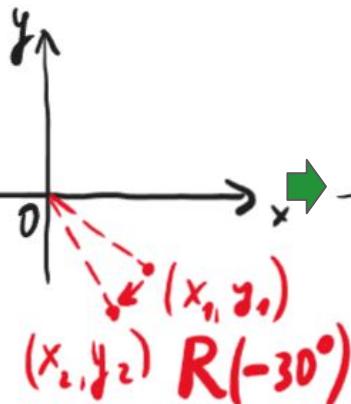
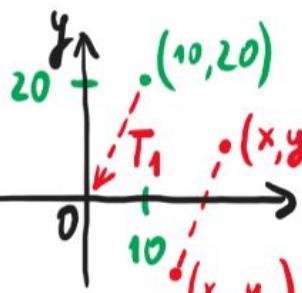
Наложение картинок: ассоциативность матриц



$$M = \underline{T_2(10, 20) \cdot S(x) \cdot R(-30^\circ) \cdot T_1(-10, -20)}$$

Как создать матрицу M которая **поворачивает** вокруг точки $(10, 20)$ на 30 градусов по часовой стрелке и **увеличивает** масштаб вокруг этой же точки $(10, 20)$ в два раза?

Наложение картинок: ассоциативность матриц



$$M = \underline{\underline{T_2^{-1}(+10,20) \cdot S(x_2) \cdot R(-30^\circ) \cdot T_1(-10,-20)}}$$

Как создать матрицу M которая **поворачивает** вокруг точки $(10, 20)$ на 30 градусов по часовой стрелке и **увеличивает** масштаб вокруг этой же точки $(10, 20)$ в два раза?

Матрица гомографии (Homography)

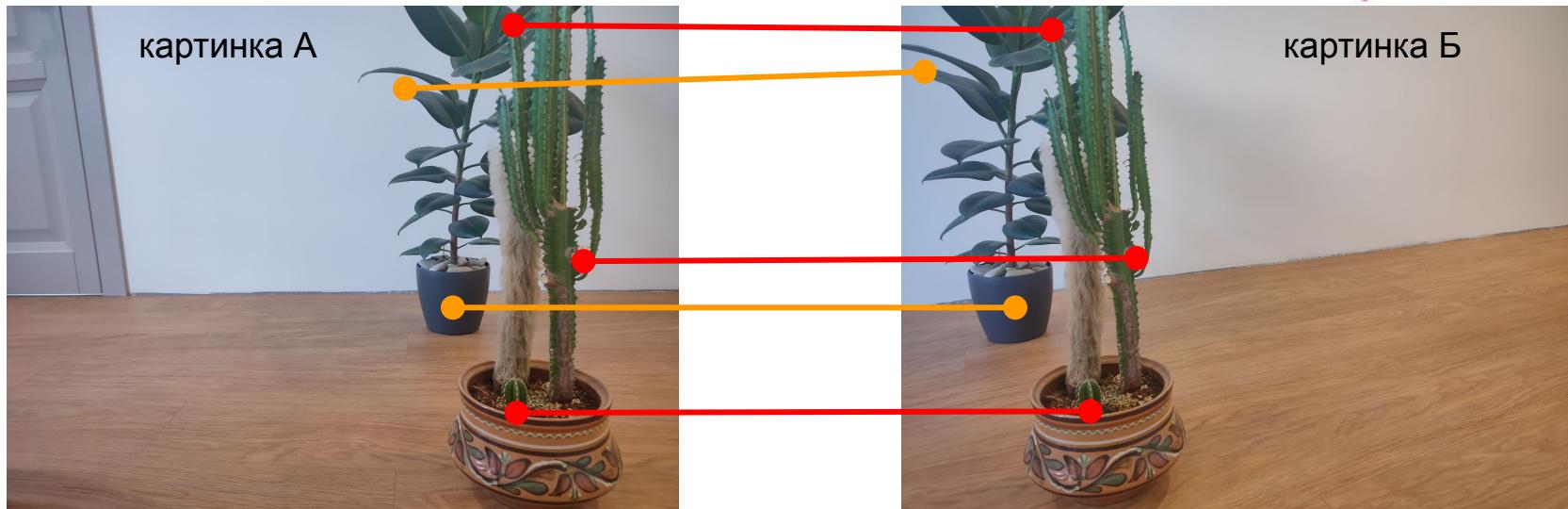
Матрица гомографии **H** описывает любое преобразование точек в одной плоскости.

$$H = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{pmatrix}$$

Матрица гомографии (Homography)

Матрица гомографии **H** описывает любое преобразование точек в одной плоскости.

$$H = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{pmatrix}$$



Матрица гомографии (Homography)

Матрица гомографии **H** описывает любое преобразование точек в одной плоскости.



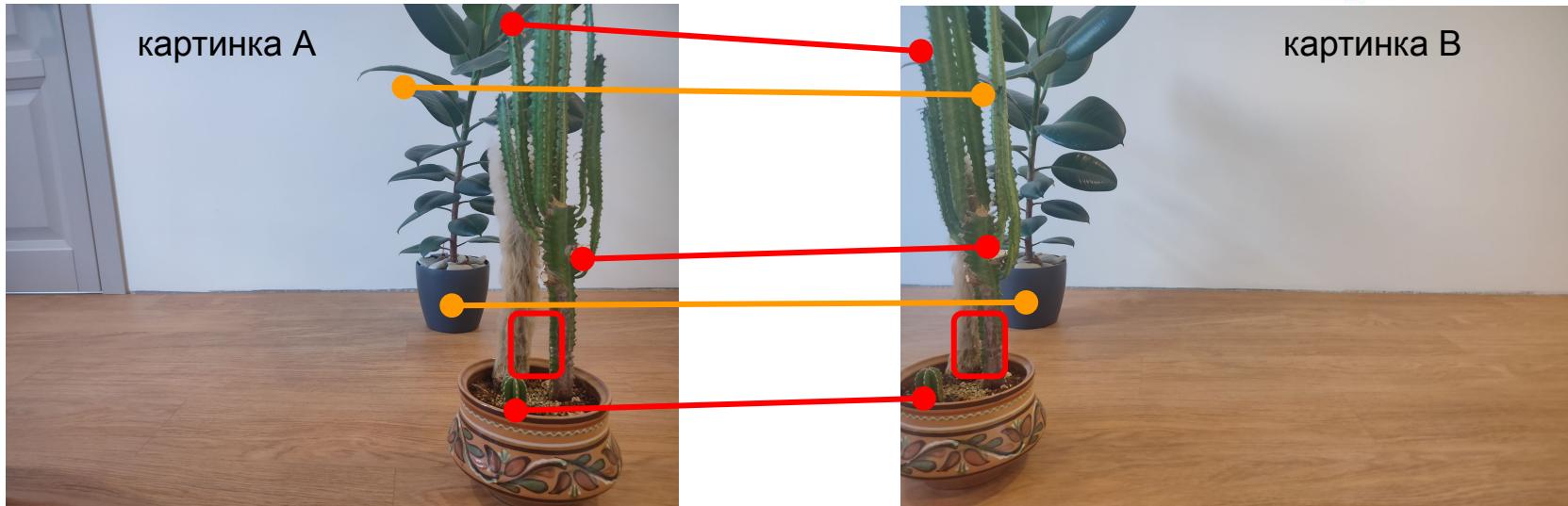
$$H = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{pmatrix}$$

Это матрица гомографии, т.к. камера не двигалась, а значит нет параллакса (смещения по глубине).
Все сопоставления в одной плоскости - плоскости проецирования картинки А.

Матрица гомографии (Homography)

Матрица гомографии **H** описывает любое преобразование точек в одной плоскости.

$$H = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{pmatrix}$$



Матрица гомографии (Homography)

Матрица гомографии **H** описывает любое преобразование точек в одной плоскости.

$$H = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{pmatrix}$$



Это **НЕ** матрица гомографии, т.к. камера **двигалась** - есть параллакс. А значит не в одной плоскости, т.к. есть смещения по глубине.

Матрица гомографии (Homography)

Матрица гомографии **H** описывает любое преобразование точек в одной плоскости.



$$H = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{pmatrix}$$

Матрица гомографии (Homography)

Матрица гомографии **H** описывает любое преобразование точек в одной плоскости.



$$H = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{pmatrix}$$

Это матрица гомографии, т.к. сопоставленные точки лежат (почти) в одной плоскости фасада стеллажа.

Матрица гомографии (Homography)

Матрица гомографии **H** описывает любое преобразование точек в одной плоскости.



$$H = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{pmatrix}$$

Источник

Матрица гомографии (Homography)

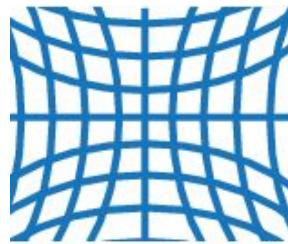
Матрица гомографии **H** описывает любое преобразование точек в одной плоскости.



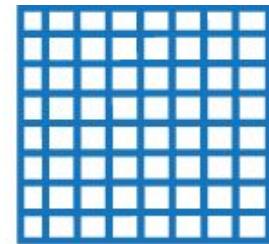
Источник

Это **НЕ** матрица гомографии, т.к. не существует плоскости. Если бы существовала - прямые остались бы прямыми, а непрямые - непрямыми. Сохранение прямых - свойство матрицы гомографии.

$$H = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{pmatrix}$$



Negative radial distortion
"pincushion"



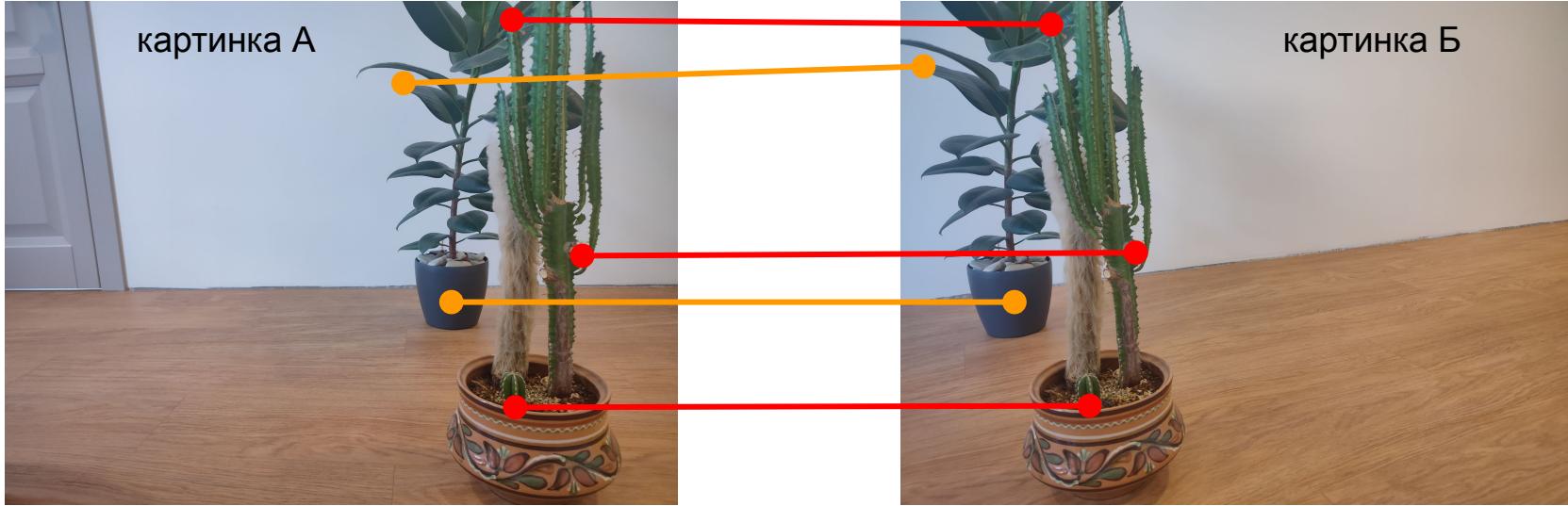
No distortion



Positive radial distortion
"barrel"

источник иллюстрации

Матрица гомографии (Homography)

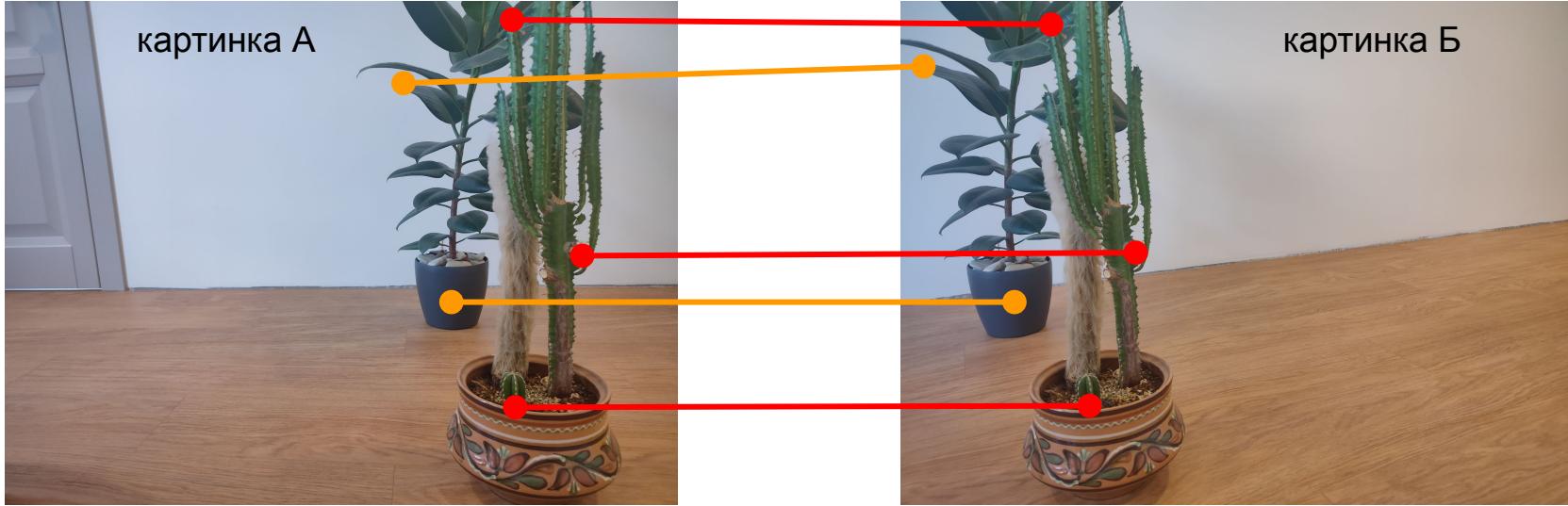


Есть n пар сопоставлений точек:

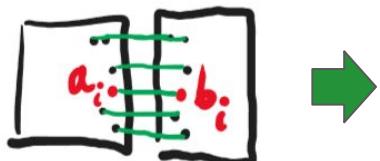


$$\left\{ \begin{array}{l} Ha_0 = b_0 \\ Ha_i = b_i \\ Ha_n = b_n \end{array} \right. \rightarrow H = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{pmatrix}$$

Матрица гомографии (Homography)



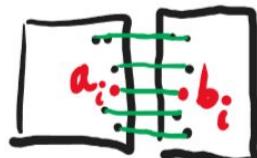
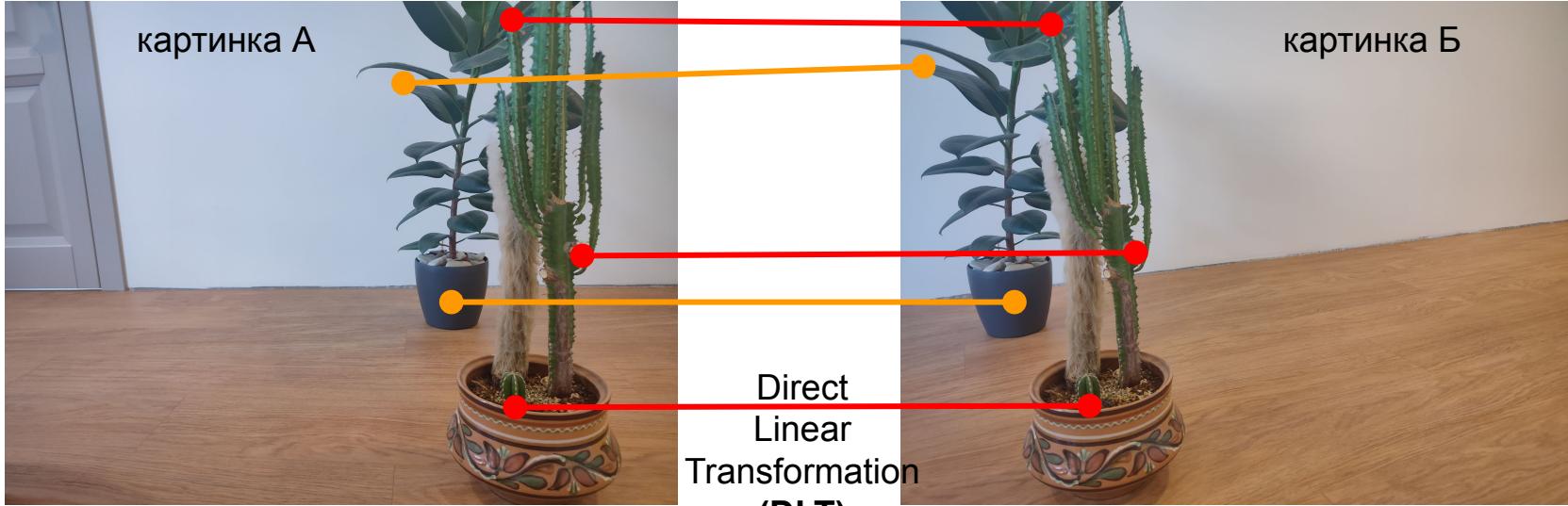
RANSAC
по 4 точкам



$$\left\{ \begin{array}{l} Ha_0 = b_0 \\ Ha_i = b_i \\ Ha_n = b_n \end{array} \right.$$

→ $H = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{pmatrix}$

Матрица гомографии (Homography)

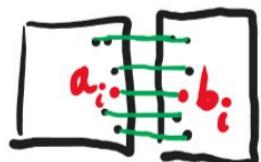
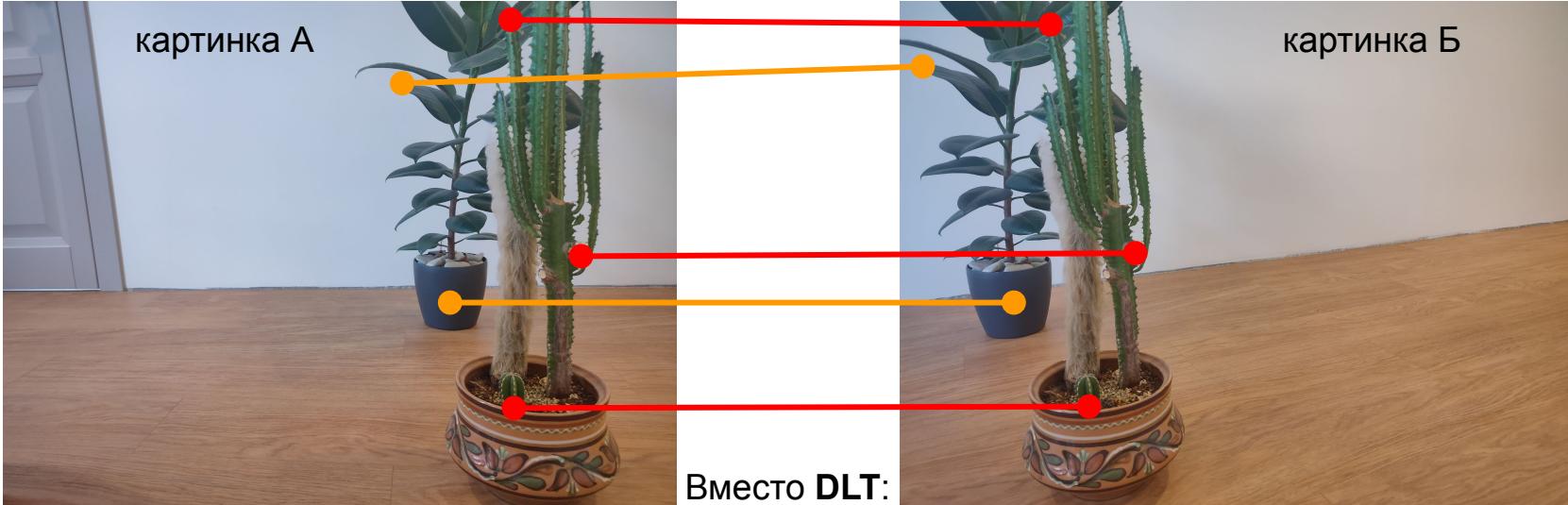


$$\begin{cases} Ha_0 = b_0 \\ Ha_i = b_i \\ Ha_n = b_n \end{cases}$$



$$H = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{pmatrix}$$

Матрица гомографии (Homography)



$$\begin{cases} Ha_0 = b_0 \\ Ha_i = b_i \\ Ha_n = b_n \end{cases}$$



$$H = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{pmatrix}$$

Автоматическое дифференцирование: **Ceres Solver**

Ceres Solver - библиотека на **C++**. Позволяет просто написав функцию которая считает “что мы хотим минимизировать” - найти решение.

Автоматическое дифференцирование: **Ceres Solver**

Ceres Solver - библиотека на **C++**. Позволяет просто написав функцию которая считает “что мы хотим минимизировать” - найти решение.

- Функция должна быть выпуклой (или почти)
- **Библиотека автоматически найдет производные**
- Градиентным спуском найдет минимум

Автоматическое дифференцирование: Dual Numbers

В исходниках **Ceres Solver** есть очень хорошее описание:

[github/ceres-solver/include/ceres/jet.h](https://github.com/ceres-solver/include/ceres/jet.h)

По аналогии с комплексными числами $a + b \cdot i$ дополним число бесконечно малой компонентой. И введем ϵ по аналогии с мнимой единицей i ($i^2 = -1$)

Автоматическое дифференцирование: Dual Numbers

В исходниках **Ceres Solver** есть очень хорошее описание:
[github/ceres-solver/include/ceres/jet.h](https://github.com/ceres-solver/include/ceres/jet.h)

По аналогии с комплексными числами $a+b\cdot i$ дополним число бесконечно малой компонентой. И введем ϵ по аналогии с мнимой единицей i ($i^2 = -1$)

$$\mathbb{R} \xrightarrow{\quad} x + x \cdot \epsilon$$

бесконечно
малое:

$$\epsilon^2 = 0$$

Автоматическое дифференцирование: Dual Numbers

R $x + x \cdot \epsilon$
бесконечно
малое:
 $\epsilon^2 = 0$

Автоматическое дифференцирование: Dual Numbers

\mathbb{R} $x + x \cdot \epsilon$
бесконечно
малое:
 $\epsilon^2 = 0$

$$f(x) = x^2$$

Автоматическое дифференцирование: Dual Numbers

\mathbb{R} $x + x \cdot \epsilon$
бесконечно
малое:
 $\epsilon^2 = 0$

$$f(x) = x^2$$

$$f(10) = 100 \quad f'_x(10) = 20$$

Автоматическое дифференцирование: Dual Numbers

\mathbb{R} $x + x \cdot \varepsilon$
бесконечно
малое:
 $\varepsilon^2 = 0$

$$f(x) = x^2$$

$$f(10) = 100 \quad f'_x(10) = 20$$

$$f(10 + \varepsilon) = (10 + \varepsilon)^2 = 100 + 20 \cdot \varepsilon + \varepsilon^2$$

Автоматическое дифференцирование: Dual Numbers

\mathbb{R} $x + x \cdot \epsilon$
бесконечно
малое:
 $\epsilon^2 = 0$

$$f(x) = x^2$$

$$f(10) = 100 \quad f'_x(10) = 20$$

$$f(10 + \epsilon) = (10 + \epsilon)^2 = 100 + 20 \cdot \epsilon + \cancel{\epsilon^2} = 0$$

Автоматическое дифференцирование: Dual Numbers

\mathbb{R} $x + x \cdot \epsilon$
бесконечно
малое:
 $\epsilon^2 = 0$

$$f(x) = x^2$$

$$f(10) = 100 \quad f'_x(10) = 20$$

$$f(10 + \epsilon) = (10 + \epsilon)^2 = \boxed{100 + 20 \cdot \epsilon} + \cancel{\epsilon^2} = 0$$

dual
number

Автоматическое дифференцирование: Dual Numbers

\mathbb{R} $x + x \cdot \epsilon$
бесконечно
малое:
 $\epsilon^2 = 0$

$$f(x) = x^2$$

$$f(10) = 100 \quad f'_x(10) = 20$$

$$f(10 + \epsilon) = (10 + \epsilon)^2 = \boxed{100 + 20 \cdot \epsilon} + \cancel{\epsilon^2}$$

$\underset{f(10)}{\parallel} \quad \underset{f'_x(10)}{\parallel} \quad \underset{0}{\parallel}$

dual number

Автоматическое дифференцирование: Dual Numbers

\mathbb{R} $x + x \cdot \epsilon$
бесконечно
малое:
 $\epsilon^2 = 0$

$$f(x) = x^2$$

$$f(10) = 100 \quad f'_x(10) = 20$$

$$f(10 + \epsilon) = (10 + \epsilon)^2 = \boxed{100 + 20 \cdot \epsilon} + \cancel{\epsilon^2}$$

$\overset{\parallel}{f(10)} \quad \overset{\parallel}{f'_x(10)} \quad \cancel{\overset{\parallel}{\epsilon^2}} = 0$

dual number

$$f(x_0 + x \cdot \epsilon) = f(x_0) + f'_x(x_0) \cdot x \cdot \epsilon$$

Автоматическое дифференцирование: Dual Numbers

```
1 struct CostFunctor {
2     template <typename T>
3     bool operator()(const T* const x, T* residual) const {
4         // f(x) = x^2
5         residual[0] = x[0] * x[0];
6         return true;
7     }
8 };
```

Автоматическое дифференцирование: Dual Numbers

```
1 struct CostFunctor {  
2     template <typename T>  
3     bool operator()(const T* const x, T* residual) const {  
4         //  $f(x) = x^2$   
5         residual[0] = x[0] * x[0];  
6         return true;  
7     }  
8 };
```

искомое
(оптимизируемые параметры)

Автоматическое дифференцирование: Dual Numbers

```
1 struct CostFunctor {  
2     template <typename T>  
3         bool operator()(const T* const x, T* residual) const {  
4             //  $f(x) = x^2$   
5             residual[0] = x[0] * x[0];  
6             return true;  
7         }  
8     };
```

Т - подставляемый тип
T=double или Jet

искомое
(оптимизируемые параметры)

Автоматическое дифференцирование: Dual Numbers

```
1 struct CostFunctor {  
2     template <typename T>  
3     bool operator()(const T* const x, T* residual) const {  
4         //  $f(x) = x^2$   
5         residual[0] = x[0] * x[0];  
6         return true;  
7     }  
8 };
```

T - подставляемый тип
T=double или Jet

искомое
(оптимизируемые параметры)

```
211 struct Jet {  
285     // The scalar part.  
286     T a;  
288     // The infinitesimal part.  
289     Eigen::Matrix<T, N, 1> v;  
294 };
```

Автоматическое дифференцирование: Dual Numbers

```
1 struct CostFunctor {  
2     template <typename T>  
3     bool operator()(const T* const x, T* residual) const {  
4         //  $f(x) = x^2$   
5         residual[0] = x[0] * x[0];  
6         return true;  
7     }  
8 };
```

T - подставляемый тип

T=double или Jet

искомое
(оптимизируемые параметры)

```
211 struct Jet {  
212     // The scalar part.  
213     T a;  
214     // The infinitesimal part.  
215     Eigen::Matrix<T, N, 1> v;  
216 };
```

Dual Number

Автоматическое дифференцирование: Dual Numbers

$$f(x_0 + \epsilon) = f(x_0) + f'_x(x_0) \cdot \epsilon$$

Проверим правило взятия производной сложной функции (**Chain Rule**).

Автоматическое дифференцирование: Dual Numbers

$$f(x_0 + \epsilon) = f(x_0) + f'_x(x_0) \cdot \epsilon$$

Проверим правило взятия производной сложной функции (**Chain Rule**).

$$[g(f(x_0))]' = g'_x(f(x_0)) \cdot f'_x(x_0)$$

Автоматическое дифференцирование: Dual Numbers

$$f(x_0 + \epsilon) = f(x_0) + f'_x(x_0) \cdot \epsilon$$

Проверим правило взятия производной сложной функции (**Chain Rule**).

$$\left[g(f(x_0)) \right]'_x = g'_x(f(x_0)) \cdot f'_x(x_0)$$

$$g(f(x_0 + \epsilon)) = g\left(f(x_0) + f'_x(x_0) \cdot \epsilon\right)$$

Автоматическое дифференцирование: Dual Numbers

$$f(x_0 + \epsilon) = f(x_0) + f'_x(x_0) \cdot \epsilon$$

Проверим правило взятия производной сложной функции (**Chain Rule**).

$$\begin{aligned} [g(f(x_0))]'_{\color{red}x} &= g'_x(f(x_0)) \cdot f'_x(x_0) \\ g(f(x_0 + \epsilon)) &= g(f(x_0) + f'_x(x_0) \cdot \epsilon) \\ &= g(f(x_0)) + g'_x(f(x_0)) \cdot f'_x(x_0) \cdot \epsilon \end{aligned}$$

Автоматическое дифференцирование: Dual Numbers

$$f(x_0 + \epsilon) = f(x_0) + f'_x(x_0) \cdot \epsilon$$

Проверим правило взятия производной сложной функции (**Chain Rule**).

$$g(f(x_0 + \epsilon)) = g(f(x_0) + f'_x(x_0) \cdot \epsilon)$$

$$= g(f(x_0)) + g'_x(f(x_0)) \cdot f'_x(x_0) \cdot \epsilon$$

$$[g(f(x_0))]'_x = g'_x(f(x_0)) \cdot f'_x(x_0)$$

Автоматическое дифференцирование: Dual Numbers

Осталось **распространить** все базовые операции с вещественных чисел на **dual numbers**:

- Сложение, умножение, вычитание, деление
- Возведение в степень
- Тригонометрические функции
- ...

Автоматическое дифференцирование: Dual Numbers

Осталось **распространить** все базовые операции с вещественных чисел на **dual numbers**:

- Сложение, **умножение**, вычитание, деление
- Возведение в степень
- Тригонометрические функции
- ...

Автоматическое дифференцирование: Dual Numbers

Умножение:

$$(f(x_0) \cdot g(x_0))^{\textcolor{red}{I}}_{\textcolor{red}{x}} =$$

Автоматическое дифференцирование: Dual Numbers

Умножение:

$$(f(x_0) \cdot g(x_0))^{\dagger} = f^{\dagger}(x_0) \cdot g(x_0) + f(x_0) \cdot g^{\dagger}(x_0)$$

Автоматическое дифференцирование: Dual Numbers

Умножение:

$$(f(x_0 + \epsilon) \cdot g(x_0 + \epsilon))' = f'(x_0) \cdot g(x_0) + f(x_0) \cdot g'(x_0)$$
$$(a + b\epsilon) \cdot (c + d\epsilon) =$$

Автоматическое дифференцирование: Dual Numbers

Умножение:

$$\begin{aligned} (f(x_0) \cdot g(x_0))' &= f'(x_0) \cdot g(x_0) + f(x_0) \cdot g'(x_0) \\ (a + b\epsilon) \cdot (c + d\epsilon) &= a \cdot c + (b \cdot c + a \cdot d) \cdot \epsilon + b \cdot d \cdot \cancel{\epsilon} = 0 \end{aligned}$$

Автоматическое дифференцирование: Dual Numbers

Умножение:

теперь это равенство неверно

$$(f(x_0) + \epsilon f'(x_0) \cdot \epsilon) \cdot (g(x_0) + \epsilon g'(x_0) \cdot \epsilon) \neq f'(x_0) \cdot g(x_0) + f(x_0) \cdot g'(x_0)$$
$$(a + b\epsilon) \cdot (c + d\epsilon) = a \cdot c + (b \cdot c + a \cdot d) \cdot \epsilon + b \cdot d \cdot \cancel{\epsilon} = 0$$

Автоматическое дифференцирование: Dual Numbers

Умножение:

теперь это равенство неверно

$$(f(x_0) + \varepsilon \cdot g(x_0)) \cdot (c + d \varepsilon) = f'(x_0) \cdot g(x_0) + f(x_0) \cdot g'(x_0)$$
$$(a + b \varepsilon) \cdot (c + d \varepsilon) = a \cdot c + (b \cdot c + a \cdot d) \cdot \varepsilon + b \cdot d \cdot \varepsilon^2 = 0$$

Автоматическое дифференцирование: Dual Numbers

Умножение:

$$(f(x_0) \cdot g(x_0))^{\prime \epsilon} = f'(x_0) \cdot g(x_0) + f(x_0) \cdot g'(x_0)$$

$$(a + b\epsilon) \cdot (c + d\epsilon) = a \cdot c + (b \cdot c + a \cdot d) \cdot \epsilon + b \cdot d \cdot \cancel{\epsilon}^0$$

$f(x_0) \cdot g(x_0)$

Автоматическое дифференцирование: Dual Numbers

Умножение:

$$(f(x_0) + \epsilon \cdot g(x_0)) \cdot (f(x_0) + \epsilon \cdot g(x_0)) = f'(x_0) \cdot g(x_0) + f(x_0) \cdot g'(x_0)$$

$$(a + b\epsilon) \cdot (c + d\epsilon) = a \cdot c + (b \cdot c + a \cdot d) \cdot \epsilon + b \cdot d \cdot \cancel{\epsilon} = a \cdot c + f(x_0) \cdot g(x_0)$$

Автоматическое дифференцирование: Dual Numbers

Умножение:

$$(f(x_0) + \epsilon \cdot g(x_0)) \cdot (f(x_0) + \epsilon \cdot g(x_0)) = f'(x_0) \cdot g(x_0) + f(x_0) \cdot g'(x_0)$$

$$(a + b\epsilon) \cdot (c + d\epsilon) = a \cdot c + (b \cdot c + a \cdot d) \cdot \epsilon + b \cdot d \cdot \cancel{\epsilon} = a \cdot c + f(x_0) \cdot g(x_0)$$

Автоматическое дифференцирование: Dual Numbers

Умножение:

$$(f(x_0) \cdot g(x_0))^{\prime} = f'(x_0) \cdot g(x_0) + f(x_0) \cdot g'(x_0)$$

$$(a + b\epsilon) \cdot (c + d\epsilon) = a \cdot c + (b \cdot c + a \cdot d) \cdot \epsilon + b \cdot d \cdot \cancel{\epsilon}^0$$

$f(x_0) \cdot g(x_0)$

Автоматическое дифференцирование: Dual Numbers

Умножение:

$$(f(x_0 + \epsilon) \cdot g(x_0 + \epsilon))' = f'(x_0) \cdot g(x_0) + f(x_0) \cdot g'(x_0)$$

$$(a + b\epsilon) \cdot (c + d\epsilon) = a \cdot c + (b \cdot c + a \cdot d) \cdot \epsilon + b \cdot d \cdot \cancel{\epsilon} = a \cdot c + f(x_0) \cdot g(x_0)$$

Автоматическое дифференцирование: Dual Numbers

Умножение:

$$(f(x_0) \cdot g(x_0))^{\prime \epsilon} = f'(x_0) \cdot g(x_0) + f(x_0) \cdot g'(x_0)$$

$$(a + b\epsilon) \cdot (c + d\epsilon) = a \cdot c + (b \cdot c + a \cdot d) \cdot \epsilon + b \cdot d \cdot \cancel{\epsilon}^0$$

$f(x_0) \cdot g(x_0)$

Автоматическое дифференцирование: Dual Numbers

Умножение:

$$(f(x_0) \cdot g(x_0))^{\prime} = f'(x_0) \cdot g(x_0) + f(x_0) \cdot g'(x_0)$$
$$(a + b\epsilon) \cdot (c + d\epsilon) = a \cdot c + (b \cdot c + a \cdot d) \cdot \epsilon + b \cdot d \cdot \cancel{\epsilon}^0$$

Diagram illustrating the multiplication of dual numbers:

- The first equation shows the derivative of the product of two functions \$f\$ and \$g\$ at point \$x_0\$. The result is the sum of the derivative of \$f\$ times \$g\$ and \$f\$ times the derivative of \$g\$.
- The second equation shows the multiplication of two dual numbers \$a + b\epsilon\$ and \$c + d\epsilon\$. The result is the product of the real parts (\$ac\$) plus the product of the derivatives (\$bc + ad\$) times \$\epsilon\$, plus the term \$bd\epsilon\$ which is crossed out and labeled as equal to zero.

Автоматическое дифференцирование: Dual Numbers

Умножение:

$$(f(x_0) + \varepsilon) \cdot (g(x_0) + \varepsilon) = f'(x_0) \cdot g(x_0) + f(x_0) \cdot g'(x_0)$$
$$(a + b\varepsilon) \cdot (c + d\varepsilon) = a \cdot c + (b \cdot c + a \cdot d) \cdot \varepsilon + b \cdot d \cdot \varepsilon^2 = a \cdot c + (b \cdot c + a \cdot d) \cdot \varepsilon + 0$$

Автоматическое дифференцирование: Dual Numbers

Умножение:

$$(f(x_0) \cdot g(x_0))' = f'(x_0) \cdot g(x_0) + f(x_0) \cdot g'(x_0)$$
$$(a + b\epsilon) \cdot (c + d\epsilon) = a \cdot c + (b \cdot c + a \cdot d) \cdot \epsilon + b \cdot d \cdot \epsilon^2 = 0$$

Diagram illustrating the multiplication of dual numbers:

- The first equation shows the derivative of the product of two functions \$f\$ and \$g\$ at point \$x_0\$. The result is the sum of the derivative of \$f\$ times \$g\$ and \$f\$ times the derivative of \$g\$.
- The second equation shows the multiplication of two dual numbers \$(a + b\epsilon)\$ and \$(c + d\epsilon)\$. The result is \$ac + (bc + ad)\epsilon + bd\epsilon^2\$. The term \$bd\epsilon^2\$ is crossed out and labeled "0".
- Annotations:
 - Red \$\epsilon\$ symbols are placed above the terms involving derivatives (\$f'(x_0)\$, \$g'(x_0)\$, \$d\epsilon\$, \$b\epsilon^2\$).
 - Yellow circles highlight the terms \$g(x_0)\$, \$g'(x_0)\$, and \$(c + d\epsilon)\$.
 - Green arrows indicate the components of the dual numbers being multiplied: \$a\$ and \$b\epsilon\$ from the first number, and \$c\$ and \$d\epsilon\$ from the second.
 - Green double-headed arrows connect the terms \$a \cdot c\$ and \$(b \cdot c + a \cdot d) \cdot \epsilon\$ to their counterparts in the first equation.

Автоматическое дифференцирование: Dual Numbers

Умножение:

$$(f(x_0 + \epsilon) \cdot g(x_0 + \epsilon))' = f'(x_0) \cdot g(x_0) + f(x_0) \cdot g'(x_0)$$
$$(a + b\epsilon) \cdot (c + d\epsilon) = a \cdot c + (b \cdot c + a \cdot d) \cdot \epsilon + b \cdot d \cdot \cancel{\epsilon} = 0$$

Diagram illustrating the multiplication of dual numbers:

- The first equation shows the derivative of the product of two functions \$f\$ and \$g\$ at a point \$x_0\$. The terms \$f'(x_0) \cdot g(x_0)\$ and \$f(x_0) \cdot g'(x_0)\$ are highlighted in green.
- The second equation shows the multiplication of two dual numbers \$(a + b\epsilon)\$ and \$(c + d\epsilon)\$. The result is \$a \cdot c + (b \cdot c + a \cdot d) \cdot \epsilon + b \cdot d \cdot \cancel{\epsilon} = 0\$. The term \$a \cdot c\$ is highlighted in green, while the term \$b \cdot d \cdot \cancel{\epsilon}\$ is crossed out with a red marker.

Автоматическое дифференцирование: Dual Numbers

Умножение: [github/ceres-solver/include/ceres/jet.h operator*](https://github.com/ceres-solver/include/ceres/jet.h)

```
347 // Binary *
348 template <typename T, int N>
349 inline Jet<T, N> operator*(const Jet<T, N>& f, const Jet<T, N>& g) {
350     return Jet<T, N>(f.a * g.a, f.a * g.v + f.v * g.a);
351 }
```

Автоматическое дифференцирование: Dual Numbers

Умножение: [github/ceres-solver/include/ceres/jet.h operator*](https://github.com/ceres-solver/include/ceres/jet.h)

```
347 // Binary *
348 template <typename T, int N>
349 inline Jet<T, N> operator*(const Jet<T, N>& f, const Jet<T, N>& g) {
350     return Jet<T, N>(f.a * g.a, f.a * g.v + f.v * g.a);
351 }
```

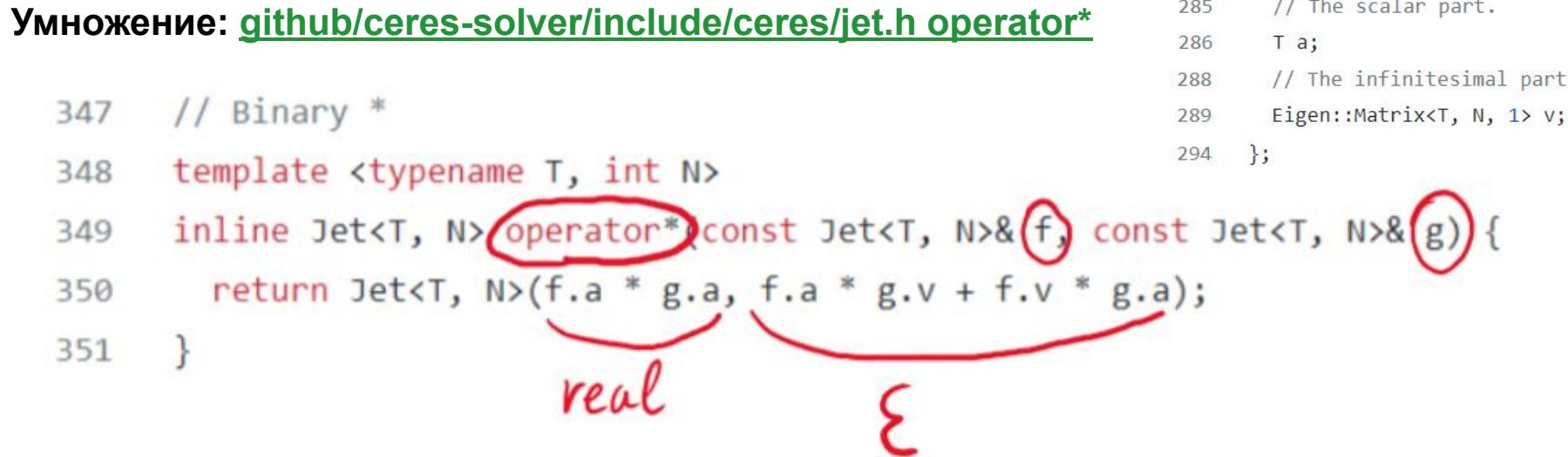
```
211 struct Jet {
285     // The scalar part.
286     T a;
288     // The infinitesimal part.
289     Eigen::Matrix<T, N, 1> v;
294 };
```

Автоматическое дифференцирование: Dual Numbers

Умножение: [github/ceres-solver/include/ceres/jet.h operator*](https://github.com/ceres-solver/include/ceres/jet.h)

```
347 // Binary *
348 template <typename T, int N>
349 inline Jet<T, N> operator*(const Jet<T, N>& f, const Jet<T, N>& g) {
350     return Jet<T, N>(f.a * g.a, f.a * g.v + f.v * g.a);
351 }
```

real {



```
211 struct Jet {
285     // The scalar part.
286     T a;
288     // The infinitesimal part.
289     Eigen::Matrix<T, N, 1> v;
294 };
```

Автоматическое дифференцирование: Dual Numbers

Умножение: [github/ceres-solver/include/ceres/jet.h operator*](https://github.com/ceres-solver/include/ceres/jet.h)

```
347 // Binary *
348 template <typename T, int N>
349 inline Jet<T, N> operator*(const Jet<T, N>& f, const Jet<T, N>& g) {
350     return Jet<T, N>(f.a * g.a, f.a * g.v + f.v * g.a);
351 }
```

```
211 struct Jet {
285     // The scalar part.
286     T a;
288     // The infinitesimal part.
289     Eigen::Matrix<T, N, 1> v;
294 };
```

$$(a+b\epsilon) \cdot (c+d\epsilon) = a \cdot c + (b \cdot c + a \cdot d) \cdot \cancel{\epsilon} + \cancel{b \cdot d} \cdot \cancel{\epsilon^2} = 0$$

Автоматическое дифференцирование: Dual Numbers

Умножение: [github/ceres-solver/include/ceres/jet.h operator*](https://github.com/ceres-solver/include/ceres/jet.h)

```
347 // Binary *
348 template <typename T, int N>
349 inline Jet<T, N> operator*(const Jet<T, N>& f, const Jet<T, N>& g) {
350     return Jet<T, N>(f.a * g.a, f.a * g.v + f.v * g.a);
351 }
```

```
211 struct Jet {
285     // The scalar part.
286     T a;
288     // The infinitesimal part.
289     Eigen::Matrix<T, N, 1> v;
294 };
```

$$(a + b\epsilon) \cdot (c + d\epsilon) = a \cdot c + (b \cdot c + a \cdot d) \cdot \epsilon + b \cdot d \cdot \epsilon^2 = 0$$

Автоматическое дифференцирование: Dual Numbers

Умножение: [github/ceres-solver/include/ceres/jet.h operator*](https://github.com/ceres-solver/include/ceres/jet.h)

```
347 // Binary *
348 template <typename T, int N>
349 inline Jet<T, N> operator*(const Jet<T, N>& f, const Jet<T, N>& g) {
350     return Jet<T, N>(f.a * g.a, f.a * g.v + f.v * g.a);
351 }
```

```
211 struct Jet {
285     // The scalar part.
286     T a;
288     // The infinitesimal part.
289     Eigen::Matrix<T, N, 1> v;
294 };
```

$$\overbrace{(a+b\epsilon)}^f \cdot \overbrace{(c+d\epsilon)}^g = a \cdot c + (b \cdot c + a \cdot d) \cdot \epsilon + b \cdot d \cdot \cancel{\epsilon} \stackrel{!}{=} 0$$

Автоматическое дифференцирование: Dual Numbers

Умножение: [github/ceres-solver/include/ceres/jet.h operator*](https://github.com/ceres-solver/include/ceres/jet.h)

```
347 // Binary *
348 template <typename T, int N>
349 inline Jet<T, N> operator*(const Jet<T, N>& f, const Jet<T, N>& g) {
350     return Jet<T, N>(f.a * g.a, f.a * g.v + f.v * g.a);
351 }
```

```
211 struct Jet {
285     // The scalar part.
286     T a;
288     // The infinitesimal part.
289     Eigen::Matrix<T, N, 1> v;
294 };
```

$$\overbrace{(a+b\epsilon)}^f \cdot \overbrace{(c+d\epsilon)}^g = a \cdot c + (b \cdot c + a \cdot d) \cdot \epsilon + b \cdot d \cdot \cancel{\epsilon} \stackrel{!}{=} 0$$

Автоматическое дифференцирование: Dual Numbers

Умножение: [github/ceres-solver/include/ceres/jet.h operator*](https://github.com/ceres-solver/include/ceres/jet.h)

```
347 // Binary *
348 template <typename T, int N>
349 inline Jet<T, N> operator*(const Jet<T, N>& f, const Jet<T, N>& g) {
350     return Jet<T, N>(f.a * g.a, f.a * g.v + f.v * g.a);
351 }
```

```
211 struct Jet {
285     // The scalar part.
286     T a;
288     // The infinitesimal part.
289     Eigen::Matrix<T, N, 1> v;
294 };
```

$$\overbrace{(a+b\epsilon)}^f \cdot \overbrace{(c+d\epsilon)}^g = \overbrace{a \cdot c} + \overbrace{(b \cdot c + a \cdot d)} \cdot \epsilon + \cancel{\overbrace{b \cdot d \cdot \epsilon^2}} = 0$$

Автоматическое дифференцирование: Dual Numbers

Умножение: [github/ceres-solver/include/ceres/jet.h operator*](https://github.com/ceres-solver/include/ceres/jet.h)

```
347 // Binary *
348 template <typename T, int N>
349 inline Jet<T, N> operator*(const Jet<T, N>& f, const Jet<T, N>& g) {
350     return Jet<T, N>(f.a * g.a, f.a * g.v + f.v * g.a);
351 }
```

```
211 struct Jet {
285     // The scalar part.
286     T a;
288     // The infinitesimal part.
289     Eigen::Matrix<T, N, 1> v;
294 };
```

$$(a+b\epsilon) \cdot (c+d\epsilon) = a \cdot c + (b \cdot c + a \cdot d) \cdot \epsilon + b \cdot d \cdot \epsilon^2 = 0$$

Автоматическое дифференцирование: Dual Numbers

Умножение: [github/ceres-solver/include/ceres/jet.h operator*](https://github.com/ceres-solver/include/ceres/jet.h)

```
347 // Binary *
348 template <typename T, int N>
349 inline Jet<T, N> operator*(const Jet<T, N>& f, const Jet<T, N>& g) {
350     return Jet<T, N>(f.a * g.a, f.a * g.v + f.v * g.a);
351 }
```

```
211 struct Jet {
285     // The scalar part.
286     T a;
288     // The infinitesimal part.
289     Eigen::Matrix<T, N, 1> v;
294 };
```

$$\overbrace{(a+b\epsilon)}^f \cdot \overbrace{(c+d\epsilon)}^g = \overbrace{a \cdot c} + \overbrace{(b \cdot c + a \cdot d)} \cdot \epsilon + \cancel{\overbrace{b \cdot d}} \stackrel{!}{=} 0$$

Автоматическое дифференцирование: Dual Numbers

Умножение: [github/ceres-solver/include/ceres/jet.h operator*](https://github.com/ceres-solver/include/ceres/jet.h)

```
347 // Binary *
348 template <typename T, int N>
349 inline Jet<T, N> operator*(const Jet<T, N>& f, const Jet<T, N>& g) {
350     return Jet<T, N>(f.a * g.a, f.a * g.v + f.v * g.a);
351 }
```

```
211 struct Jet {
285     // The scalar part.
286     T a;
288     // The infinitesimal part.
289     Eigen::Matrix<T, N, 1> v;
294 };
```

$$(a+b\epsilon) \cdot (c+d\epsilon) = a \cdot c + (b \cdot c + a \cdot d) \cdot \epsilon + b \cdot d \cdot \epsilon^2 = 0$$

The diagram illustrates the multiplication of two dual numbers. The first number, f , is represented as $a + b\epsilon$ with a highlighted in yellow and b in orange. The second number, g , is represented as $c + d\epsilon$ with c highlighted in yellow and d in orange. The result of the multiplication is shown as $a \cdot c + (b \cdot c + a \cdot d) \cdot \epsilon + b \cdot d \cdot \epsilon^2$. The terms $a \cdot c$ and $(b \cdot c + a \cdot d) \cdot \epsilon$ are highlighted in green, while $b \cdot d \cdot \epsilon^2$ is crossed out with a large red 'X' and labeled with a question mark '?'.

Автоматическое дифференцирование: Dual Numbers

Умножение: [github/ceres-solver/include/ceres/jet.h operator*](https://github.com/ceres-solver/include/ceres/jet.h)

```
347 // Binary *
348 template <typename T, int N>
349 inline Jet<T, N> operator*(const Jet<T, N>& f, const Jet<T, N>& g) {
350     return Jet<T, N>(f.a * g.a, f.a * g.v + f.v * g.a);
351 }
```

```
211 struct Jet {
285     // The scalar part.
286     T a;
288     // The infinitesimal part.
289     Eigen::Matrix<T, N, 1> v;
294 };
```

$$(a + b\epsilon) \cdot (c + d\epsilon) = a \cdot c + (b \cdot c + a \cdot d) \cdot \epsilon + b \cdot d \cdot \epsilon^2 = 0$$

Автоматическое дифференцирование: Dual Numbers

Умножение: [github/ceres-solver/include/ceres/jet.h operator*](https://github.com/ceres-solver/include/ceres/jet.h)

```
347 // Binary *
348 template <typename T, int N>
349 inline Jet<T, N> operator*(const Jet<T, N>& f, const Jet<T, N>& g) {
350     return Jet<T, N>(f.a * g.a, f.a * g.v + f.v * g.a);
351 }
```

```
211 struct Jet {
285     // The scalar part.
286     T a;
288     // The infinitesimal part.
289     Eigen::Matrix<T, N, 1> v;
294 };
```

$$(a + b\epsilon) \cdot (c + d\epsilon) = a \cdot c + (b \cdot c + a \cdot d) \cdot \epsilon + b \cdot d \cdot \epsilon^2 = 0$$

Автоматическое дифференцирование: Dual Numbers

Умножение: [github/ceres-solver/include/ceres/jet.h operator*](https://github.com/ceres-solver/include/ceres/jet.h)

```
347 // Binary *
348 template <typename T, int N>
349 inline Jet<T, N> operator*(const Jet<T, N>& f, const Jet<T, N>& g) {
350     return Jet<T, N>(f.a * g.a, f.a * g.v + f.v * g.a);
351 }
```

```
211 struct Jet {
285     // The scalar part.
286     T a;
288     // The infinitesimal part.
289     Eigen::Matrix<T, N, 1> v;
294 };
```

$$(a+b\epsilon) \cdot (c+d\epsilon) = a \cdot c + (b \cdot c + a \cdot d) \cdot \epsilon + b \cdot d \cdot \epsilon^2 = 0$$

The diagram illustrates the multiplication of two dual numbers. The first number, f , is represented as $a + b\epsilon$. The second number, g , is represented as $c + d\epsilon$. The product is calculated as follows:

- The scalar part of the result is $a \cdot c$.
- The infinitesimal part of the result is $(b \cdot c + a \cdot d) \cdot \epsilon$.
- The term $b \cdot d \cdot \epsilon^2$ is crossed out with a large red X.
- The final result is set equal to zero, indicated by a red double equals sign ($= 0$).

Наложение картинок

- 1) SIFT Detector + Descriptor
- 2) Nearest Neighbor matching + K-Ratio test + Left-Right check + Cluster filtering

Но у нас все еще остаются выбросы! (ошибочные сопоставления, **outliers**)

4. Cluster filtering



5. RANSAC



Наложение картинок

- 1) SIFT Detector + Descriptor
- 2) Nearest Neighbor matching + K-Ratio test + Left-Right check + Cluster filtering
+ **RANSAC** для поиска **Homography** матрицы:
 - 2.1) много раз берем случайные четыре сопоставления*

* Почему именно четыре? В матрице гомографии 9 неизвестных. Нижняя правая неизвестная ни на что не влияет (умножение матрицы на константу не меняет ее преобразование). Осталось 8 неизвестных (**8 степеней свободы**). Каждая пара точек - это два линейных уравнения (про **x** и про **y**). Чтобы найти 8 неизвестных нужно хотя бы 8 уравнений, значит нужно хотя бы 4 точки.

Наложение картинок

- 1) SIFT Detector + Descriptor
- 2) Nearest Neighbor matching + K-Ratio test + Left-Right check + Cluster filtering
+ **RANSAC** для поиска **Homography** матрицы:
 - 2.1) много раз берем случайные четыре сопоставления
 - 2.2) поверив в их переход строим **Homography** матрицу (DLT)


$$\{ \begin{array}{l} Ha_0 = b_0 \\ Ha_i = b_i \\ Ha_n = b_n \end{array} \rightarrow H = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{pmatrix}$$

Наложение картинок

- 1) SIFT Detector + Descriptor
- 2) Nearest Neighbor matching + K-Ratio test + Left-Right check + Cluster filtering
+ **RANSAC** для поиска **Homography** матрицы:
 - 2.1) много раз берем случайные четыре сопоставления
 - 2.2) поверив в их переход строим **Homography** матрицу (DLT)
 - 2.3) ищем число поддерживающих эту матрицу (число корректных - **inliers**)

Наложение картинок

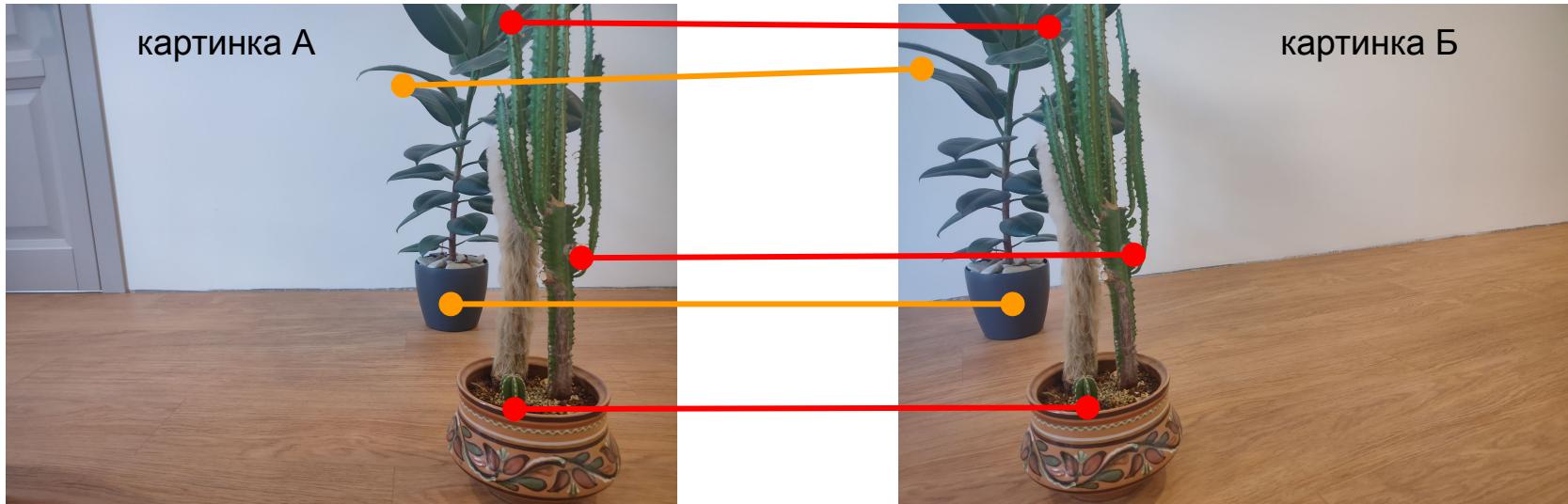
- 1) SIFT Detector + Descriptor
- 2) Nearest Neighbor matching + K-Ratio test + Left-Right check + Cluster filtering + **RANSAC** для поиска **Homography** матрицы:
 - 2.1) много раз берем случайные четыре сопоставления
 - 2.2) поверив в их переход строим **Homography** матрицу (DLT)
 - 2.3) ищем число поддерживающих эту матрицу (число корректных - **inliers**)
 - 2.4) матрица с наибольшей поддержкой (наибольшим числом **inliers**) - верна

Наложение картинок

- 1) SIFT Detector + Descriptor
- 2) Nearest Neighbor matching + K-Ratio test + Left-Right check + Cluster filtering + **RANSAC** для поиска **Homography** матрицы:
 - 2.1) много раз берем случайные четыре сопоставления
 - 2.2) поверив в их переход строим **Homography** матрицу (DLT)
 - 2.3) ищем число поддерживающих эту матрицу (число корректных - **inliers**)
 - 2.4) матрица с наибольшей поддержкой (наибольшим числом **inliers**) - верна
- 3) По **inliers** давайте заново найдем Homography матрицу **H** через Ceres Solver (на этот раз минимизируя общую ошибку)

Наложение картинок

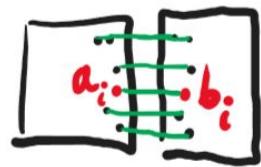
По **inliers** давайте заново найдем Homography матрицу **H** через **Ceres Solver** (на этот раз минимизируя общую ошибку)



$$\begin{cases} Ha_0 = b_0 \\ Ha_i = b_i \\ Ha_n = b_n \end{cases} \rightarrow H = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{pmatrix}$$

Наложение картинок

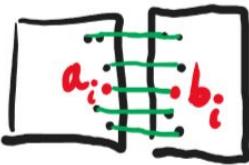
По **inliers** давайте заново найдем Homography матрицу **H** через **Ceres Solver** (на этот раз минимизируя общую ошибку)



$$\left\{ \begin{array}{l} H a_0 = b_0 \\ H a_i = b_i \\ H a_n = b_n \end{array} \right.$$

Наложение картинок

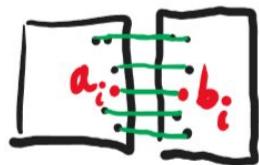
По **inliers** давайте заново найдем Homography матрицу **H** через **Ceres Solver** (на этот раз минимизируя общую ошибку)



→ { $\begin{aligned} H a_0 - b_0 &= 0 \\ H a_i - b_i &= 0 \\ H a_n - b_n &= 0 \end{aligned}$ }

Наложение картинок

По **inliers** давайте заново найдем Homography матрицу **H** через **Ceres Solver** (на этот раз минимизируя общую ошибку)



$$\{ \begin{array}{l} Ha_0 - b_0 \approx 0 \\ Ha_i - b_i \approx 0 \\ Ha_n - b_n \approx 0 \end{array}$$

Наложение картинок

По **inliers** давайте заново найдем Homography матрицу H через **Ceres Solver** (на этот раз минимизируя общую ошибку)

$$f(H) = \sum (H \cdot a_i - b_i)^2$$

Наложение картинок

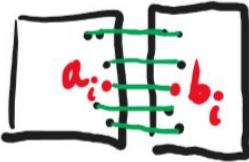
По **inliers** давайте заново найдем Homography матрицу \mathbf{H} через **Ceres Solver** (на этот раз минимизируя общую ошибку)

The diagram illustrates the process of finding a homography matrix \mathbf{H} using Ceres Solver. On the left, two images are shown with corresponding feature points a_i and b_i marked by green lines. A green arrow points from this image to a mathematical expression. The expression consists of three terms, each involving the homography matrix \mathbf{H} , a source point a_i , and a target point b_i . The terms are separated by plus signs and enclosed in curly braces, indicating they are summed. Below the expression is a green bracket under the entire sum, followed by the equation $f(\underline{\mathbf{H}}) \rightarrow 0$, where $\underline{\mathbf{H}}$ indicates the matrix \mathbf{H} is being solved for.

$$f(\underline{\mathbf{H}}) \rightarrow 0$$

Наложение картинок

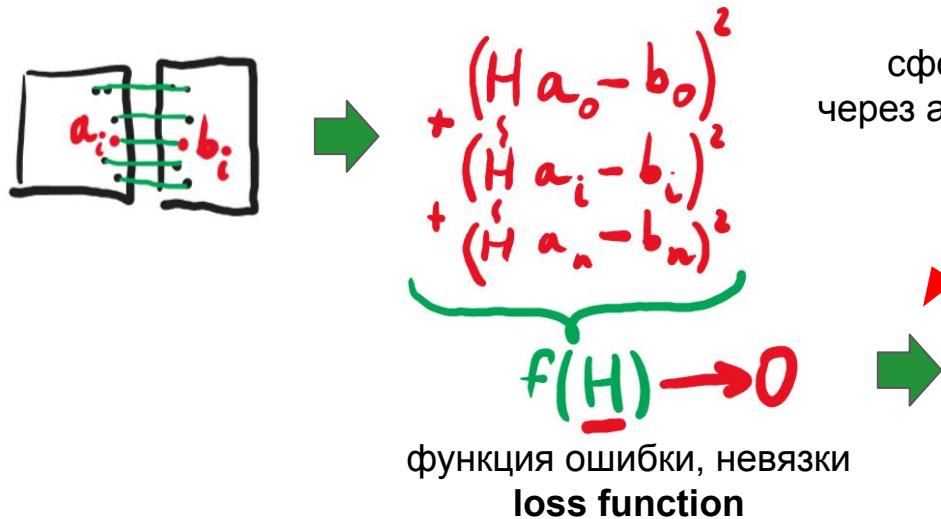
По **inliers** давайте заново найдем Homography матрицу H через **Ceres Solver** (на этот раз минимизируя общую ошибку)


$$\begin{aligned} & \xrightarrow{\text{ }} \\ & + (H a_0 - b_0)^2 \\ & + (H a_i - b_i)^2 \\ & + (H a_n - b_n)^2 \\ & f(\underline{H}) \rightarrow 0 \end{aligned}$$

функция ошибки, невязки
loss function

Наложение картинок

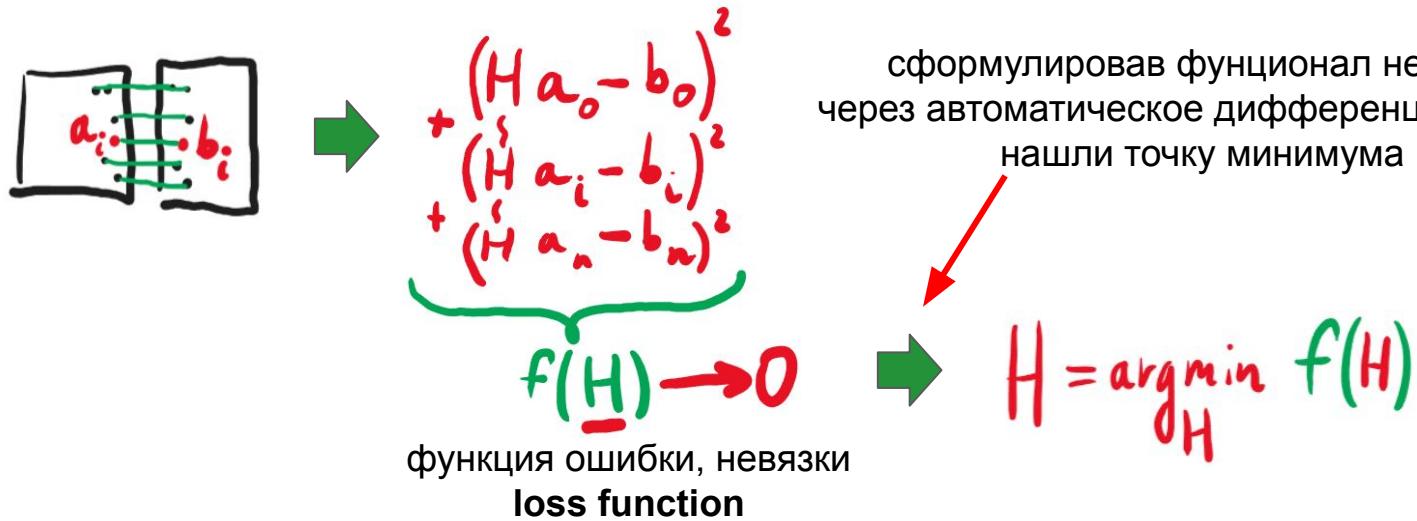
По **inliers** давайте заново найдем Homography матрицу **H** через **Ceres Solver** (на этот раз минимизируя общую ошибку)



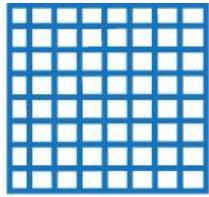
сформулировав функционал невязки
через автоматическое дифференцирование
нашли точку минимума

Наложение картинок

По **inliers** давайте заново найдем Homography матрицу H через **Ceres Solver** (на этот раз минимизируя общую ошибку)



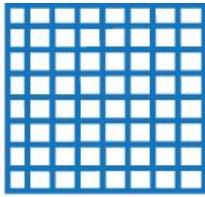
Искажение радиальной дисторсии в линзе



No distortion

[источник иллюстрации](#)

Искажение радиальной дисторсии в линзе



No distortion

источник иллюстрации

???????

???????



Источник



Искажение радиальной дисторсии в линзе

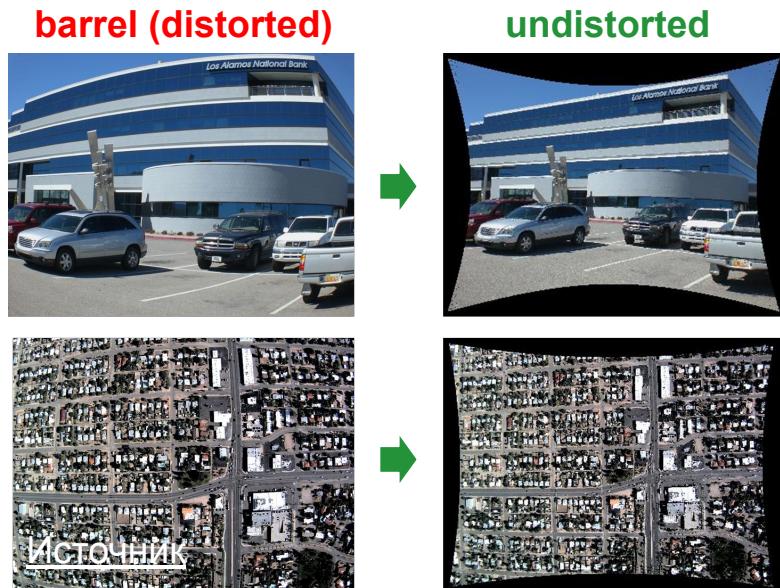


???????

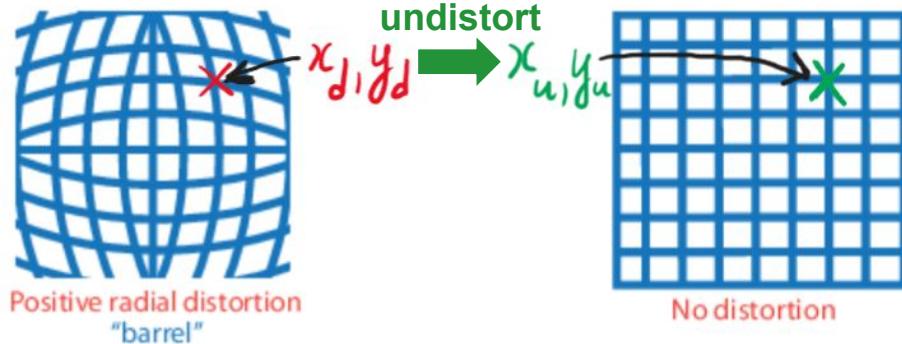
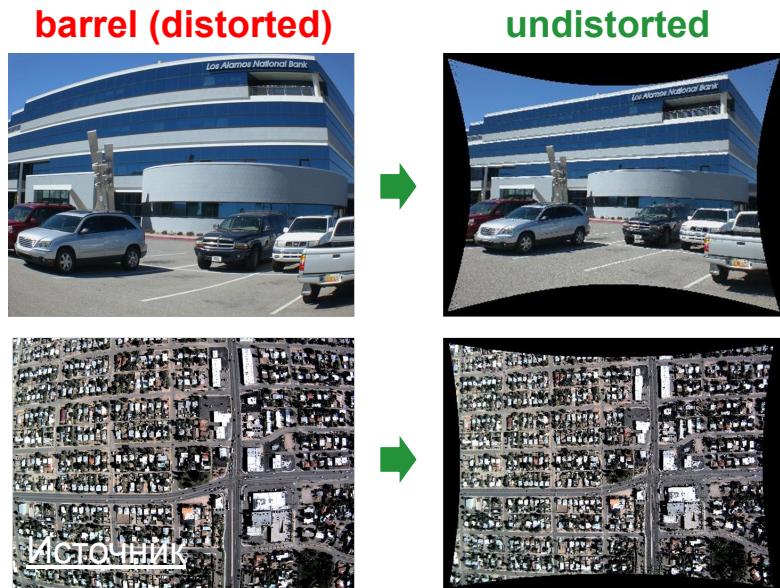
???????



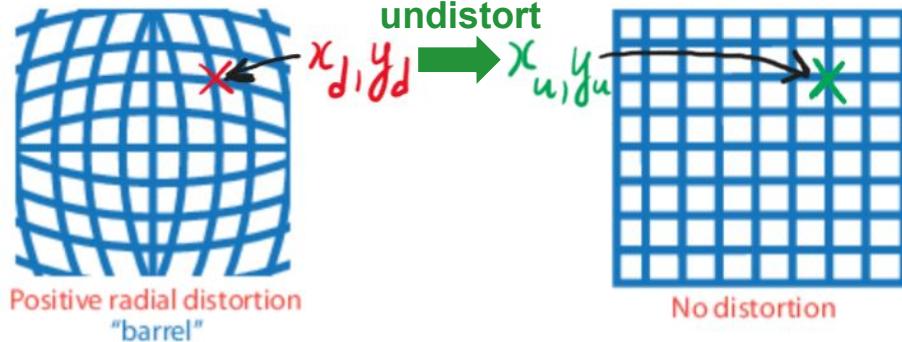
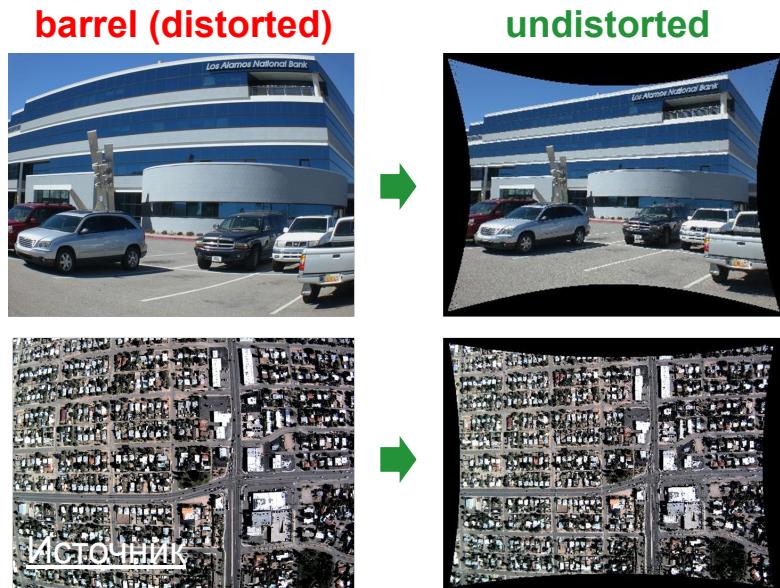
Искажение радиальной дисторсии в линзе



Искажение радиальной дисторсии в линзе



Искажение радиальной дисторсии в линзе



Искажение радиальной дисторсии в линзе

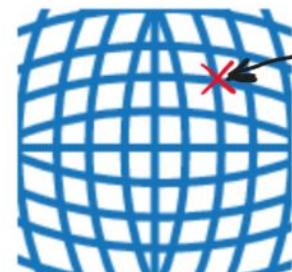


$K_1, K_2 \dots$ - коэффициенты радиальной дисторсии

barrel (distorted)

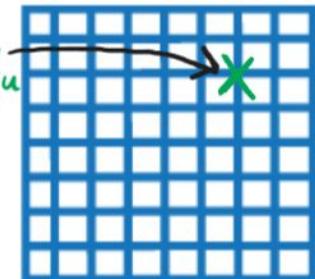


undistorted



Positive radial distortion
"barrel"

undistort
 $x_d, y_d \rightarrow x_u, y_u$

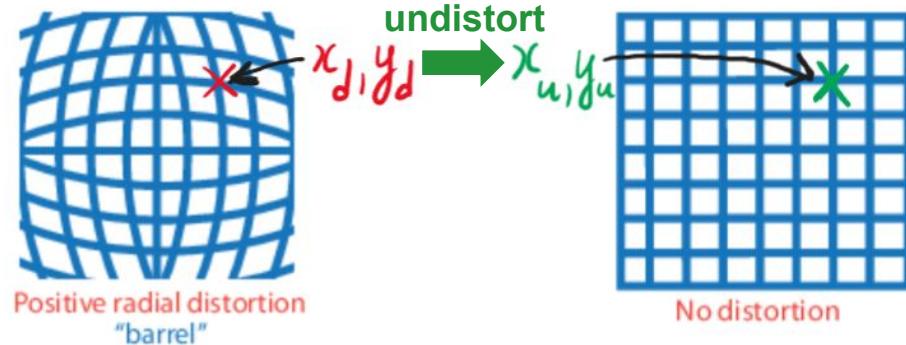
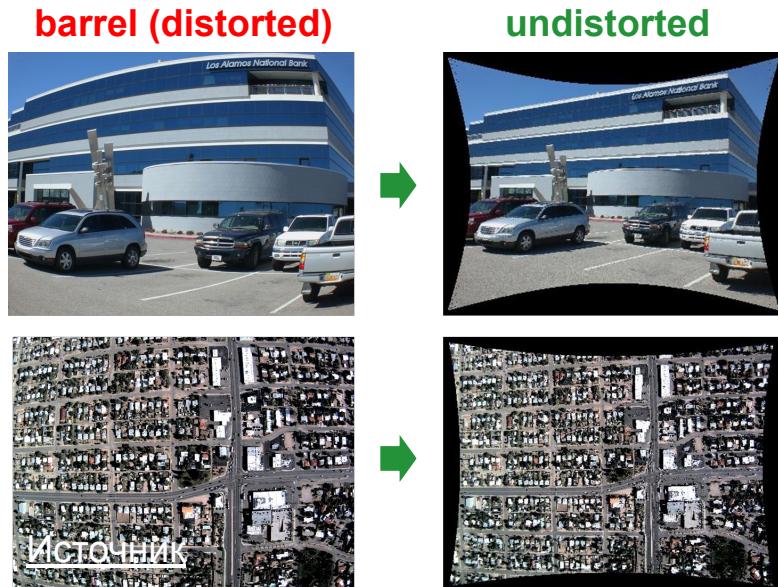


No distortion

Искажение радиальной дисторсии в линзе



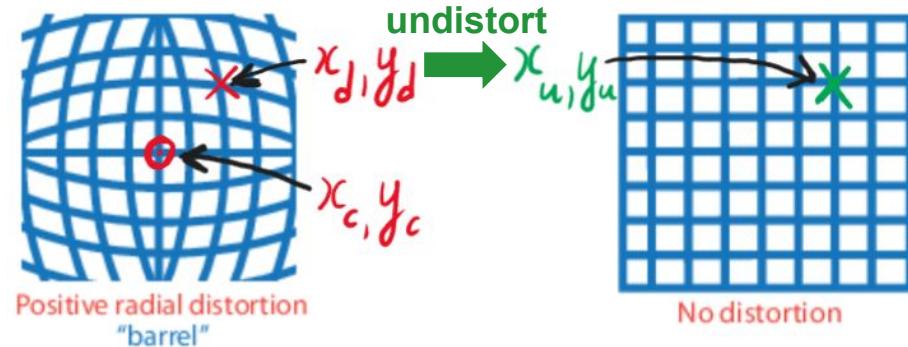
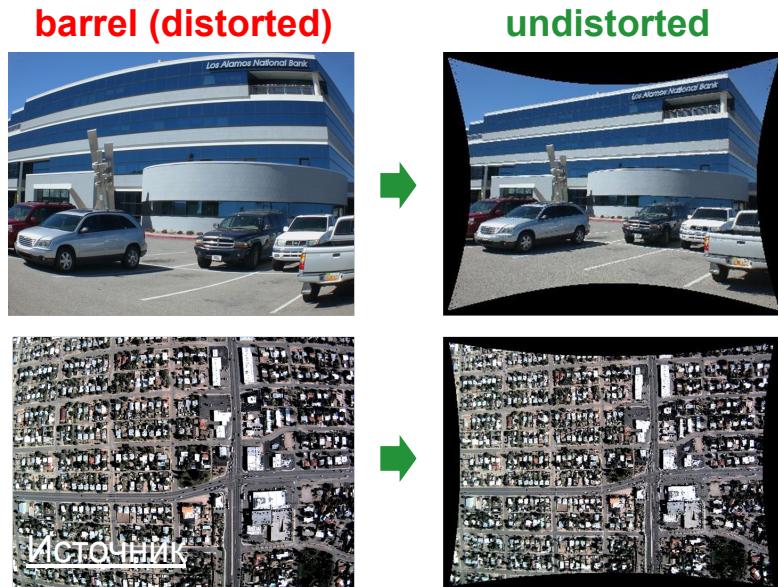
$K_1, K_2 \dots$ - коэффициенты радиальной дисторсии
 (x_c, y_c) - центр оптической оси (центр картинки)



Искажение радиальной дисторсии в линзе



$K_1, K_2 \dots$ - коэффициенты радиальной дисторсии
 (x_c, y_c) - центр оптической оси (центр картинки)

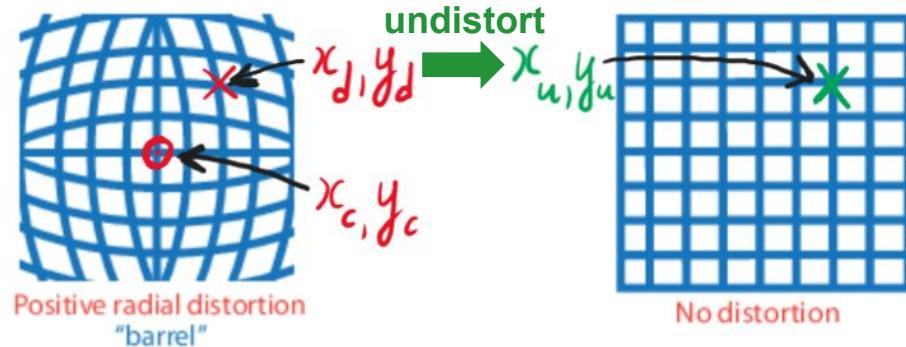
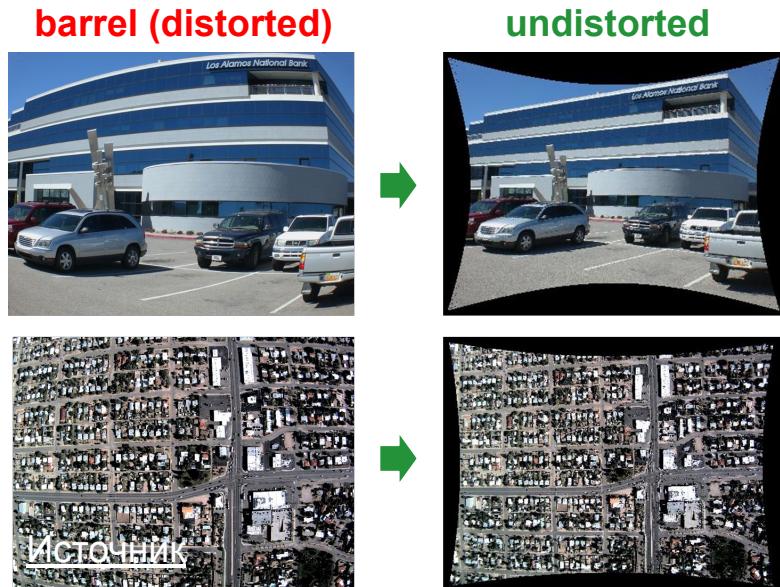


Искажение радиальной дисторсии в линзе



$K_1, K_2 \dots$ - коэффициенты радиальной дисторсии
 (x_c, y_c) - центр оптической оси (центр картинки)

$$r = \sqrt{(x_d - x_c)^2 + (y_d - y_c)^2}$$

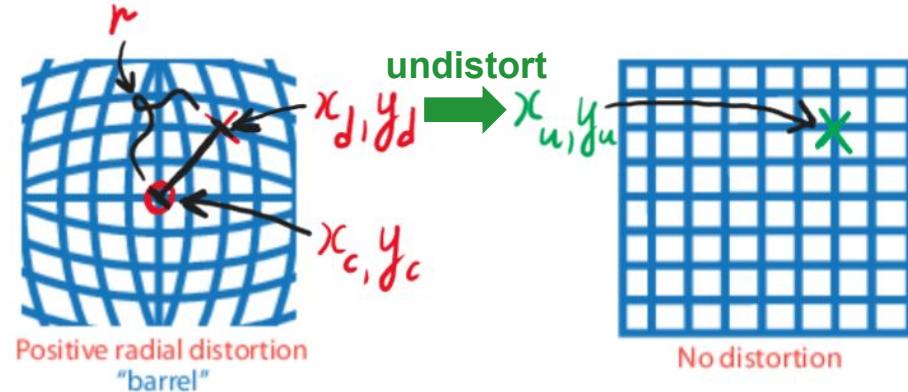
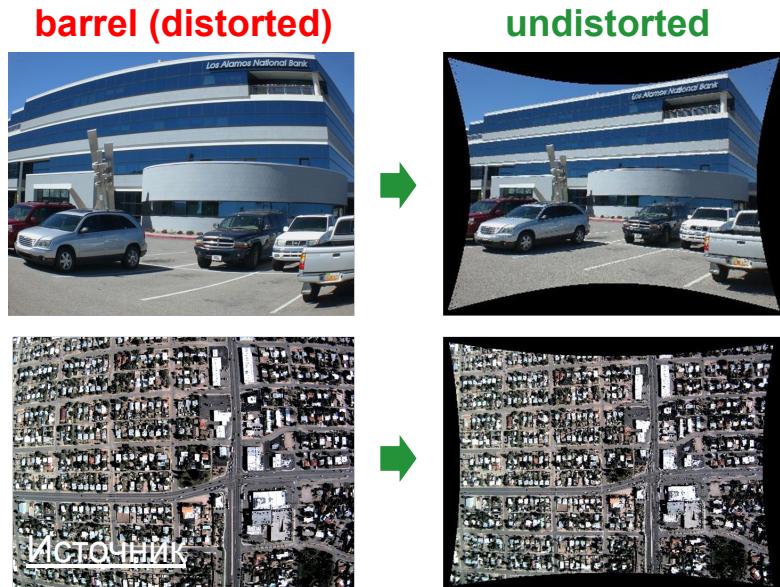


Искажение радиальной дисторсии в линзе

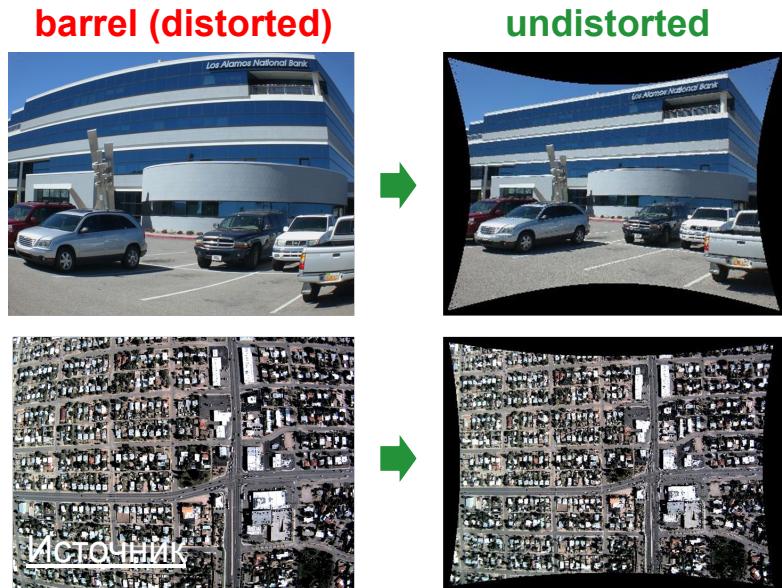


$K_1, K_2 \dots$ - коэффициенты радиальной дисторсии
 (x_c, y_c) - центр оптической оси (центр картинки)

$$r = \sqrt{(x_d - x_c)^2 + (y_d - y_c)^2}$$



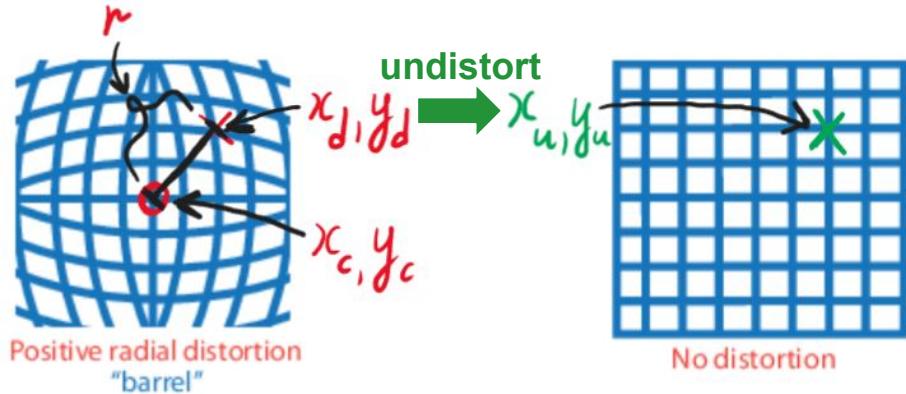
Искажение радиальной дисторсии в линзе



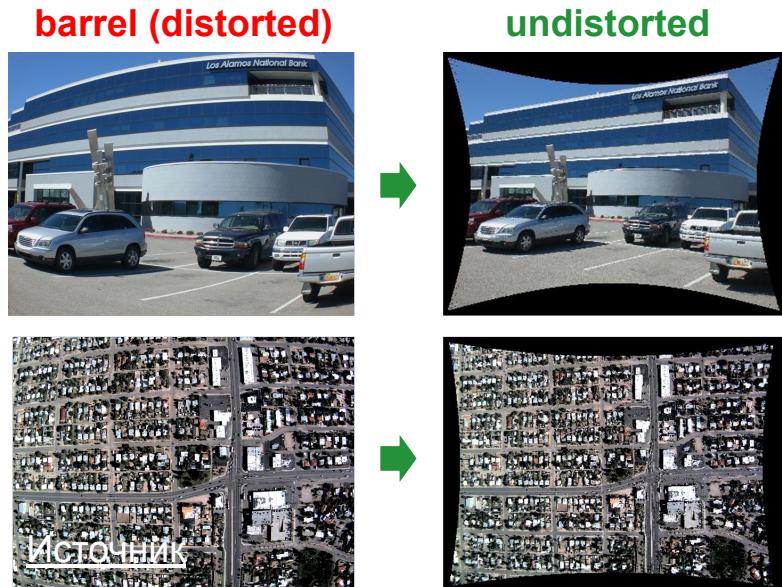
$K_1, K_2 \dots$ - коэффициенты радиальной дисторсии
 (x_c, y_c) - центр оптической оси (центр картинки)

$$r = \sqrt{(x_d - x_c)^2 + (y_d - y_c)^2}$$

$$x_u = x_d + (x_d - x_c)(K_1 r^2 + K_2 r^4 + \dots)$$



Искажение радиальной дисторсии в линзе

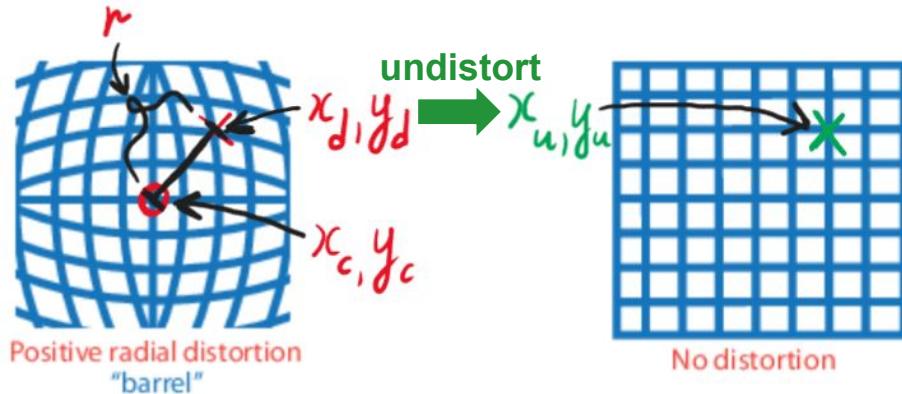


$K_1, K_2 \dots$ - коэффициенты радиальной дисторсии
(x_c, y_c) - центр оптической оси (центр картинки)

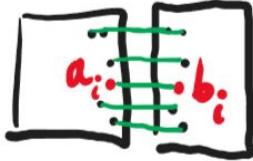
$$r = \sqrt{(x_d - x_c)^2 + (y_d - y_c)^2}$$

$$x_u = x_d + (x_d - x_c)(K_1 r^2 + K_2 r^4 + \dots)$$

$$y_u = y_d + (y_d - y_c)(K_1 r^2 + K_2 r^4 + \dots)$$



Наложение картинок с устранением искажений



$$\begin{aligned} & \rightarrow \\ & + (H a_0 - b_0)^2 \\ & + (H a_i - b_i)^2 \\ & + (H a_n - b_n)^2 \end{aligned}$$

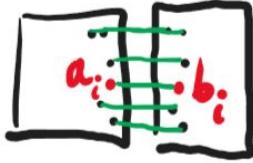
$f(\underline{H}) \rightarrow 0$

функция ошибки, невязки
loss function

сформулировав функционал невязки
через автоматическое дифференцирование
нашли точку минимума

$$H = \underset{H}{\operatorname{argmin}} f(H)$$

Наложение картинок с устранением искажений



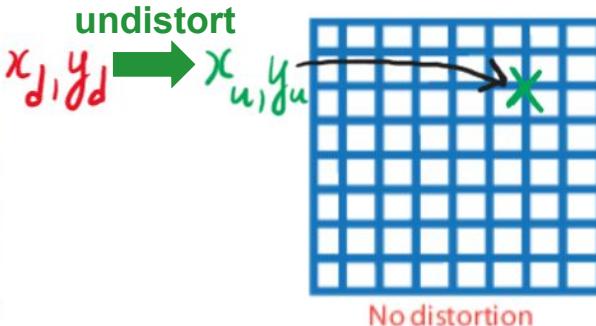
$$\begin{aligned} & \rightarrow \\ & + (H a_0 - b_0)^2 \\ & + (H a_i - b_i)^2 \\ & + (H a_n - b_n)^2 \end{aligned}$$

$f(H) \rightarrow 0$

сформулировав функционал невязки
через автоматическое дифференцирование
нашли точку минимума

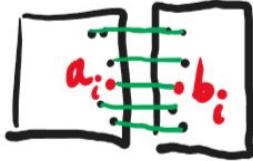
$$H = \underset{H}{\operatorname{argmin}} f(H)$$

функция ошибки, невязки
loss function



$K_1, K_2 \dots$ - коэффициенты радиальной дисторсии

Наложение картинок с устранением искажений



$$\begin{aligned} & \rightarrow \\ & + (H a_0 - b_0)^2 \\ & + (H a_i - b_i)^2 \\ & + (H a_n - b_n)^2 \end{aligned}$$

$f(H) \rightarrow 0$

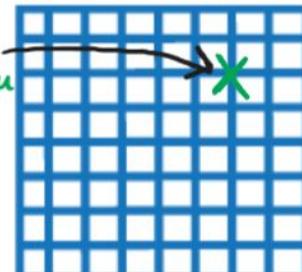
сформулировав функционал невязки
через автоматическое дифференцирование
нашли точку минимума

$$H = \underset{H}{\operatorname{argmin}} f(H)$$

функция ошибки, невязки
loss function



$$undistort \quad x_d, y_d \rightarrow$$



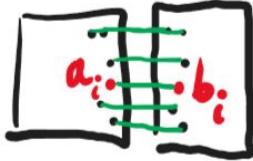
No distortion

$$x_u, y_u = g_k(x_d, y_d)$$

$K_1, K_2 \dots$ - коэффициенты радиальной дисторсии

Positive radial distortion
"barrel"

Наложение картинок с устранением искажений



$$f(H) = \sum_{i=1}^n (H a_i - b_i)^2$$

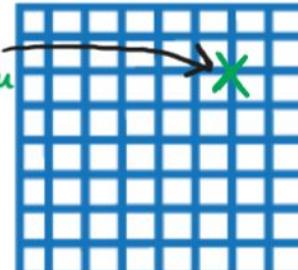
функция ошибки, невязки
loss function

сформулировав функционал невязки
через автоматическое дифференцирование
нашли точку минимума

$$H = \arg \min_H f(H)$$



undistort
 x_d, y_d



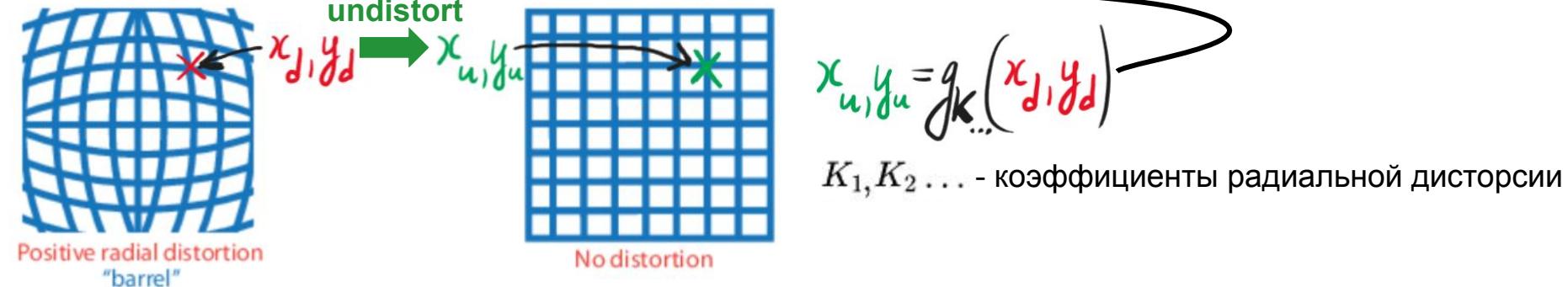
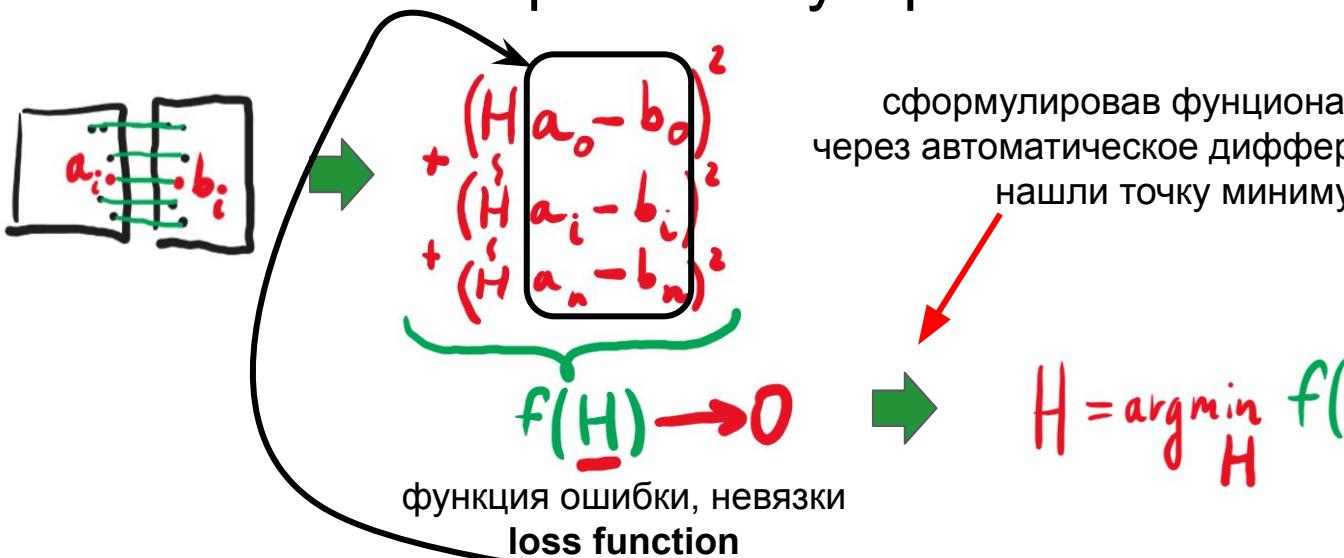
No distortion

$$x_u, y_u = g_k(x_d, y_d)$$

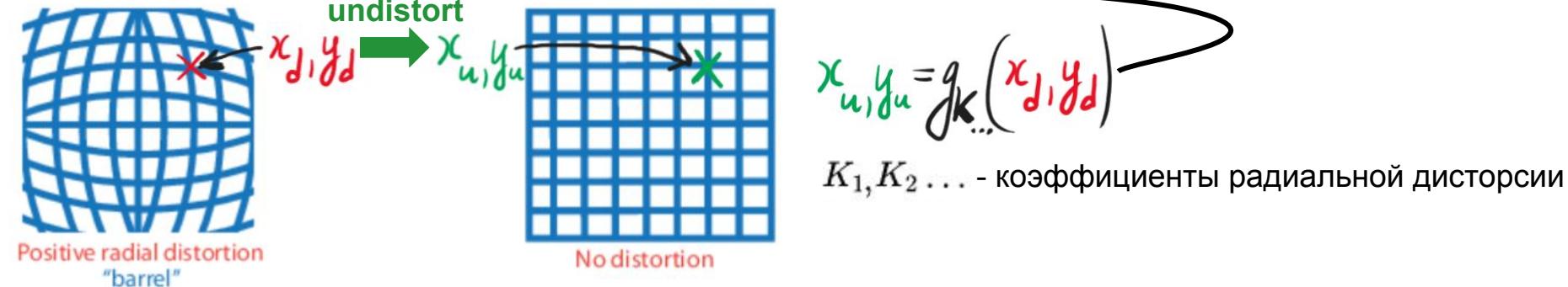
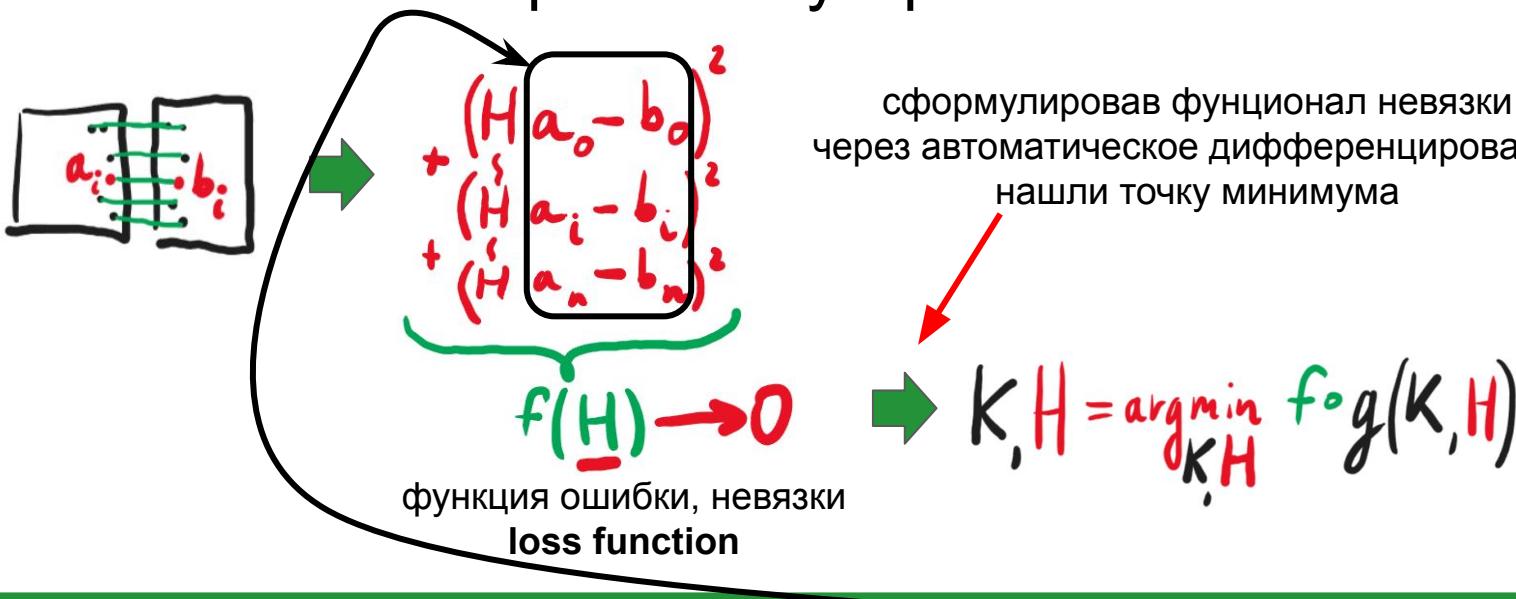
K_1, K_2, \dots - коэффициенты радиальной дисторсии

Positive radial distortion
"barrel"

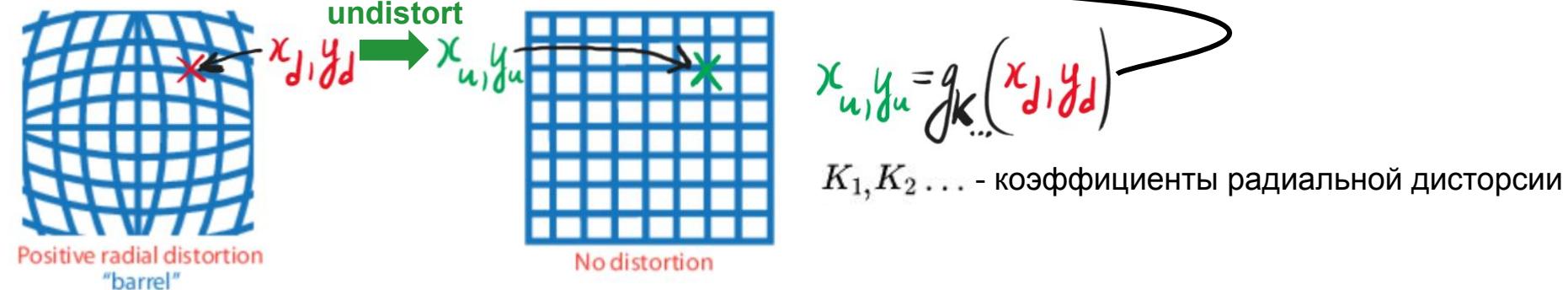
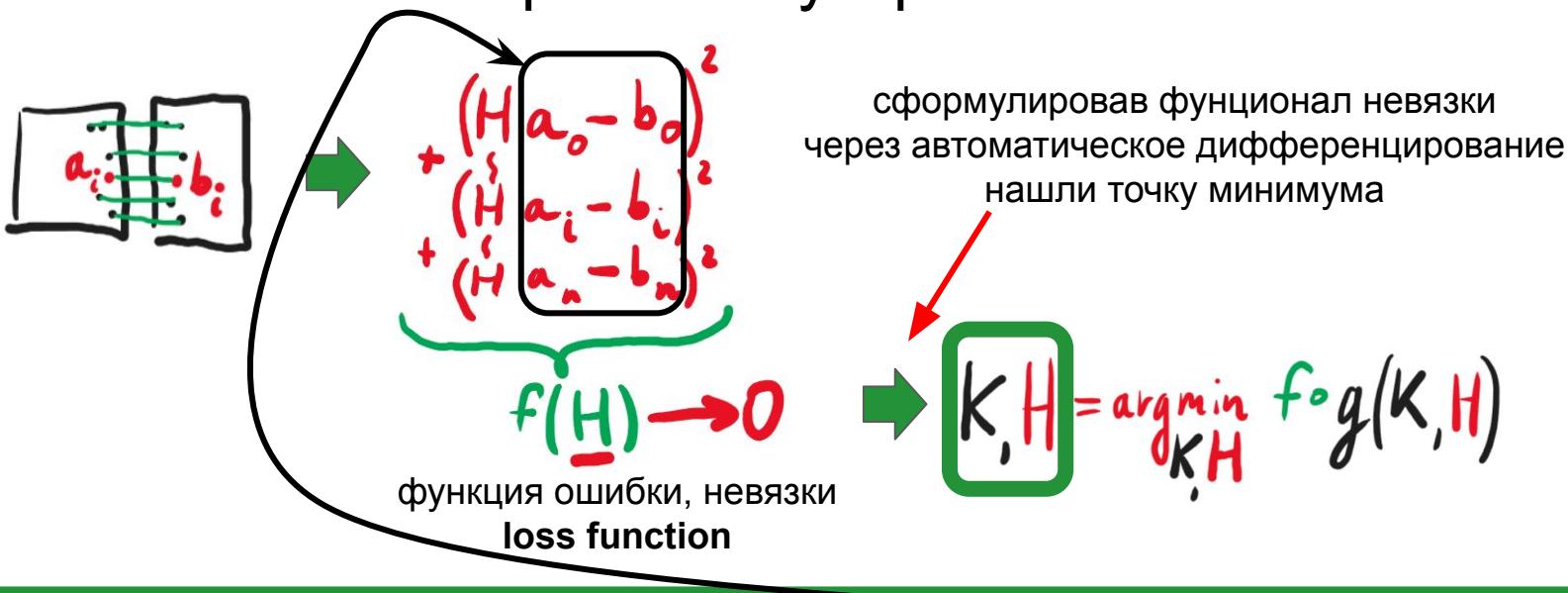
Наложение картинок с устранением искажений



Наложение картинок с устранением искажений



Наложение картинок с устранением искажений



Наложение картинок

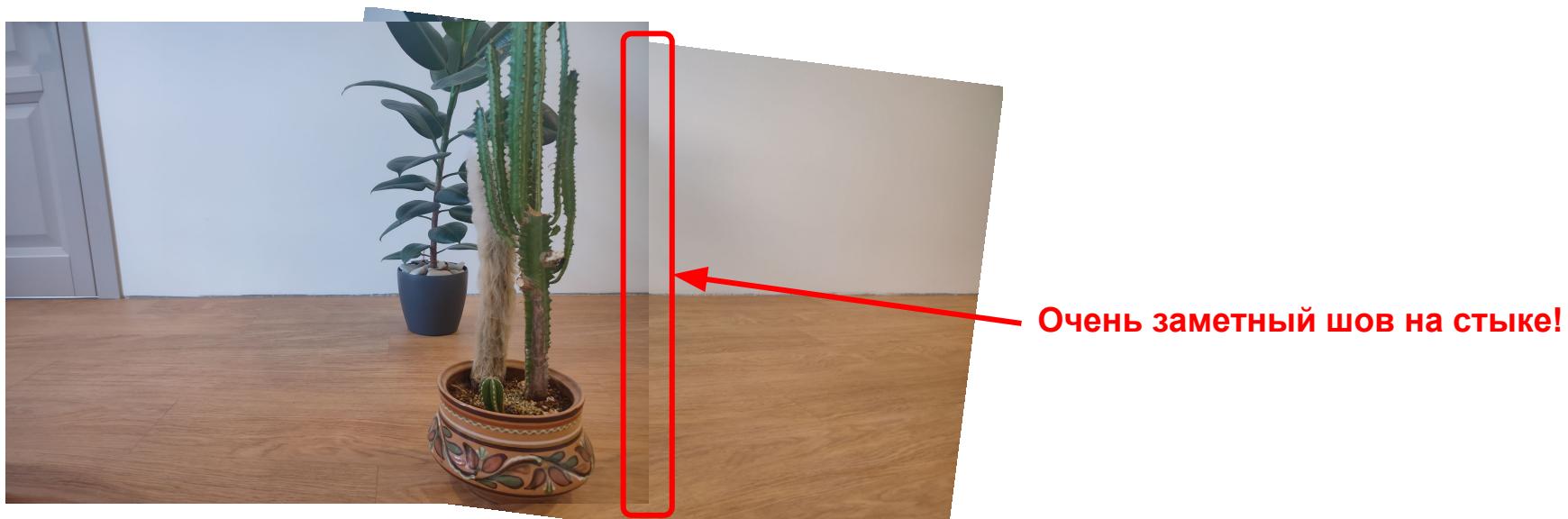
- 1) SIFT Detector + Descriptor
- 2) Nearest Neighbor matching + K-Ratio test + Left-Right check + Cluster filtering + **RANSAC** для поиска **Homography** матрицы:
 - 2.1) много раз берем случайные четыре сопоставления:
 - 2.2) поверив в их переход строим **Homography** матрицу (DLT)
 - 2.3) ищем число поддерживающих эту матрицу (число корректных - **inliers**)
 - 2.4) матрица с наибольшей поддержкой (наибольшим числом **inliers**) - верна
- 3) По **inliers** давайте заново найдем Homography матрицу **H** через Ceres Solver (на этот раз минимизируя общую ошибку).

Наложение картинок + устранение искажений дисторсии

- 1) SIFT Detector + Descriptor
- 2) Nearest Neighbor matching + K-Ratio test + Left-Right check + Cluster filtering + **RANSAC** для поиска **Homography** матрицы:
 - 2.1) много раз берем случайные четыре сопоставления:
 - 2.2) поверив в их переход строим **Homography** матрицу (DLT)
 - 2.3) ищем число поддерживающих эту матрицу (число корректных - **inliers**)
 - 2.4) матрица с наибольшей поддержкой (наибольшим числом **inliers**) - верна
- 3) По **inliers** давайте заново найдем Homography матрицу **H** через Ceres Solver (на этот раз минимизируя общую ошибку).
При этом ищем еще и коэффициенты дисторсии **K** - т.к. в функции невязки применяется undistort-формула к точкам (перед применением матрицы **H**).

Устранение шва

Теперь все фотографии сопоставлены, но:





<https://www.sensefly.com/education/datasets/>



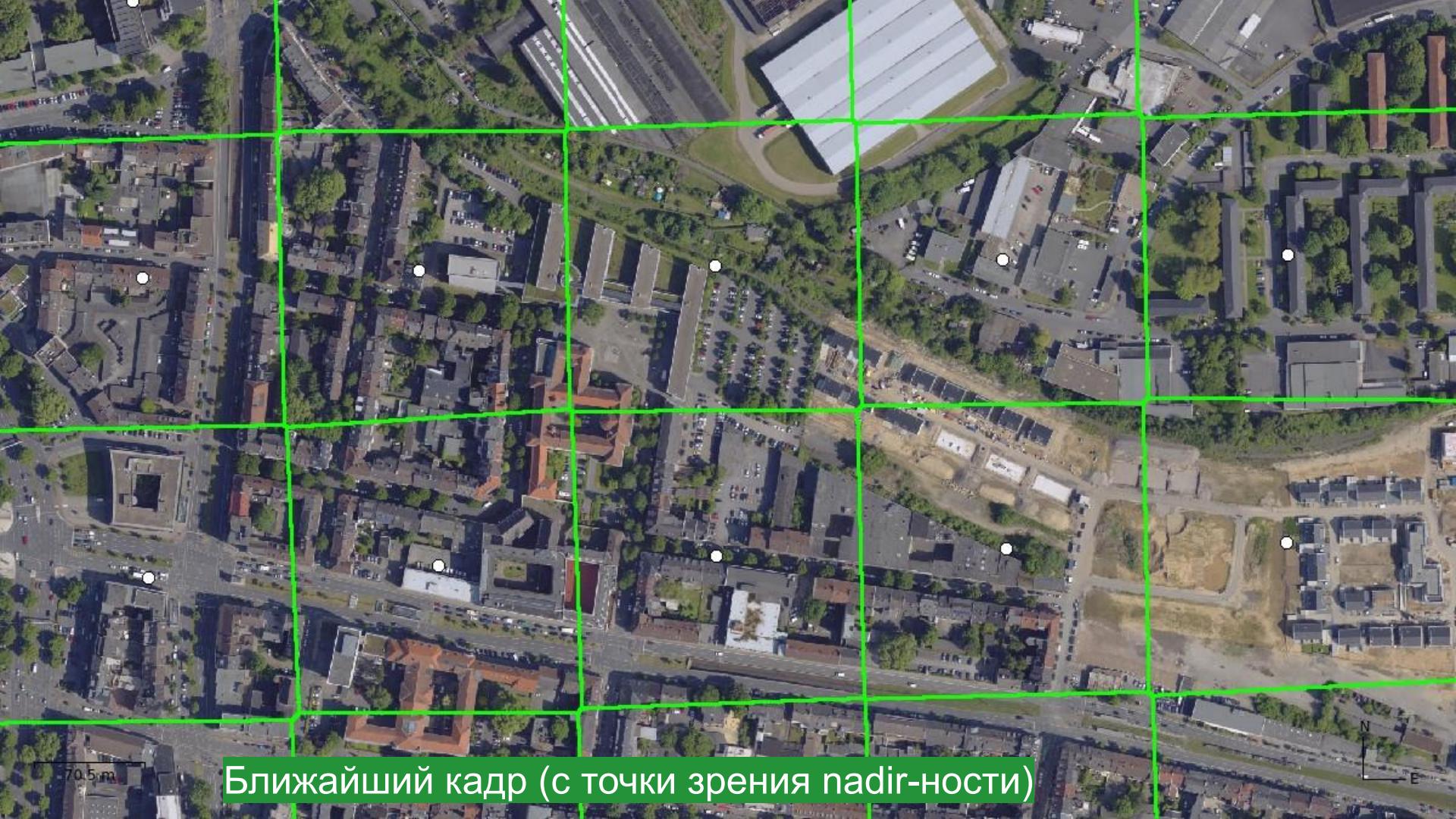
<https://www.sensefly.com/education/datasets/>

Усреднение по всем кадрам

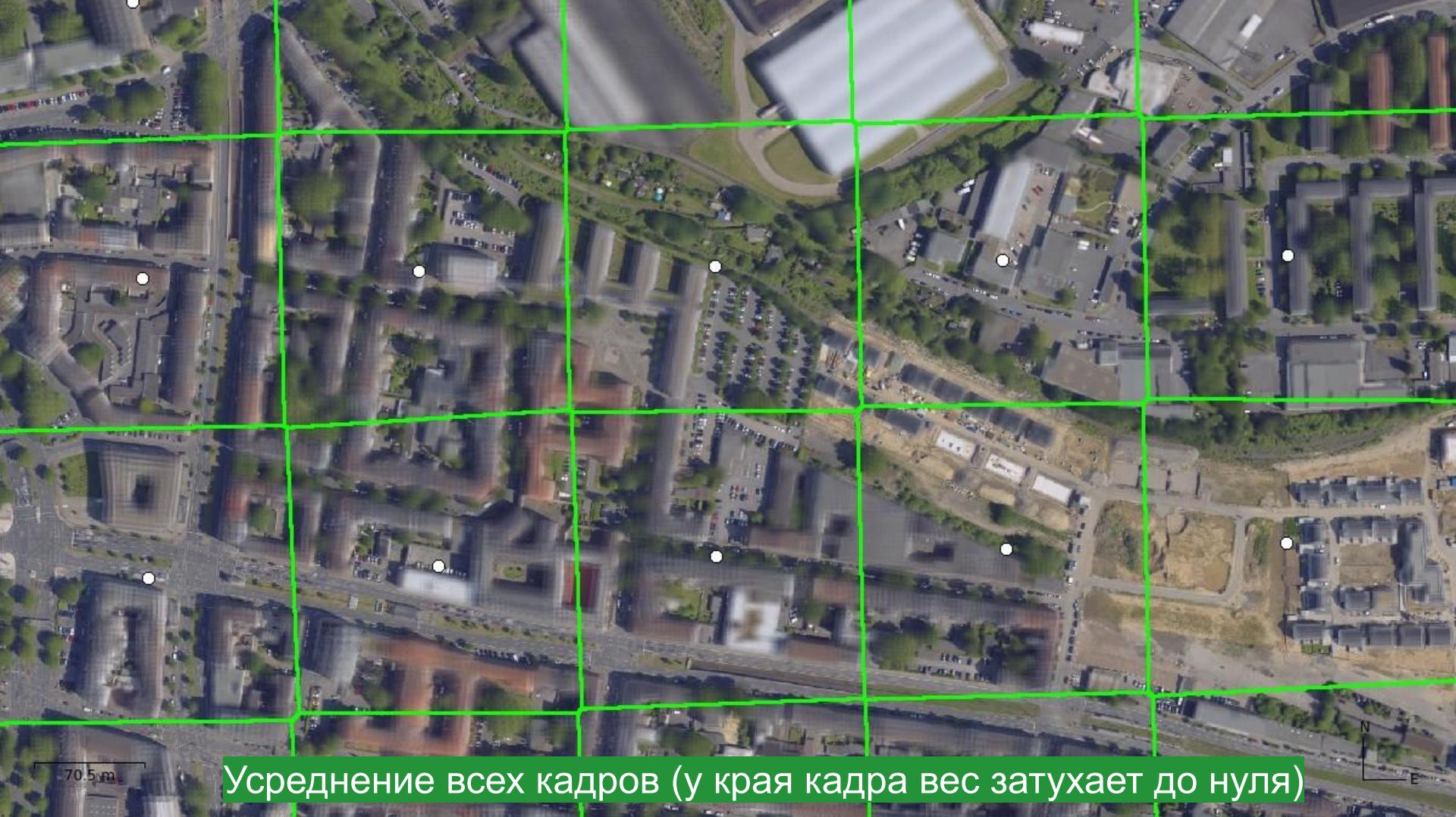


Отдельный кадр



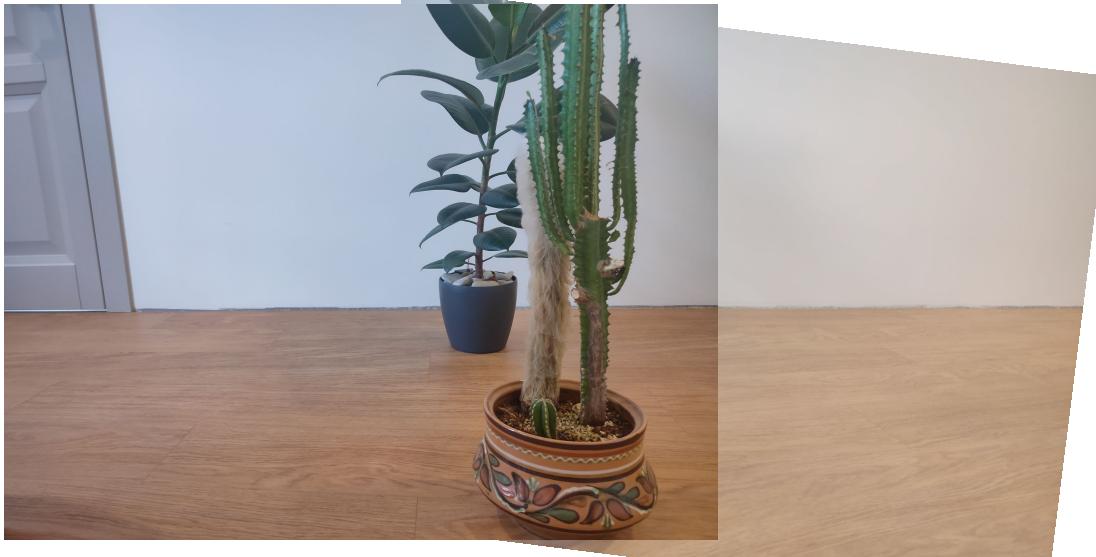


Ближайший кадр (с точки зрения nadir-ности)

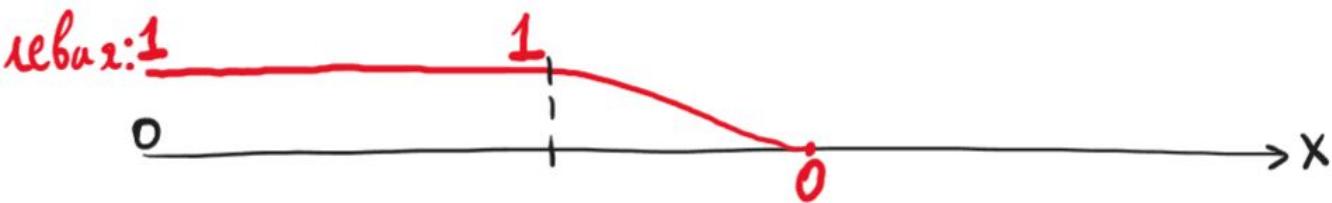
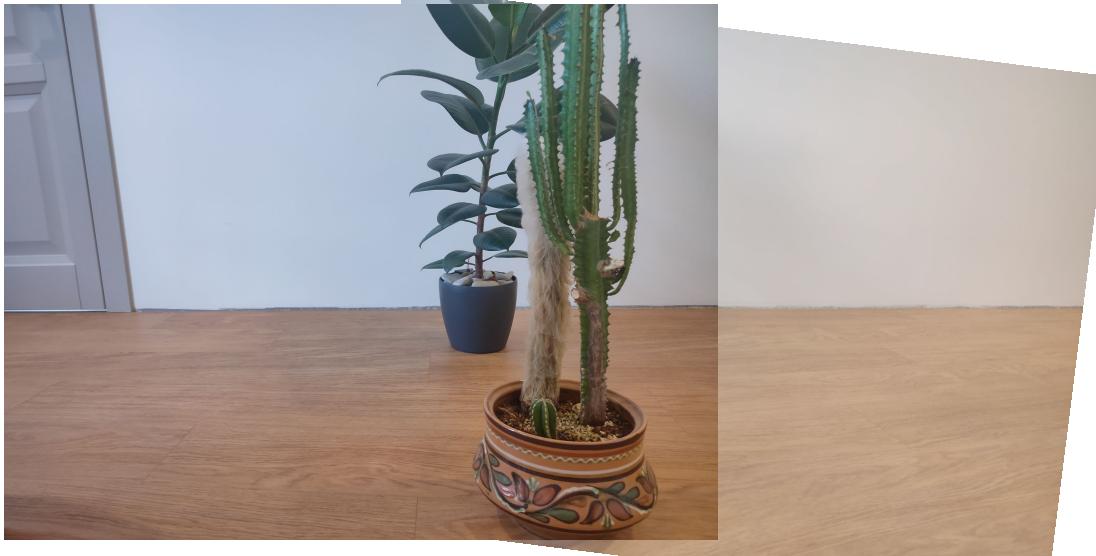


Усреднение всех кадров (у края кадра вес затухает до нуля)

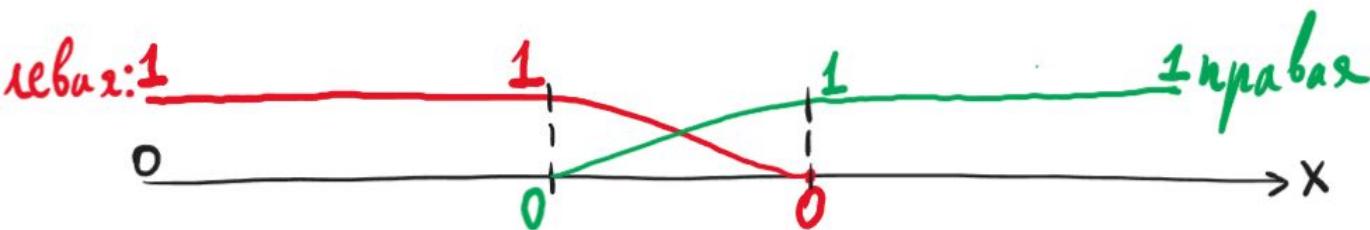
Устранение шва



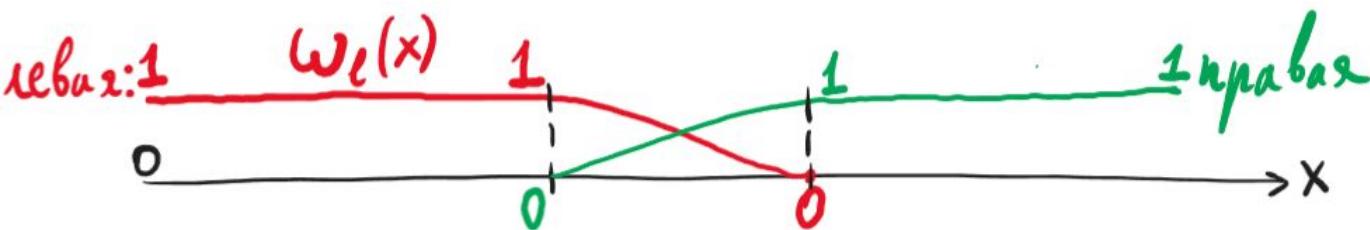
Устранение шва



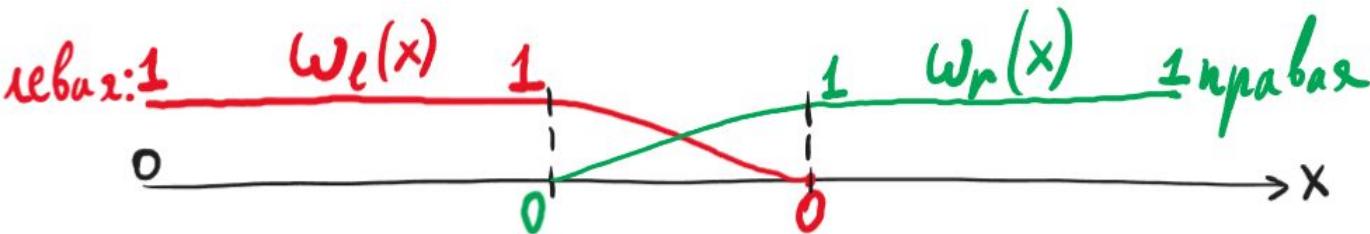
Устранение шва



Устранение шва



Устранение шва

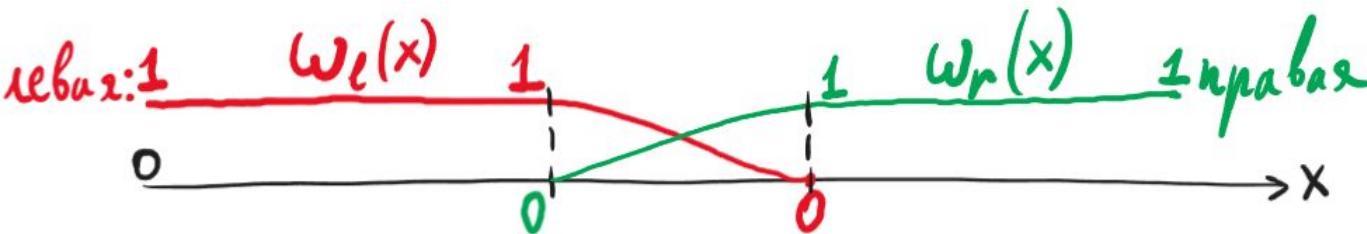


Устранение шва



$$I_l(x)$$

$$I_r(x)$$



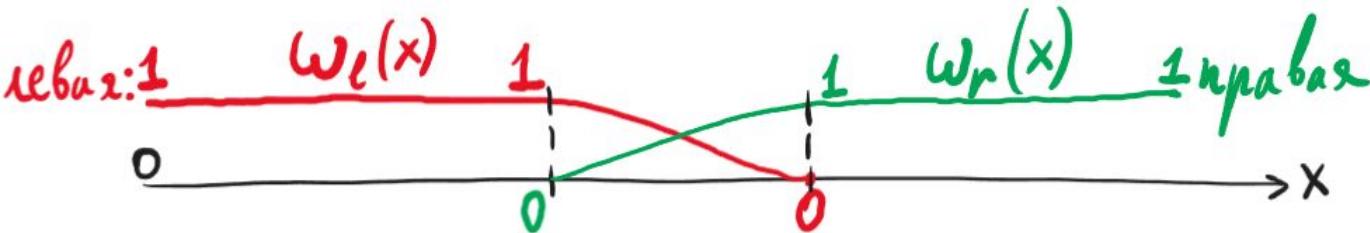
Устранение шва



$$I_l(x)$$

$$I_r(x)$$

$$P_{ano}(x) = I_l(x) \cdot w_l(x) + I_r(x) \cdot w_r(x)$$



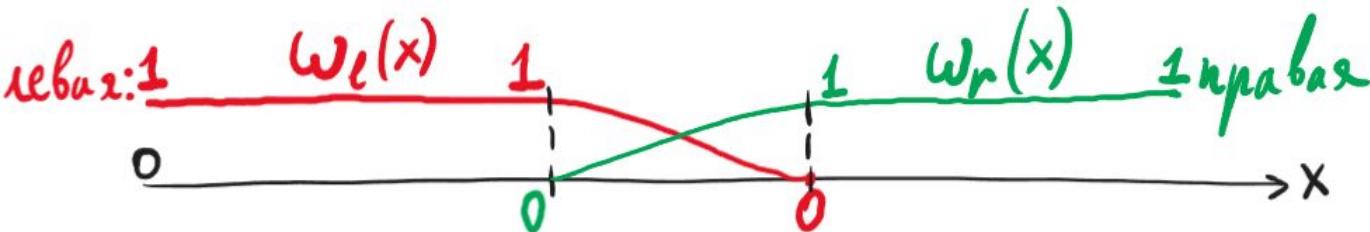
Устранение шва



$$I_l(x)$$

$$I_r(x)$$

$$P_{ano}(x) = \frac{I_l(x) \cdot w_l(x) + I_r(x) \cdot w_r(x)}{w_l(x) + w_r(x)}$$



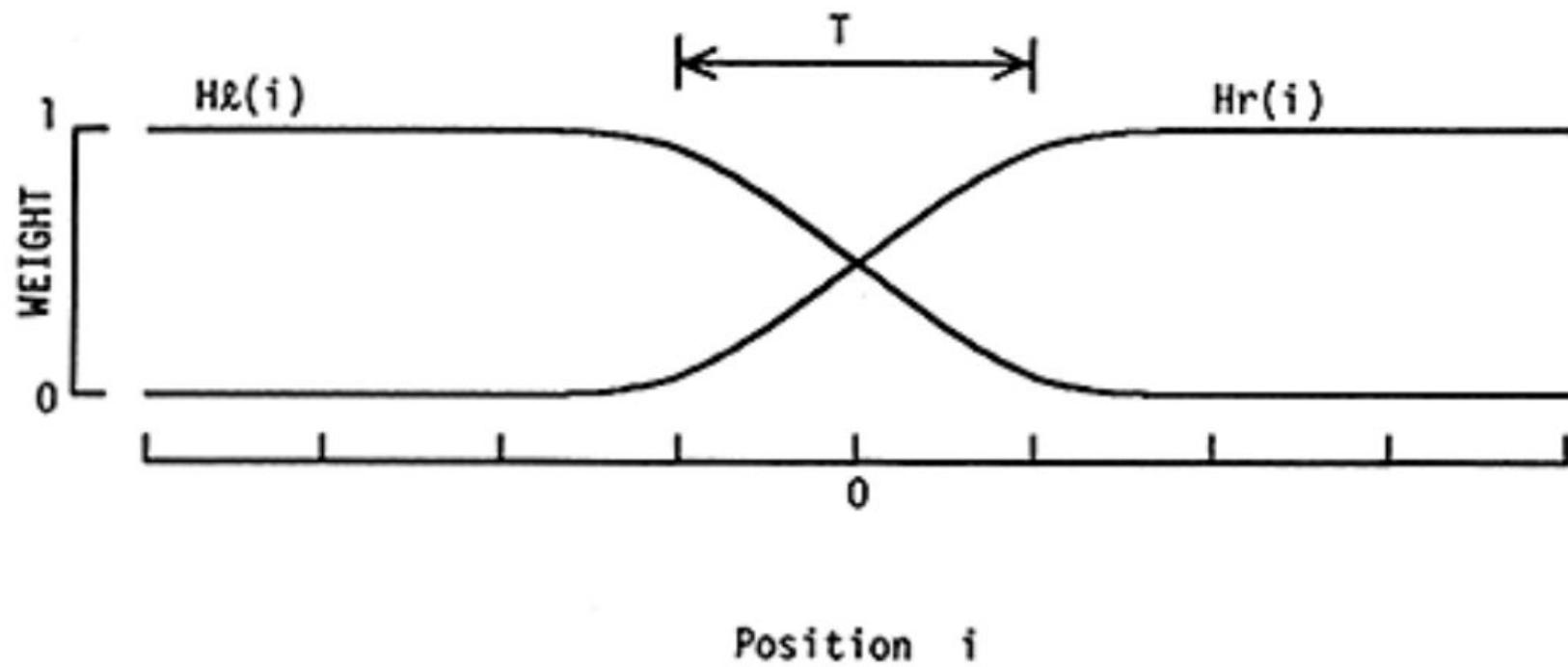


Fig. 2. The weighted average method may be used to avoid seams when mosaics are constructed from overlapped images. Each image is multiplied by a weighting function which decreases monotonically across its border; the resulting images are then summed to form the mosaic. Example weighting functions are shown here in one dimension. The width of the transition zone T is a critical parameter for this method.

[A Multiresolution Spline With Application to Image Mosaics, Burt et al., 1983](#)

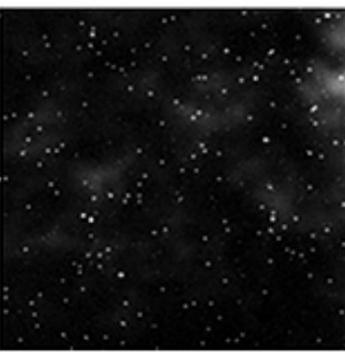
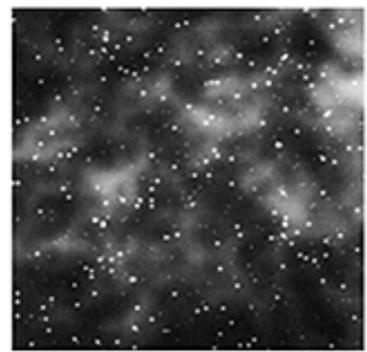


Fig. 3. Common artifacts of the weighted average techniques are demonstrated in these attempts to spline two synthetic images of stars (Figure 3a and 3b). These differ only in **mean gray level** and a **slight vertical shift**.

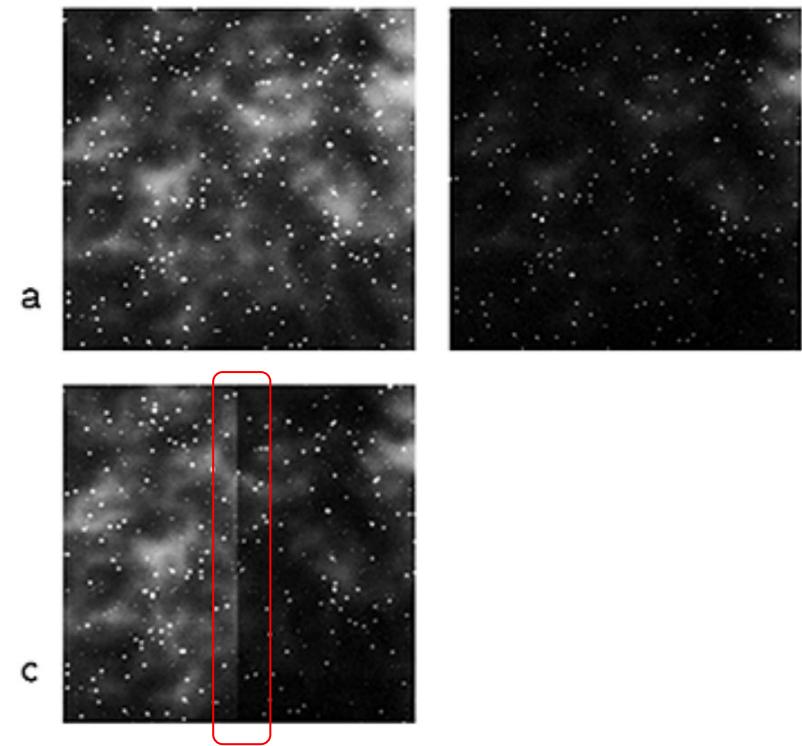
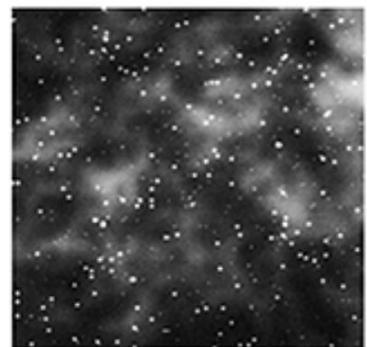
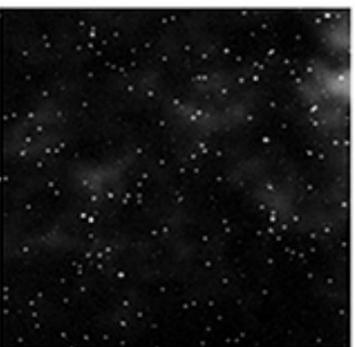


Fig. 3. Common artifacts of the weighted average techniques are demonstrated in these attempts to spline two synthetic images of stars (Figure 3a and 3b). These differ only in **mean gray level** and a **slight vertical shift**.

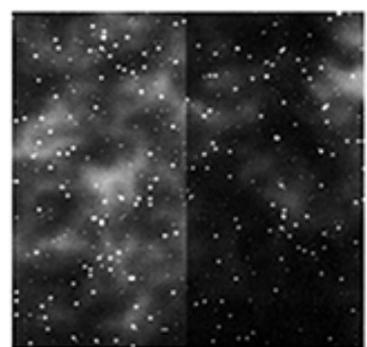
b A seam is clearly visible when the left half of figure 3a is joined with the right half of Figure 3b without any adjustment in gray level, as shown in Figure 3c.



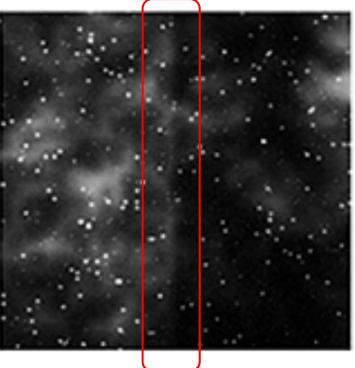
a



b



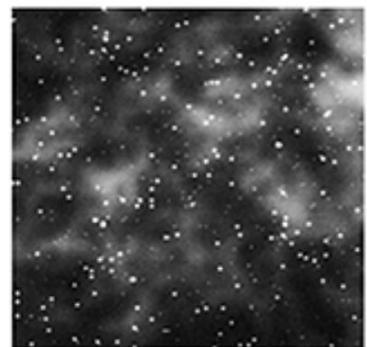
c



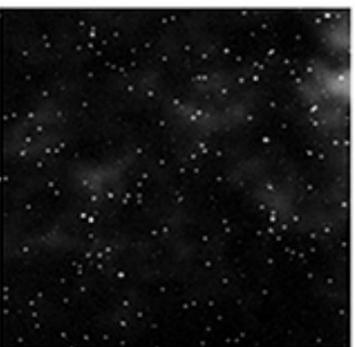
d

Fig. 3. Common artifacts of the weighted average techniques are demonstrated in these attempts to spline two synthetic images of stars (Figure 3a and 3b). These differ only in **mean gray level** and a **slight vertical shift**.

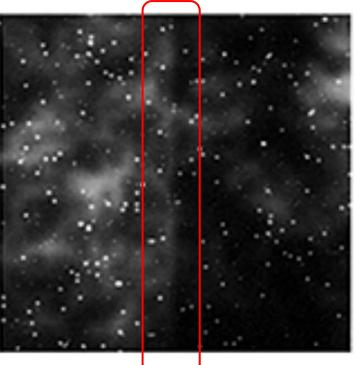
A seam is clearly visible when the left half of figure 3a is joined with the right half of Figure 3b without any adjustment in gray level, as shown in Figure 3c. The seam is **still visible when the weighted average technique is used with a narrow transition zone (Figure 3d)**.



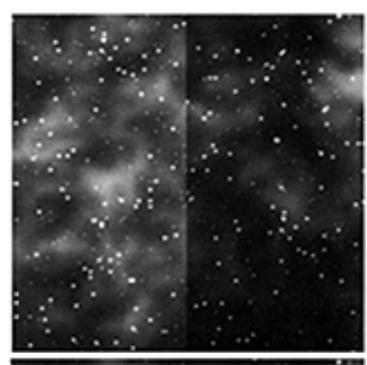
a



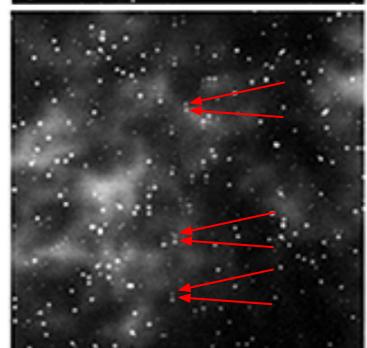
b



c



d



e

Fig. 3. Common artifacts of the weighted average techniques are demonstrated in these attempts to spline two synthetic images of stars (Figure 3a and 3b). These differ only in **mean gray level** and a **slight vertical shift**.

b A seam is clearly visible when the left half of figure 3a is joined with the right half of Figure 3b without any adjustment in gray level, as shown in Figure 3c. The seam is **still visible** when the **weighted average technique is used with a narrow transition zone** (Figure 3d). However, if the **transition zone is wide**, features within the zone appear double (Figure 3e).

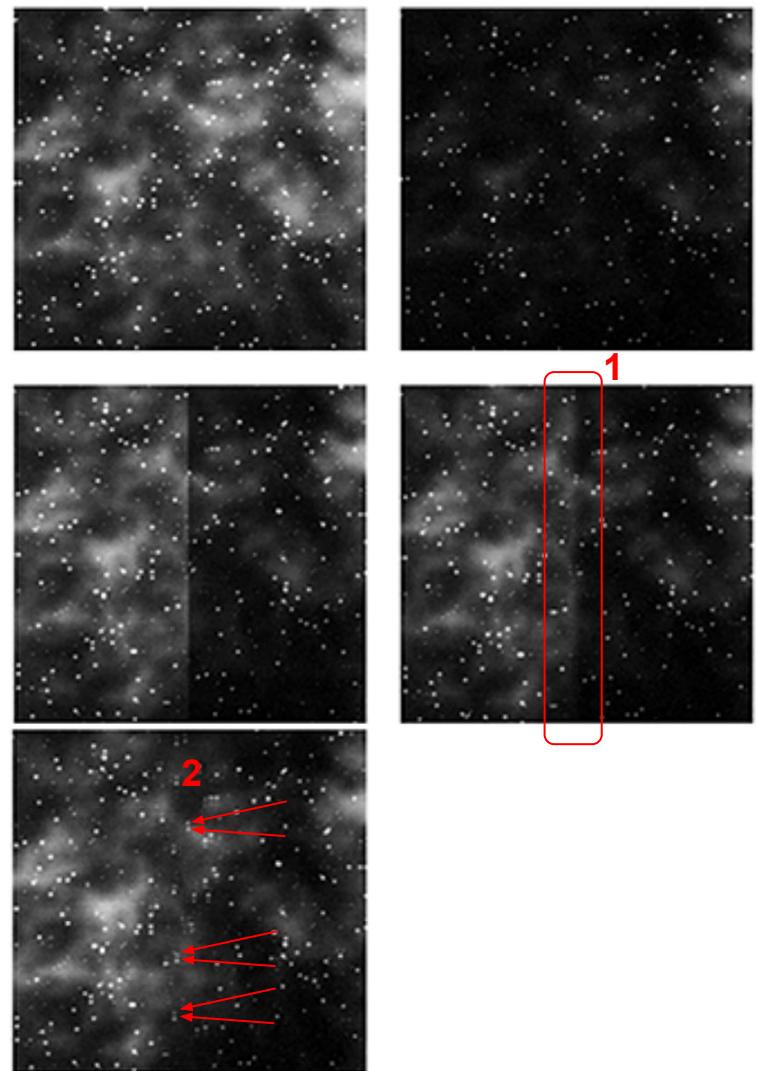
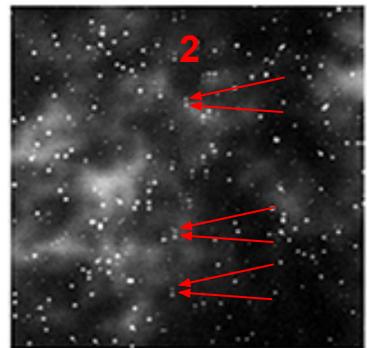
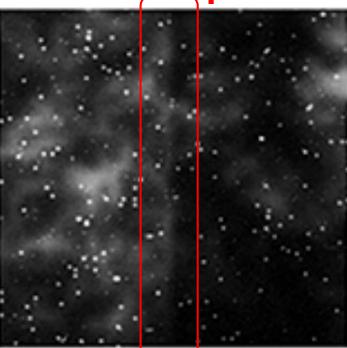
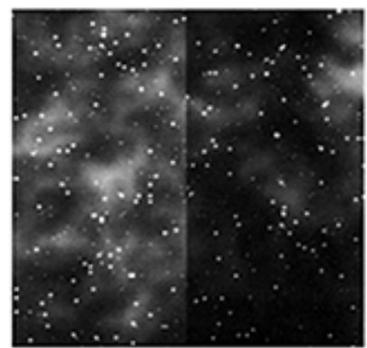
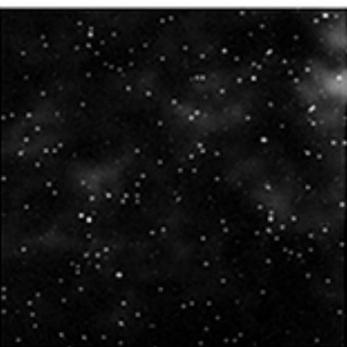
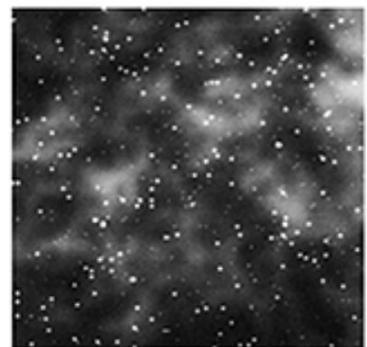


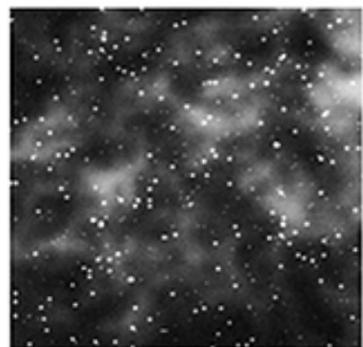
Fig. 3. Common artifacts of the weighted average techniques are demonstrated in these attempts to spline two synthetic images of stars (Figure 3a and 3b). These differ only in **mean gray level** and a **slight vertical shift**.

b A seam is clearly visible when the left half of figure 3a is joined with the right half of Figure 3b without any adjustment in gray level, as shown in Figure 3c. The seam is **still visible when the weighted average technique is used with a narrow transition zone (Figure 3d)**. However, if the transition zone is wide, features within the zone appear double (Figure 3e).

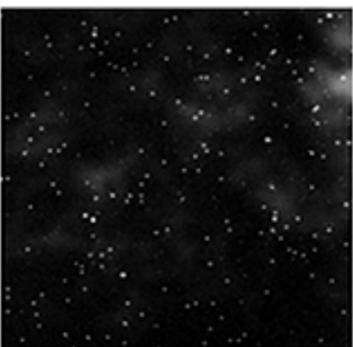
The first of these artifacts is due to a gray level mismatch at low spatial frequencies, while the second is **due to a position mismatch at high frequencies**. Both are avoided in the multiresolution method (Figure 3f).



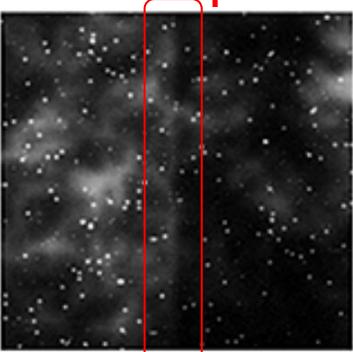
1. Чтобы подавить “широкие/большие” проблемы - перепады яркости - нужна широкая зона перехода весов.



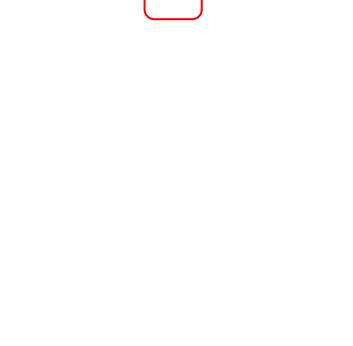
a



b



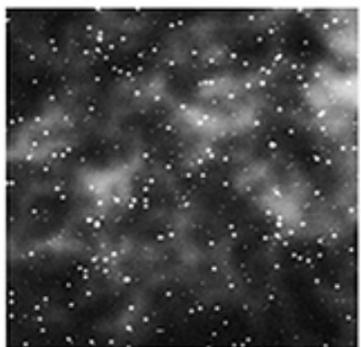
d



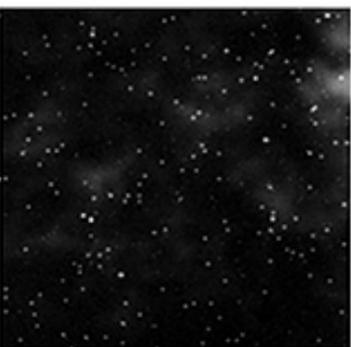
e

1. Чтобы подавить “широкие/большие” проблемы - перепады яркости - нужна **широкая зона** перехода весов.

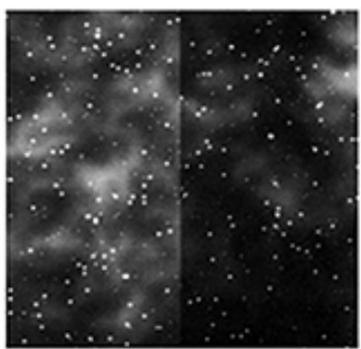
2. Чтобы не двоилось - нужно чтобы “**маленькие**” объекты - звезды - обычно брались ровно с одного кадра - например это так если мы используем **узкую зону** перехода весов.



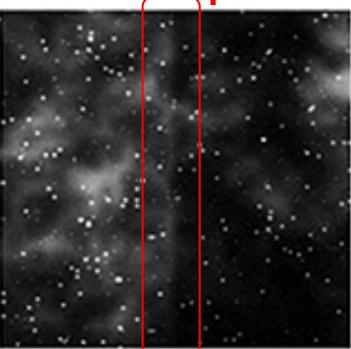
a



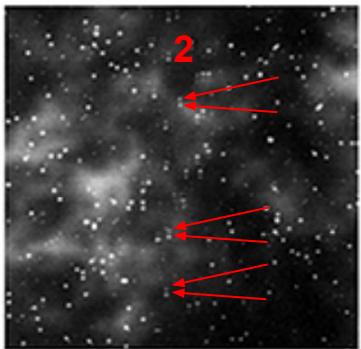
b



c



d



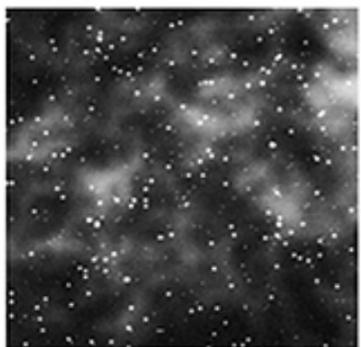
e

1. Чтобы подавить “широкие/большие” проблемы - перепады яркости - нужна **широкая зона** перехода весов.

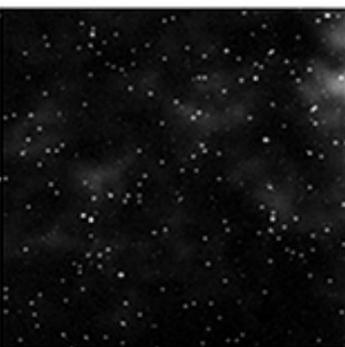
2. Чтобы не двоилось - нужно чтобы “**маленькие**” объекты - звезды - обычно брались ровно с одного кадра - например это так если мы используем **узкую зону** перехода весов.

Как же взять лучшее от двух миров?

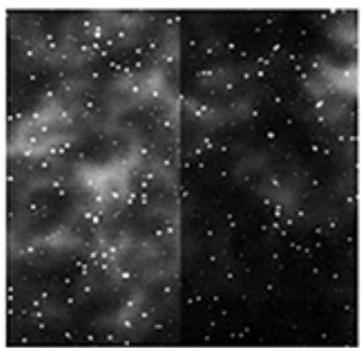
[A Multiresolution Spline With Application to Image Mosaics. Burt et al., 1983](#)



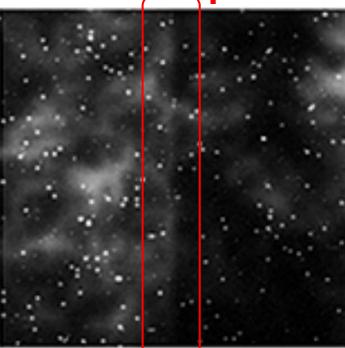
a



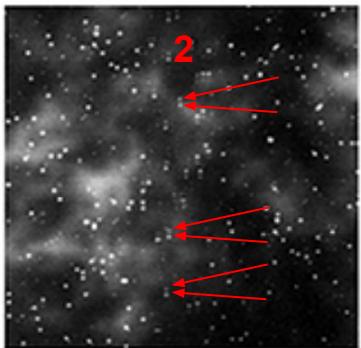
b



c



d



e

f

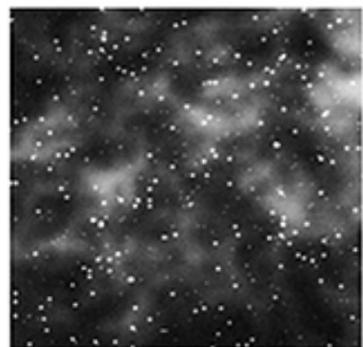
1. Чтобы подавить “широкие/большие” проблемы - перепады яркости - нужна **широкая зона** перехода весов.

2. Чтобы не двоилось - нужно чтобы “маленькие” объекты - звезды - обычно брались ровно с одного кадра - например это так если мы используем **узкую зону** перехода весов.

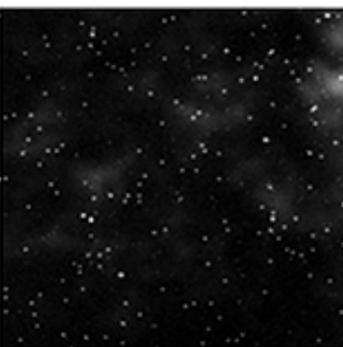
Как же взять лучшее от двух миров?

Суметь разбить картинку на разные частоты. И отдельно обрабатывать разные по масштабу объекты.

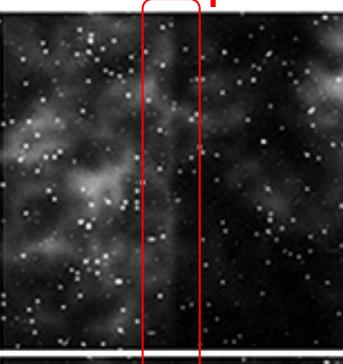
[A Multiresolution Spline With Application to Image Mosaics. Burt et al., 1983](#)



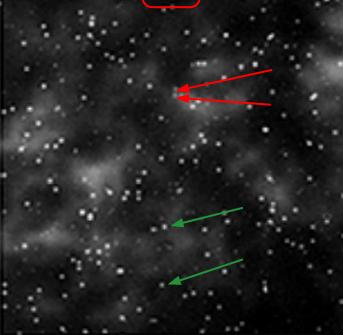
a



b



d



f

1. Чтобы подавить “широкие/большие” проблемы - перепады яркости - нужна **широкая зона** перехода весов.

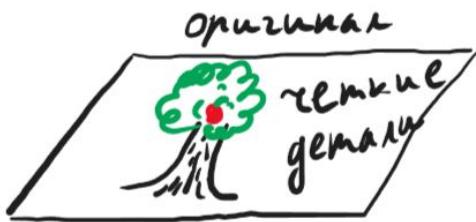
2. Чтобы не двоилось - нужно чтобы “маленькие” объекты - звезды - обычно брались ровно с одного кадра - например это так если мы используем **узкую зону** перехода весов.

Как же взять лучшее от двух миров?

Суметь разбить картинку на разные частоты. И отдельно обрабатывать разные по масштабу объекты.

[A Multiresolution Spline With Application to Image Mosaics. Burt et al., 1983](#)

Разложим картинку на частоты



Разложим картинку на частоты

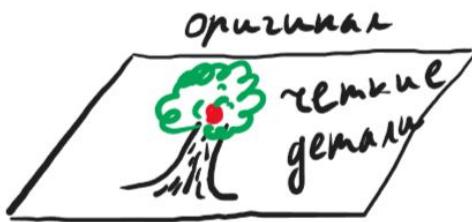


размытие (blur)

размытые
границы

A hand-drawn illustration of the same tree, but with a blurry effect applied to the edges. The drawing is enclosed in a trapezoidal frame. The text "размытые границы" is written to the right of the tree.

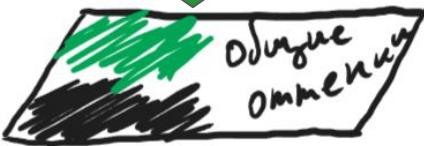
Разложим картинку на частоты



размытие (blur)



размытие (blur)



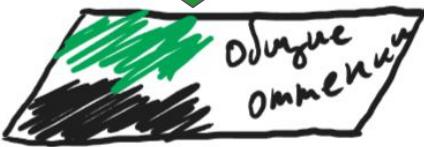
Разложим картинку на частоты



размытие (blur)



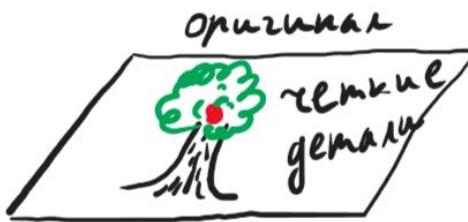
размытие (blur)



размытие (blur)

что будет на **самом последнем уровне???**

Разложим картинку на частоты



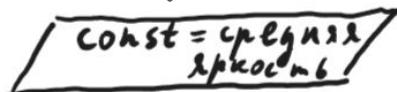
размытие (blur)



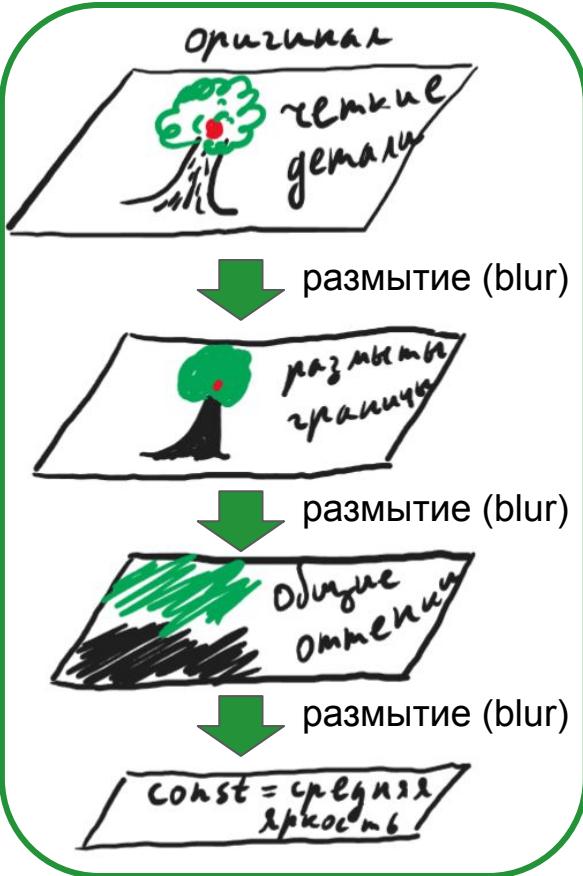
размытие (blur)



размытие (blur)

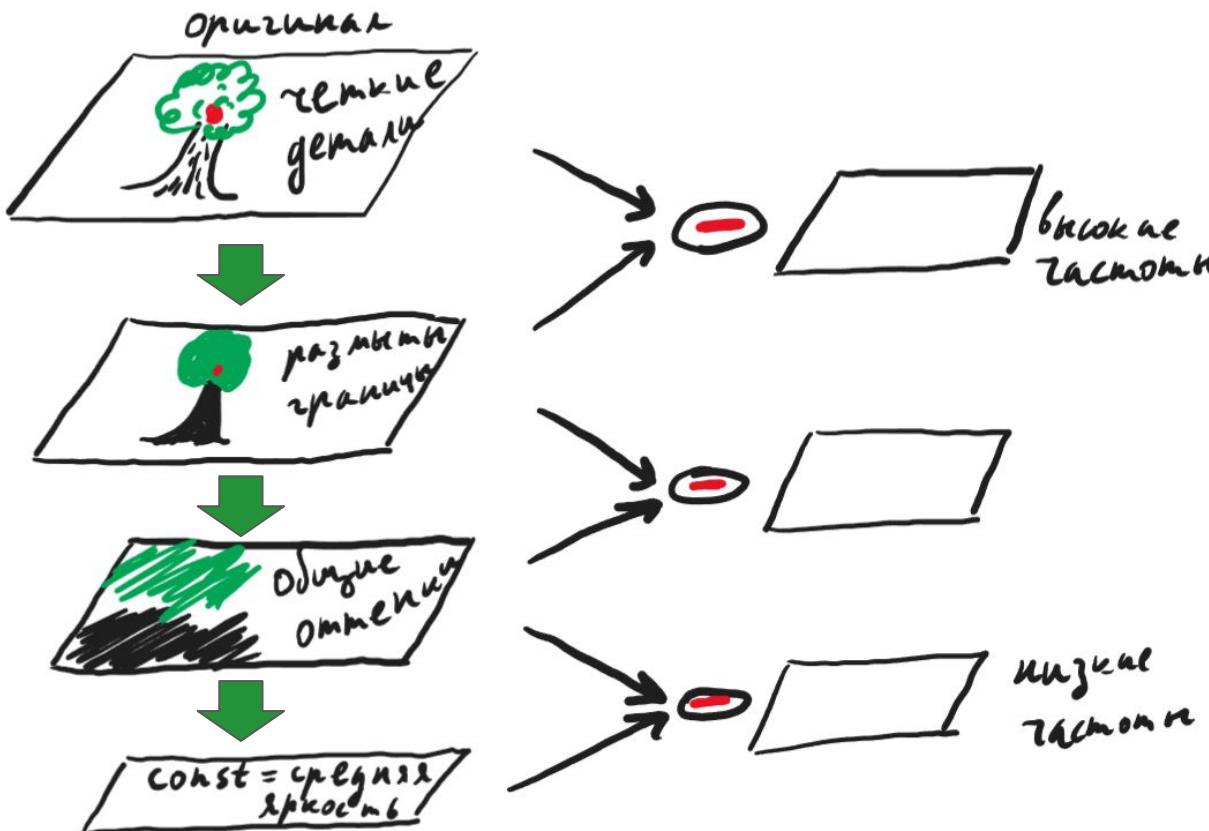


Разложим картинку на частоты



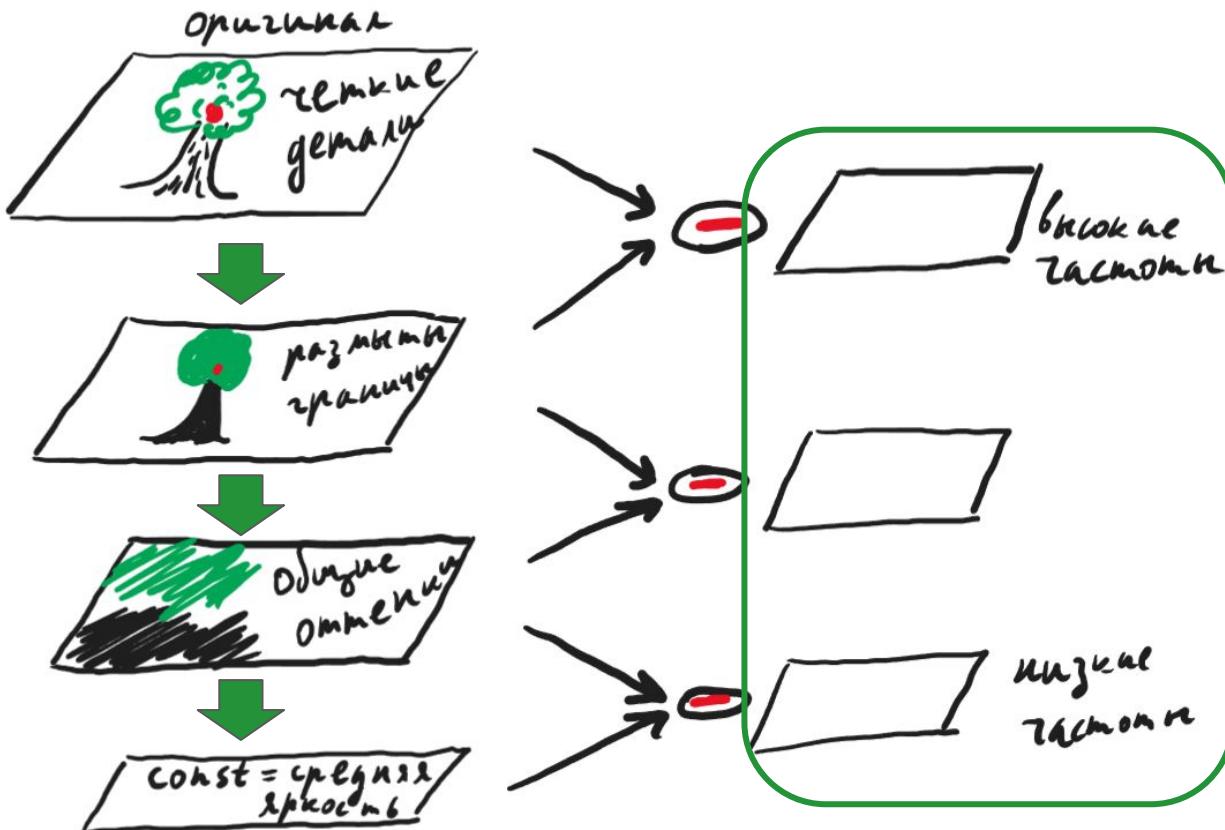
Gaussian Pyramid
(т.к. гауссово размытие)

Разложим картинку на частоты



Gaussian Pyramid

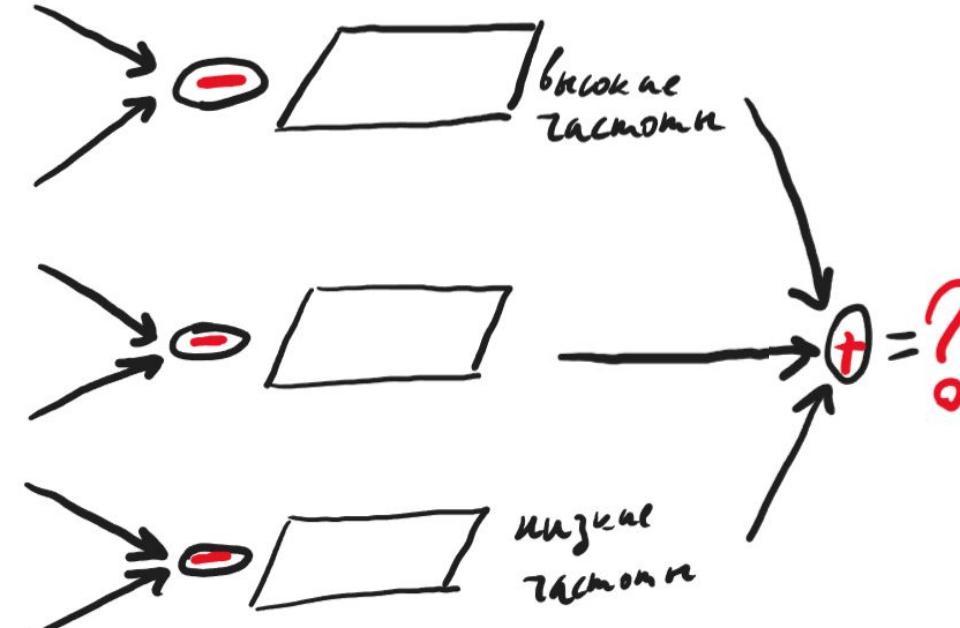
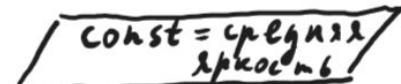
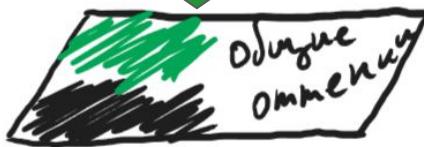
Разложим картинку на частоты



Difference of Gaussian
(DoG)

Gaussian Pyramid

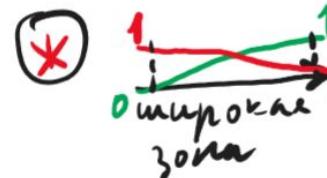
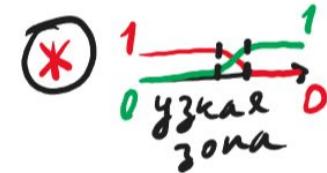
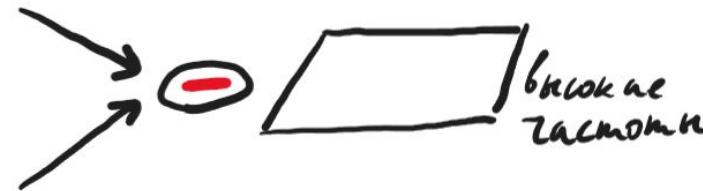
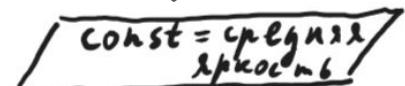
Разложим картинку на частоты



Gaussian Pyramid

Difference of Gaussian
(DoG)

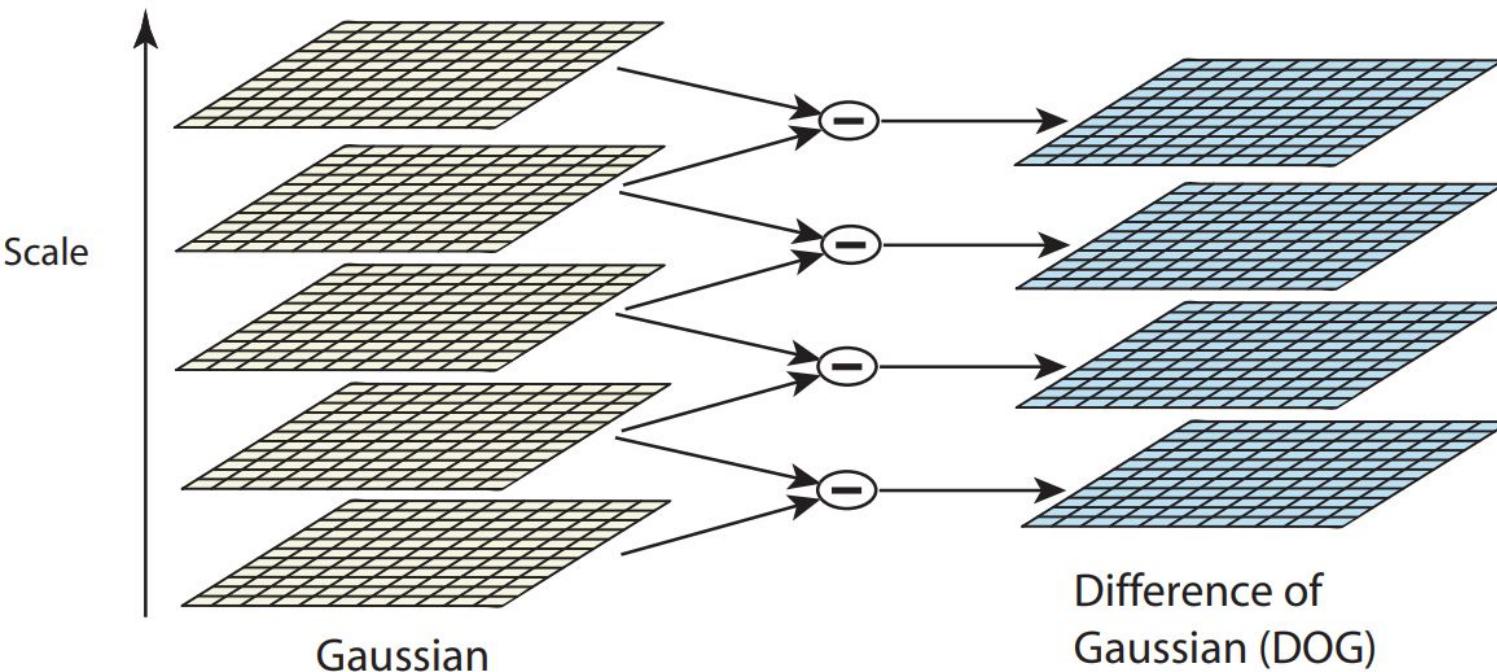
Разложим картинку на частоты



Gaussian Pyramid

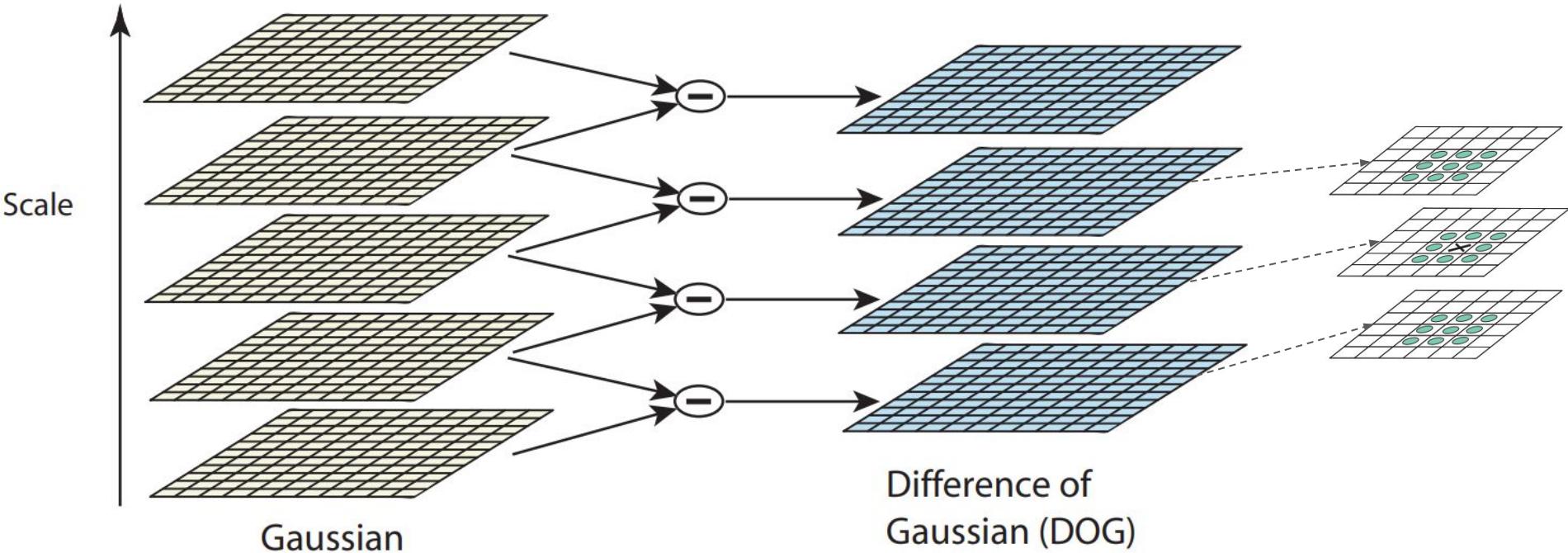
Difference of Gaussian
(DoG)

Разложим картинку на частоты



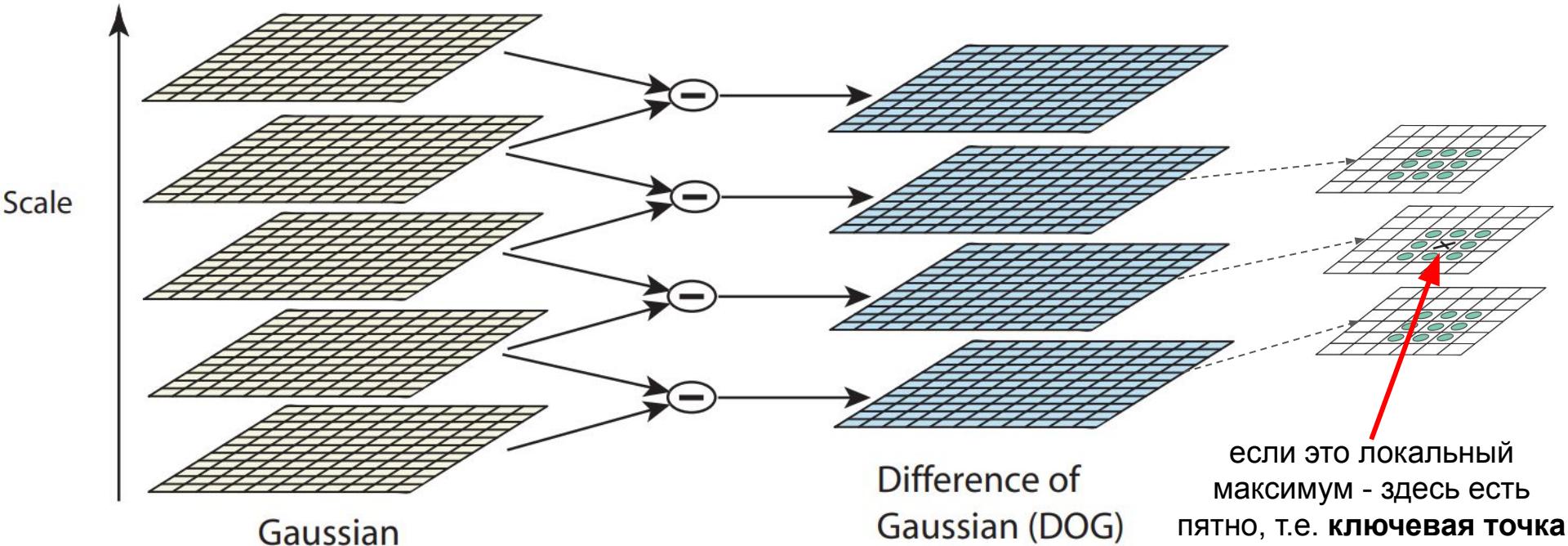
Вот такая иллюстрация есть в статье про SIFT Detector... Что она там делает?

Разложим картинку на частоты



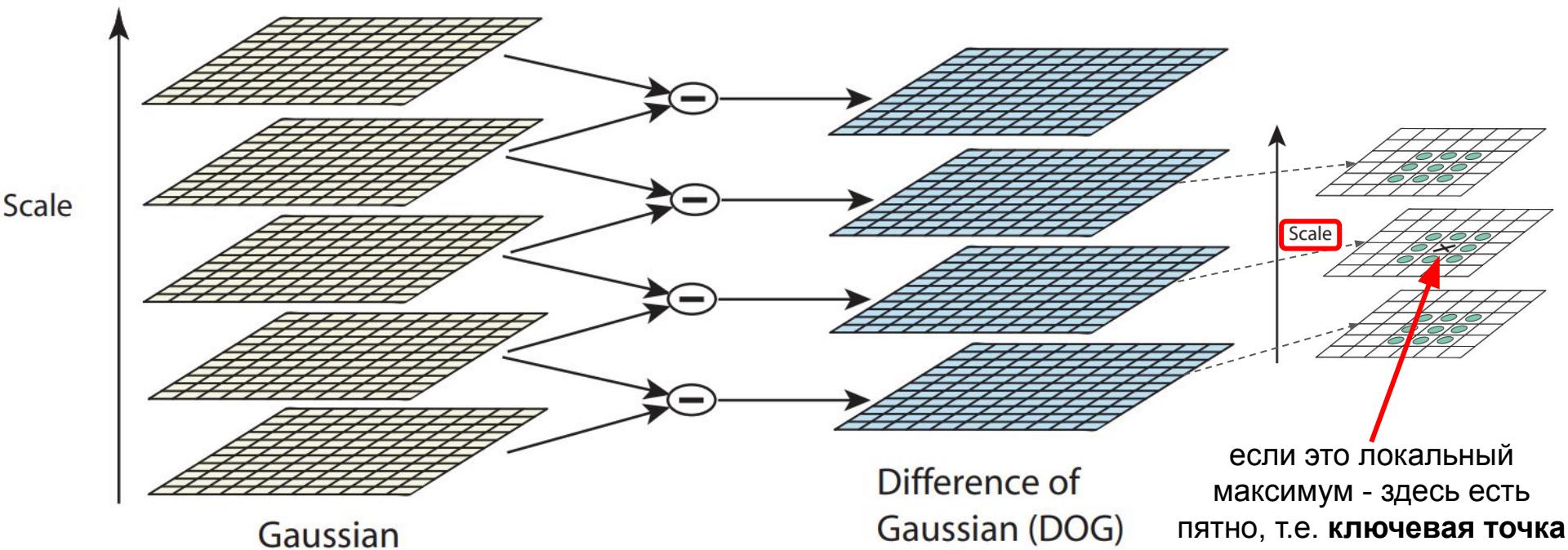
Вот такая иллюстрация есть в статье про SIFT Detector... Что она там делает?

Разложим картинку на частоты



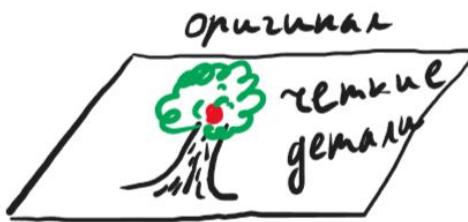
Вот такая иллюстрация есть в статье про SIFT Detector... Что она там делает?

Разложим картинку на частоты



Вот такая иллюстрация есть в статье про SIFT Detector... Что она там делает?

Разложим картинку на частоты



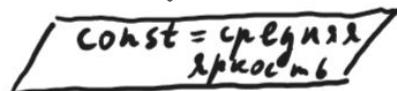
размытие (blur)



размытие (blur)

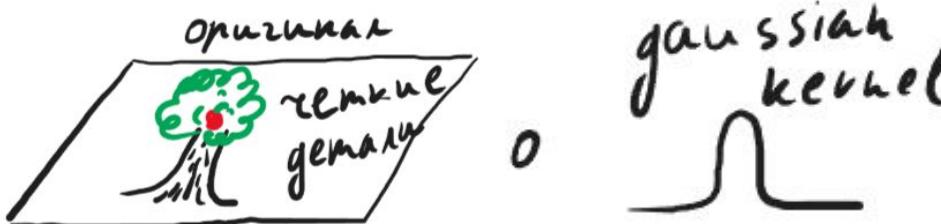


размытие (blur)



Gaussian Pyramid

Разложим картинку на частоты



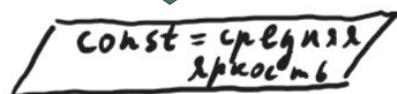
размытие (blur)



размытие (blur)

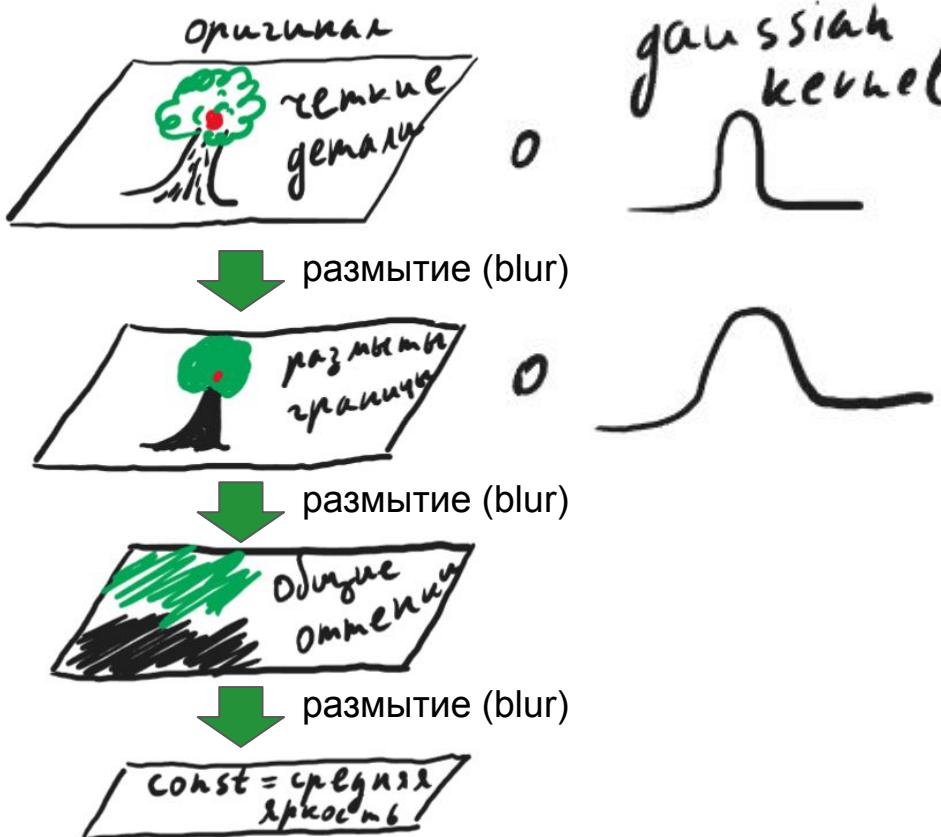


размытие (blur)



Gaussian Pyramid

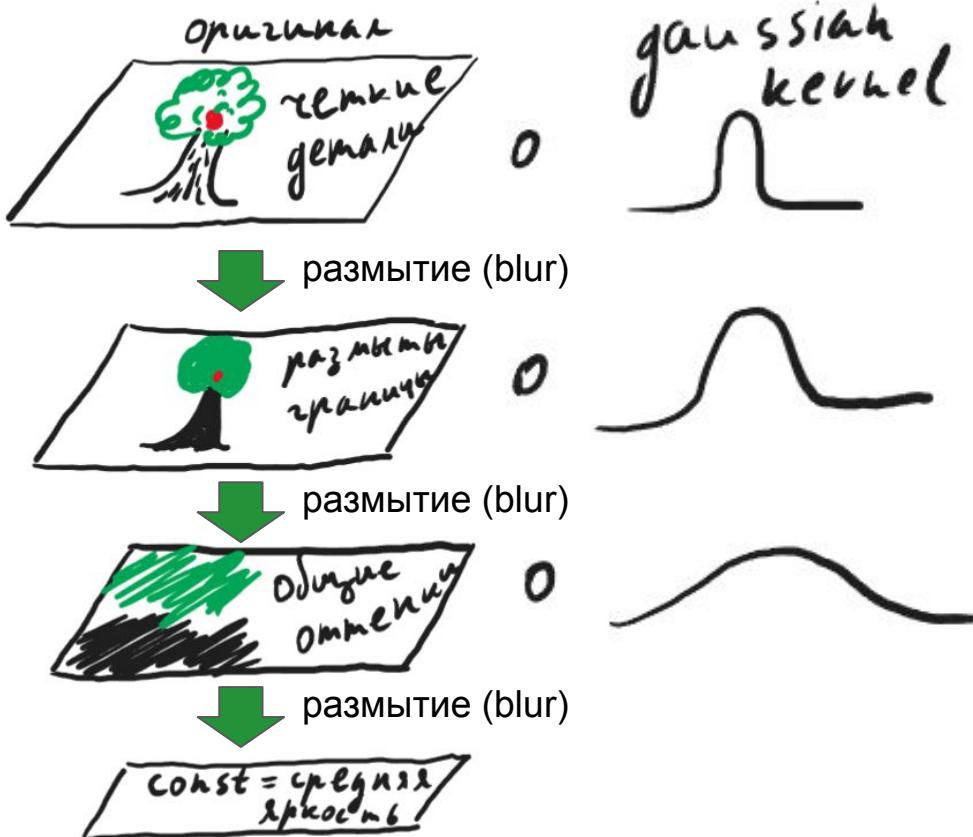
Разложим картинку на частоты



- фильтрует (убирает) **высокие частоты** (т.е. частоты выше некоторого порога)
например **шум**

Gaussian Pyramid

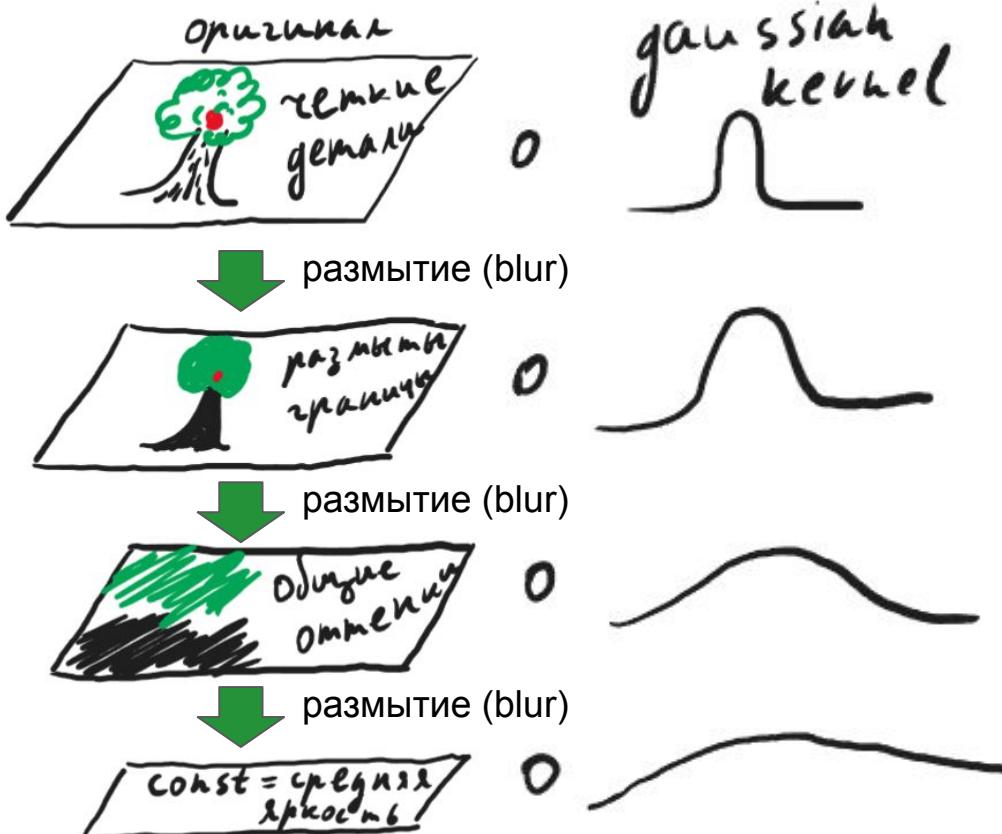
Разложим картинку на частоты



- фильтрует (убирает) **высокие частоты** (т.е. частоты выше некоторого порога)
например **шум**
- фильтрует (убирает) **средние частоты** (т.е. частоты выше некоторого порога)

Gaussian Pyramid

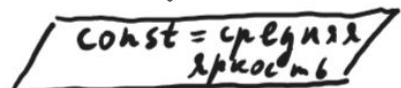
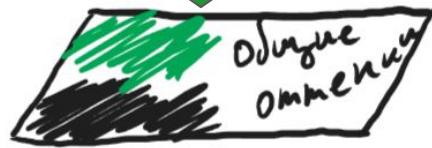
Разложим картинку на частоты



- фильтрует (убирает) **высокие частоты** (т.е. частоты выше некоторого порога)
например **шум**
- фильтрует (убирает) **средние частоты** (т.е. частоты выше некоторого порога)
- фильтрует (убирает) **почти все частоты** кроме уже совсем низких

Gaussian Pyramid

Разложим картинку на частоты



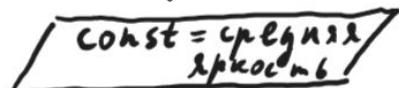
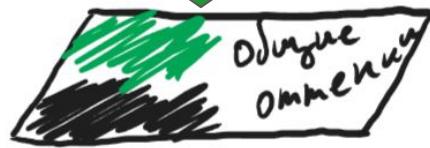
Gaussian Pyramid

убраны высокие частоты

убраны высокие частоты и средние частоты

Difference of Gaussian
(DoG)

Разложим картинку на частоты



Gaussian Pyramid

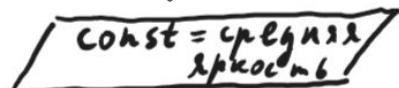
убраны высокие частоты

???

убраны высокие частоты и средние частоты

Difference of Gaussian
(DoG)

Разложим картинку на частоты



Gaussian Pyramid

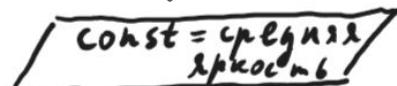
убраны высокие частоты

высоких частот не было ни сверху ни снизу

убраны высокие частоты и средние частоты

Difference of Gaussian
(DoG)

Разложим картинку на частоты



Gaussian Pyramid

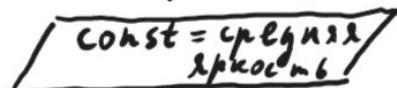
убраны высокие частоты

высоких частот не было ни сверху ни снизу
низкие частоты были и там и там - сократились

убраны высокие частоты и средние частоты

Difference of Gaussian
(DoG)

Разложим картинку на частоты



Gaussian Pyramid

убраны высокие частоты

высоких частот не было ни сверху ни снизу
низкие частоты были и там и там - сократились
остались только **средние** частоты

убраны высокие частоты и средние частоты

Difference of Gaussian
(DoG)

Разложим картинку на частоты

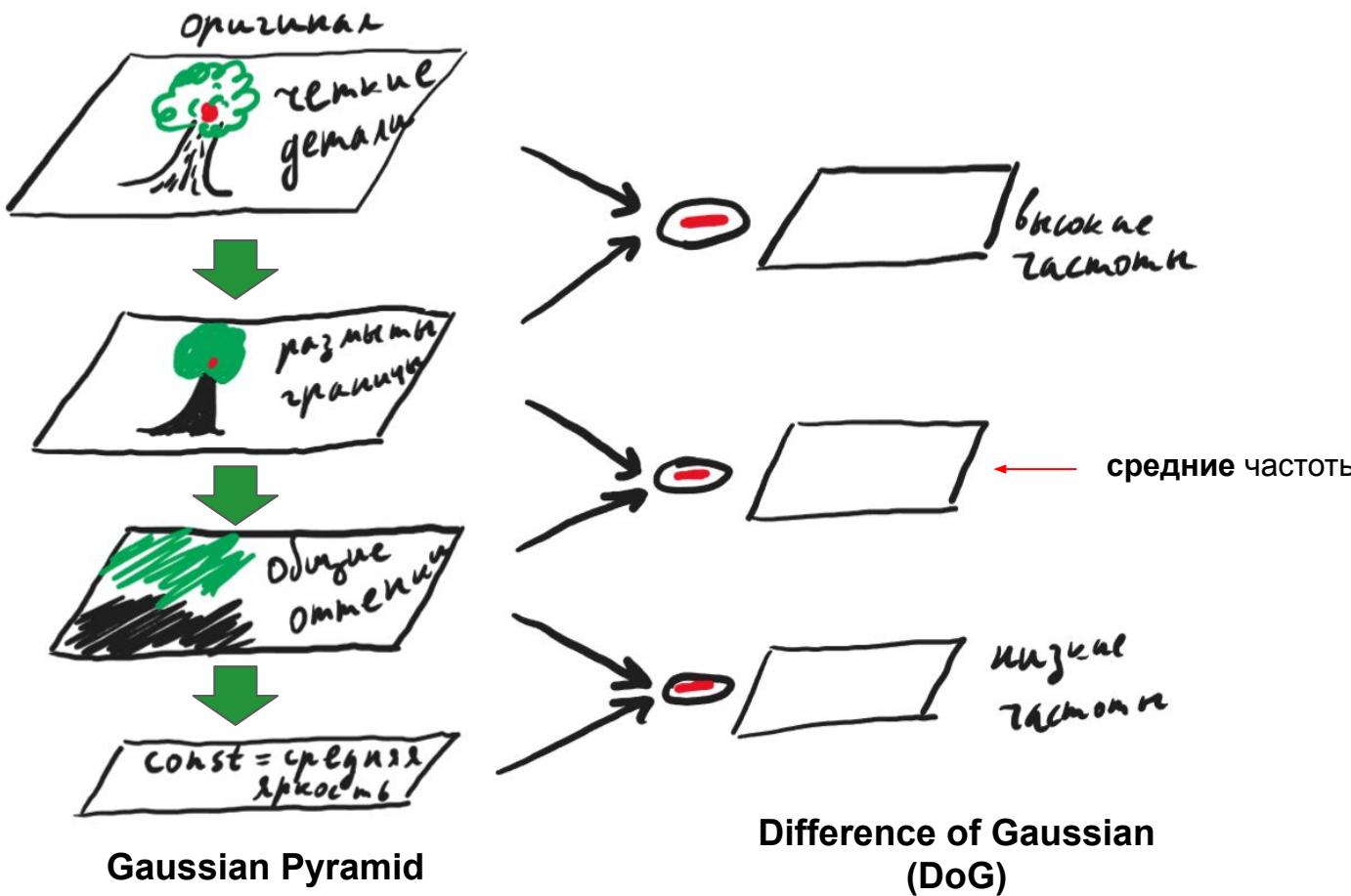




Fig. 8. The spline may be used to combine oddly shaped regions of very different images. The portion of Figure 8a within the region indicated by the mask in Figure 8c is inserted in the portion of Figure 8b which is outside this mask region (Figure 8d).



Fig. 8. The spline may be used to combine oddly shaped regions of very different images. The portion of Figure 8a within the region indicated by the mask in Figure 8c is inserted in the portion of Figure 8b which is outside this mask region (Figure 8d).

A Multiresolution Spline With Application to Image Mosaics, Burt et al., 1983

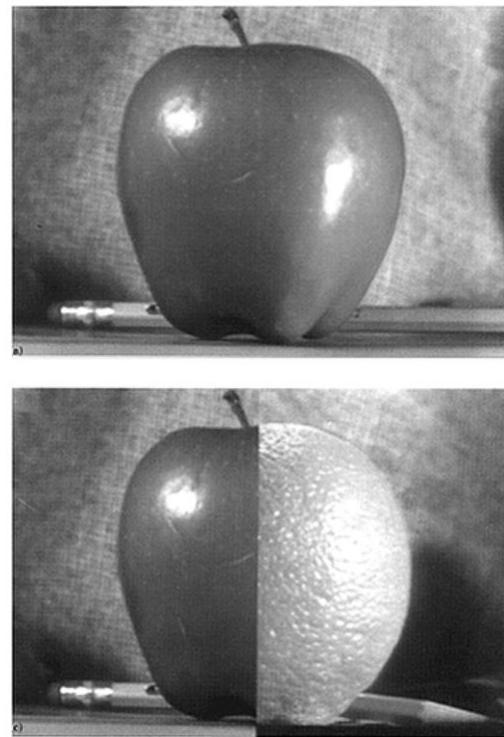


Fig. 7. The spline can be used to combine very different images. Here the left half of an apple is combined with the right half of an orange.



Fig. 8. The spline may be used to combine oddly shaped regions of very different images. The portion of Figure 8a within the region indicated by the mask in Figure 8c is inserted in the portion of Figure 8b which is outside this mask region (Figure 8d).

A Multiresolution Spline With Application to Image Mosaics, Burt et al., 1983

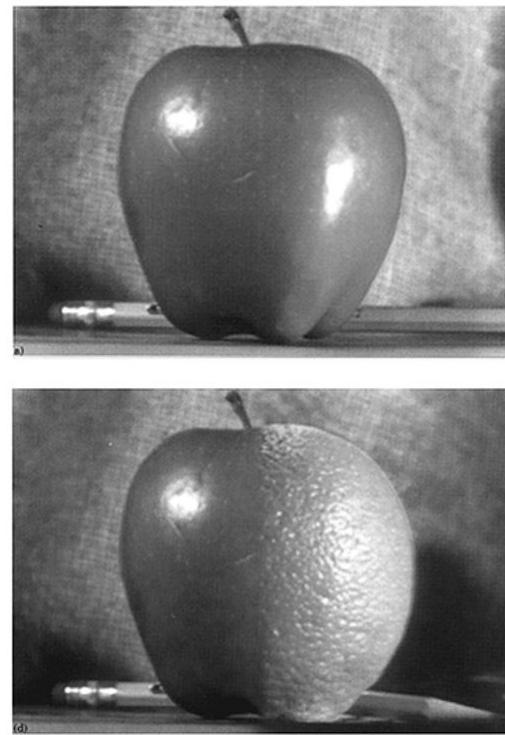
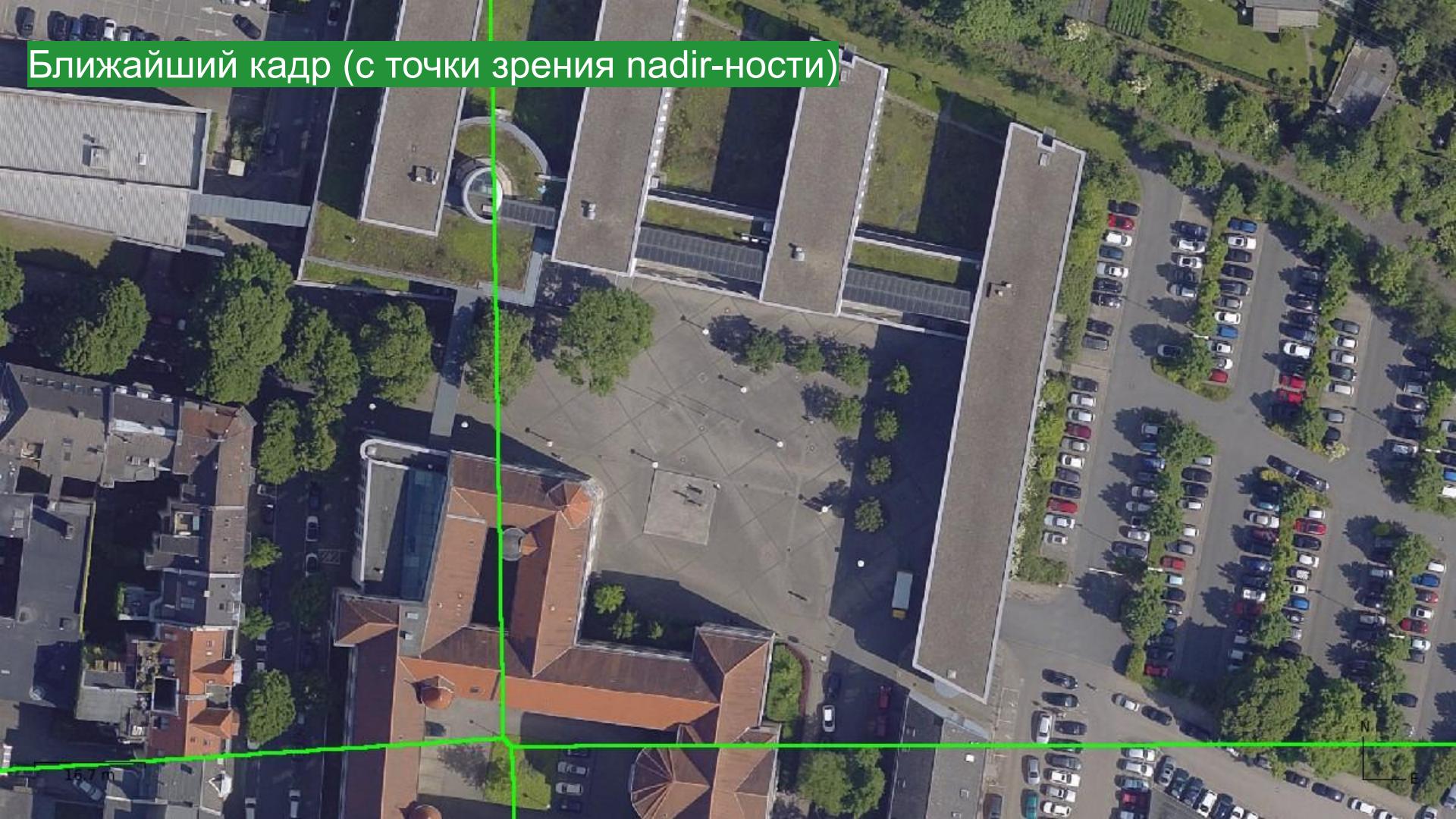
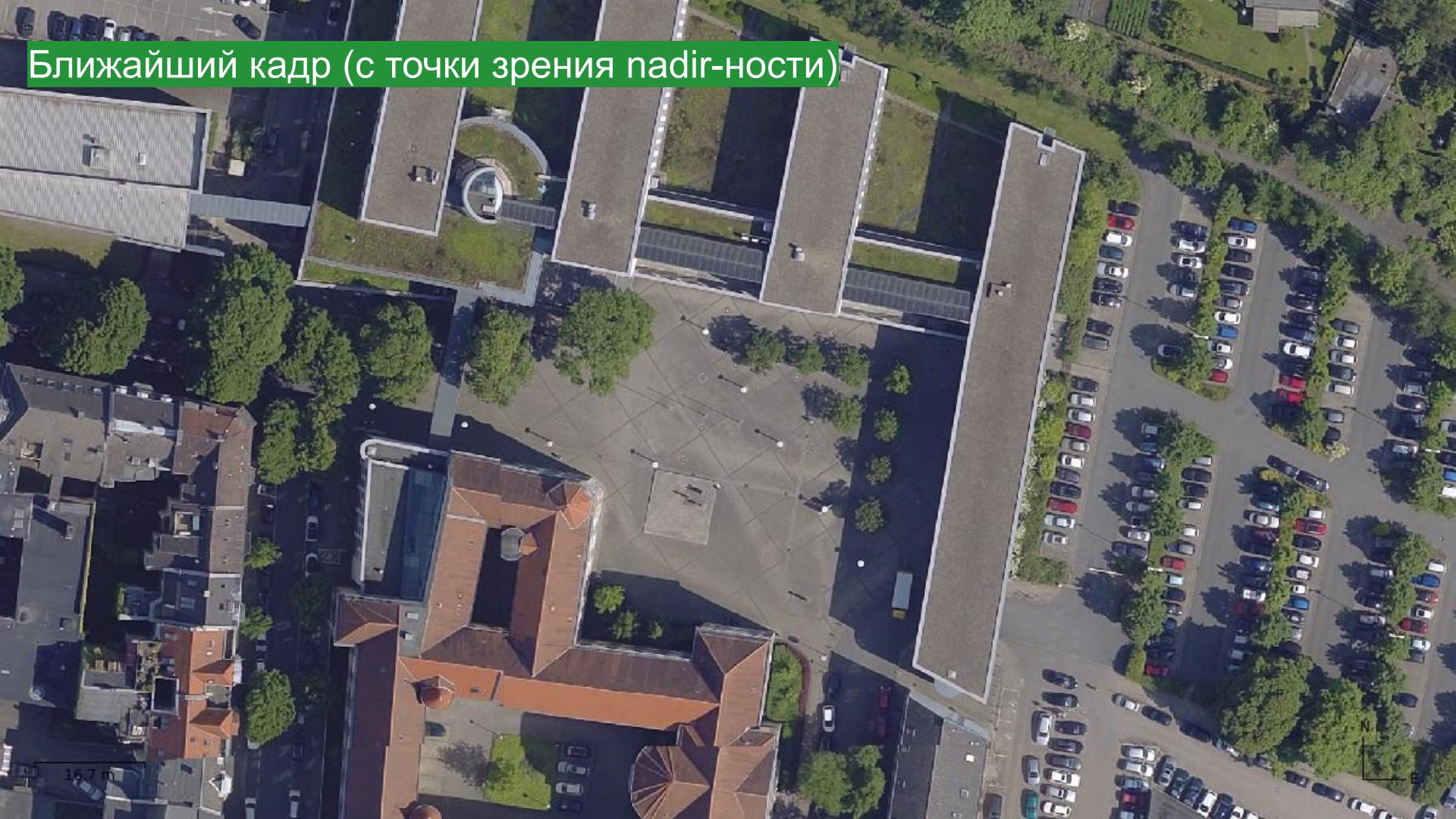


Fig. 7. The spline can be used to combine very different images. Here the left half of an apple is combined with the right half of an orange.

Ближайший кадр (с точки зрения nadir-ности)



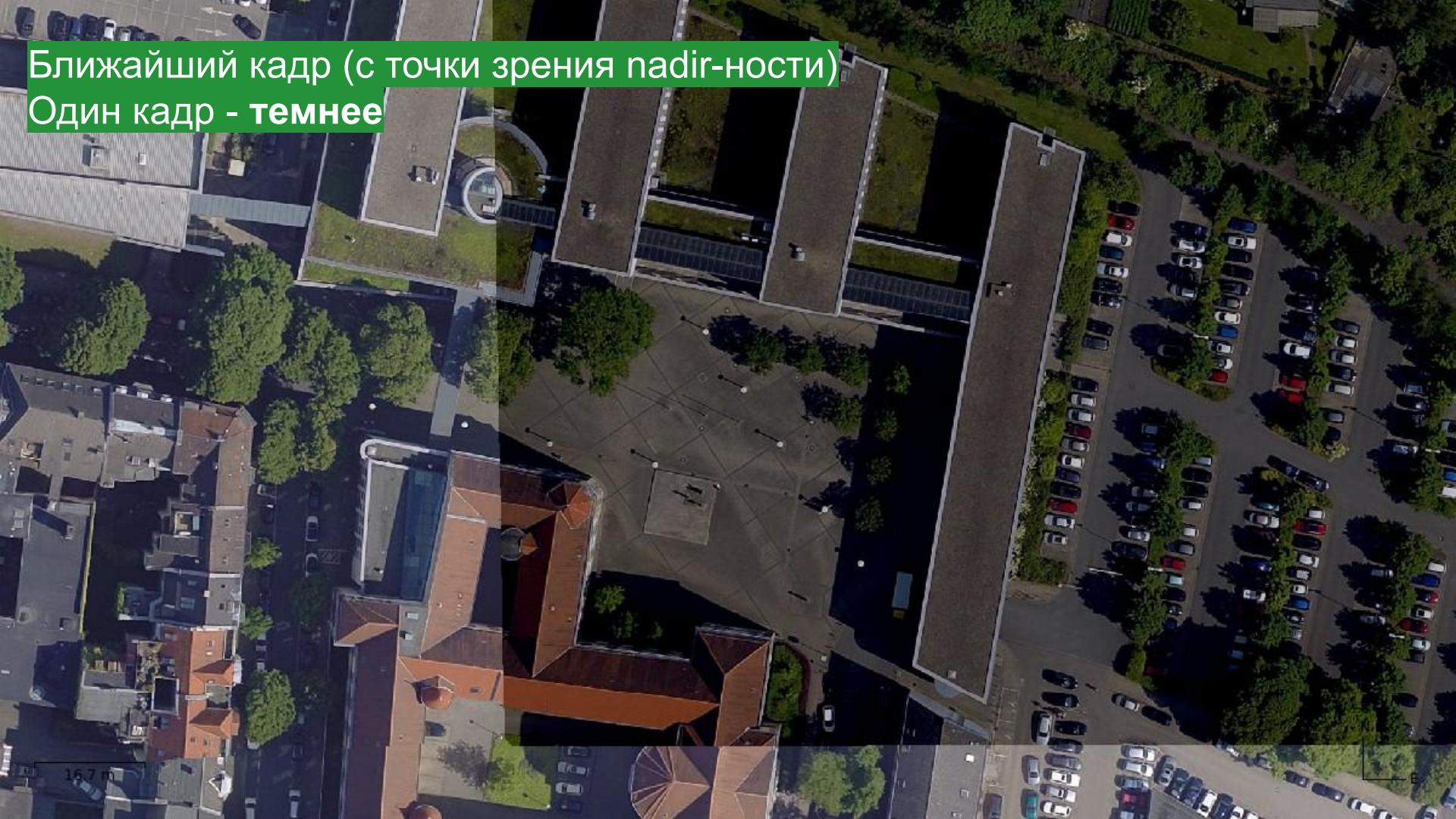
Ближайший кадр (с точки зрения nadir-ности)



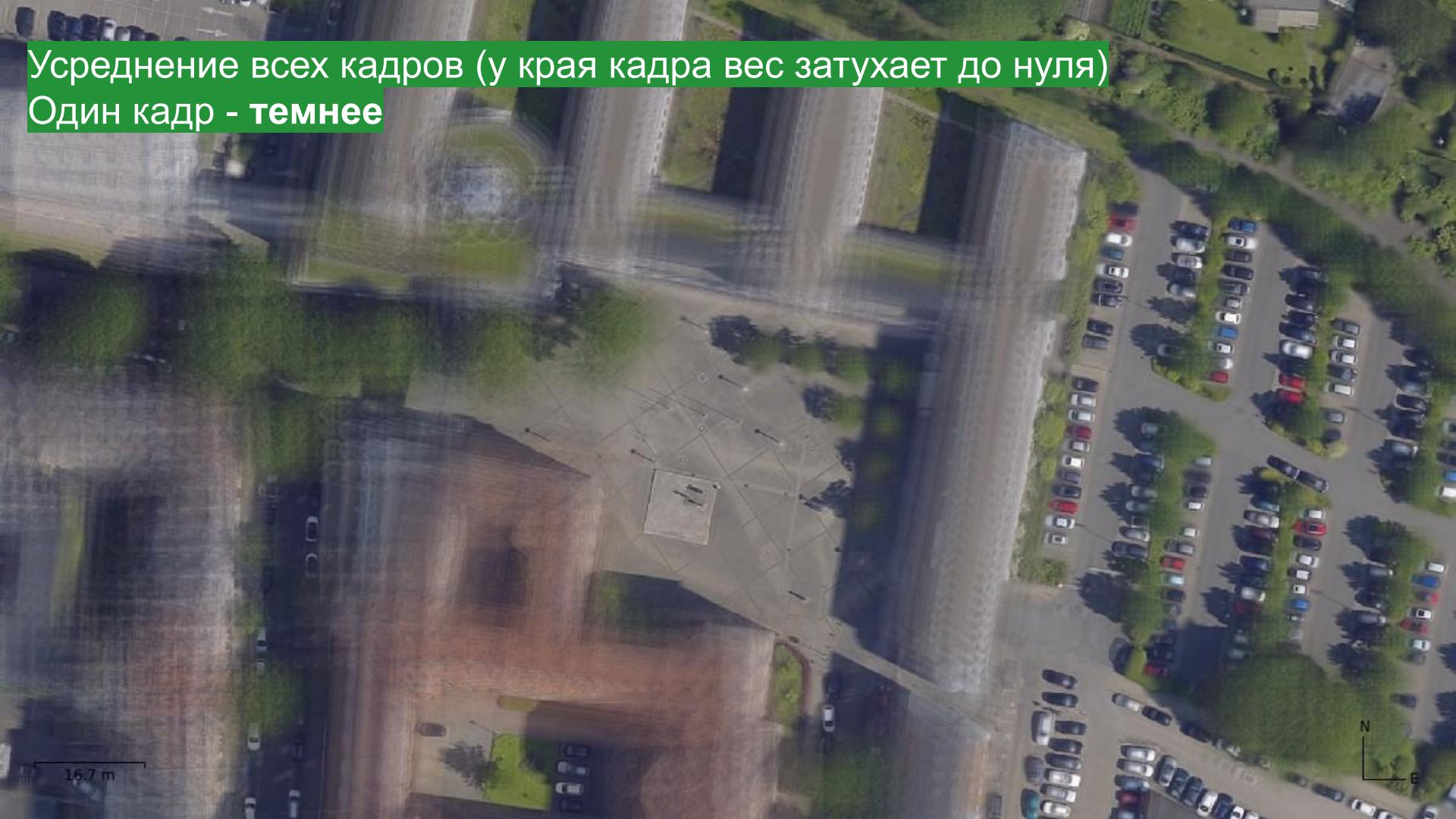
16.7 m

Ближайший кадр (с точки зрения nadir-ности)

Один кадр - темнее

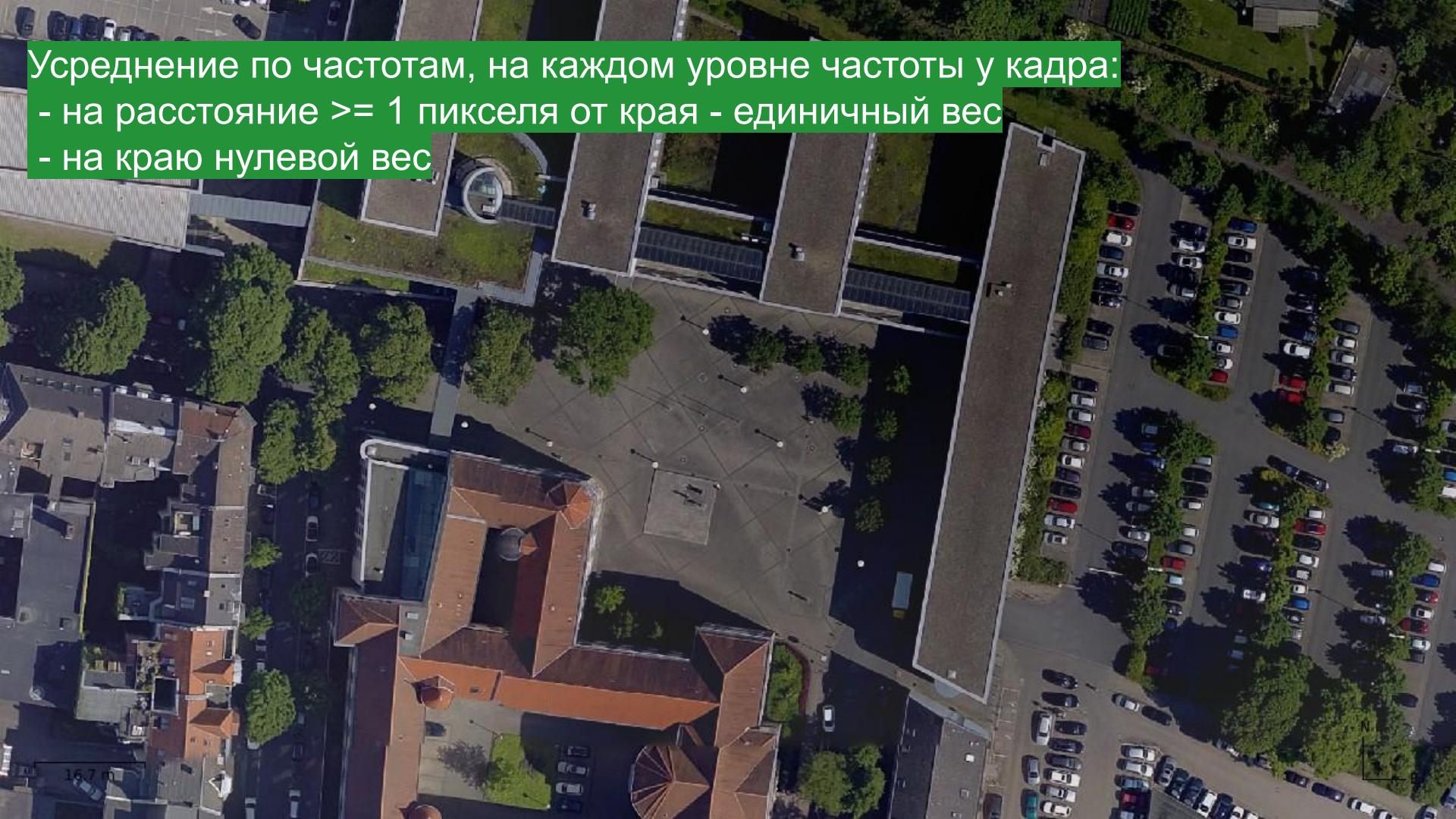


Усреднение всех кадров (у края кадра вес затухает до нуля)
Один кадр - темнее

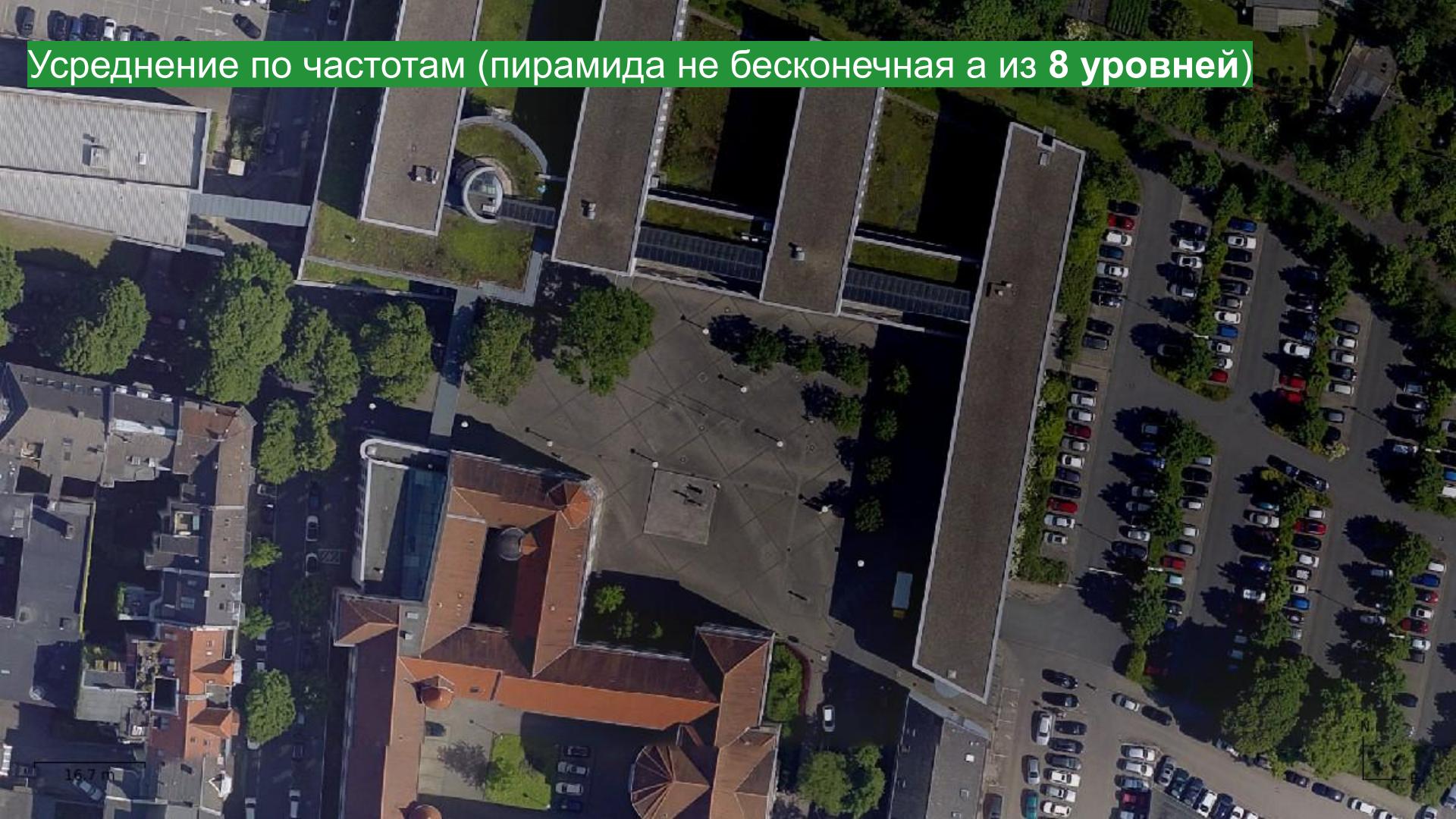


Усреднение по частотам, на каждом уровне частоты у кадра:

- на расстояние ≥ 1 пикселя от края - единичный вес
- на краю нулевой вес

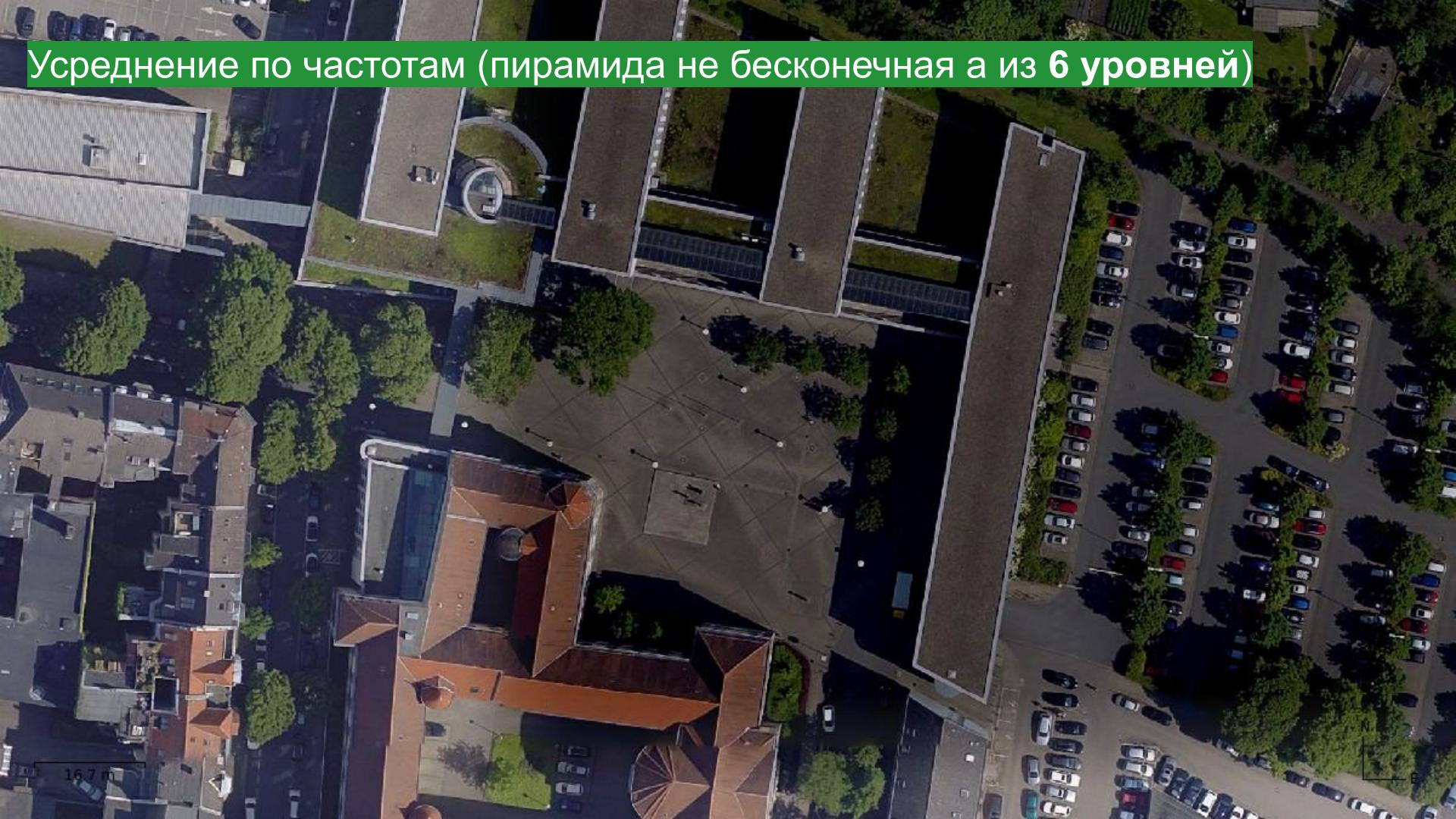


Усреднение по частотам (пирамида не бесконечная а из 8 уровней)



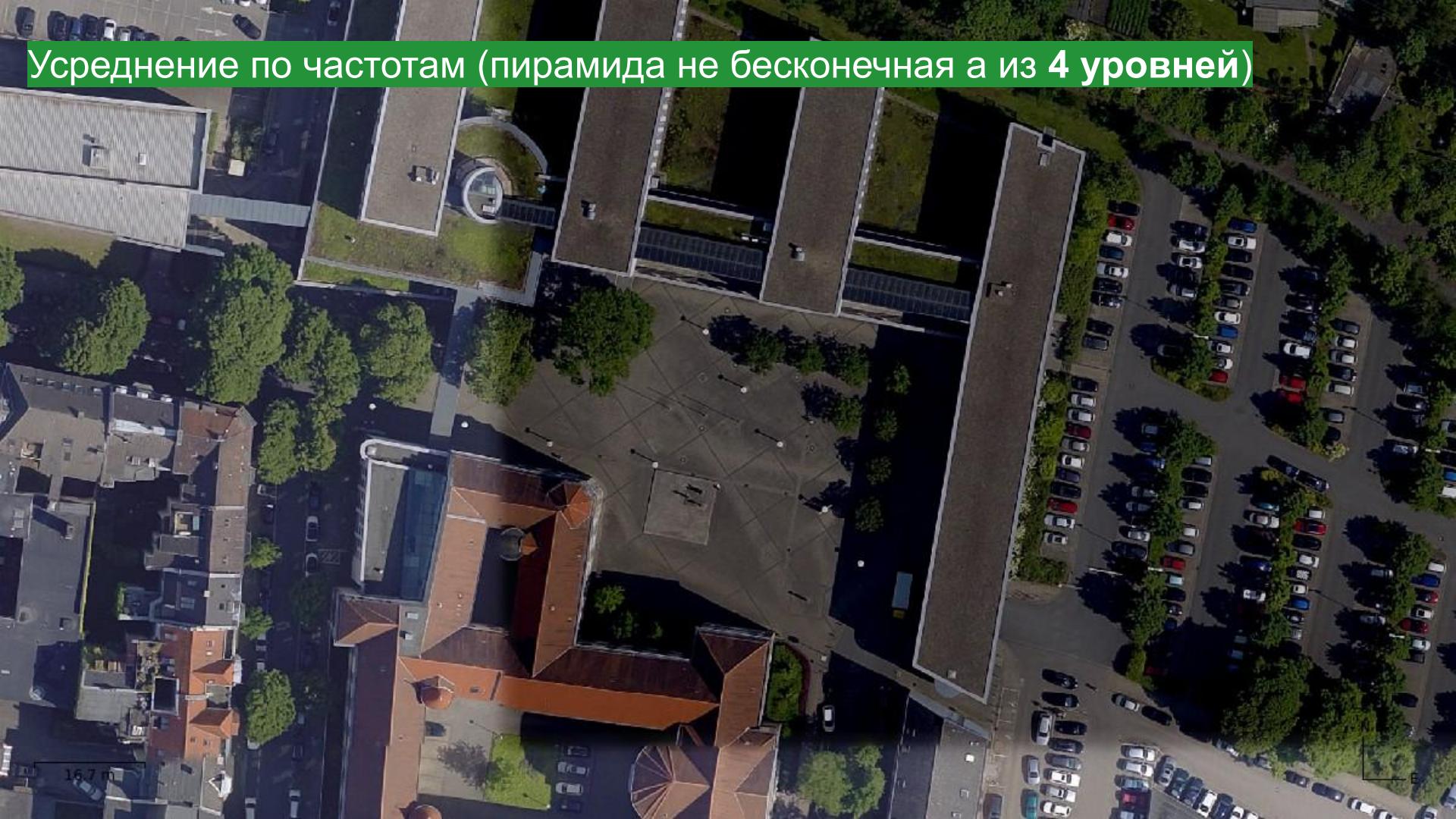
16.7 m

Усреднение по частотам (пирамида не бесконечная а из 6 уровней)



16.7 m

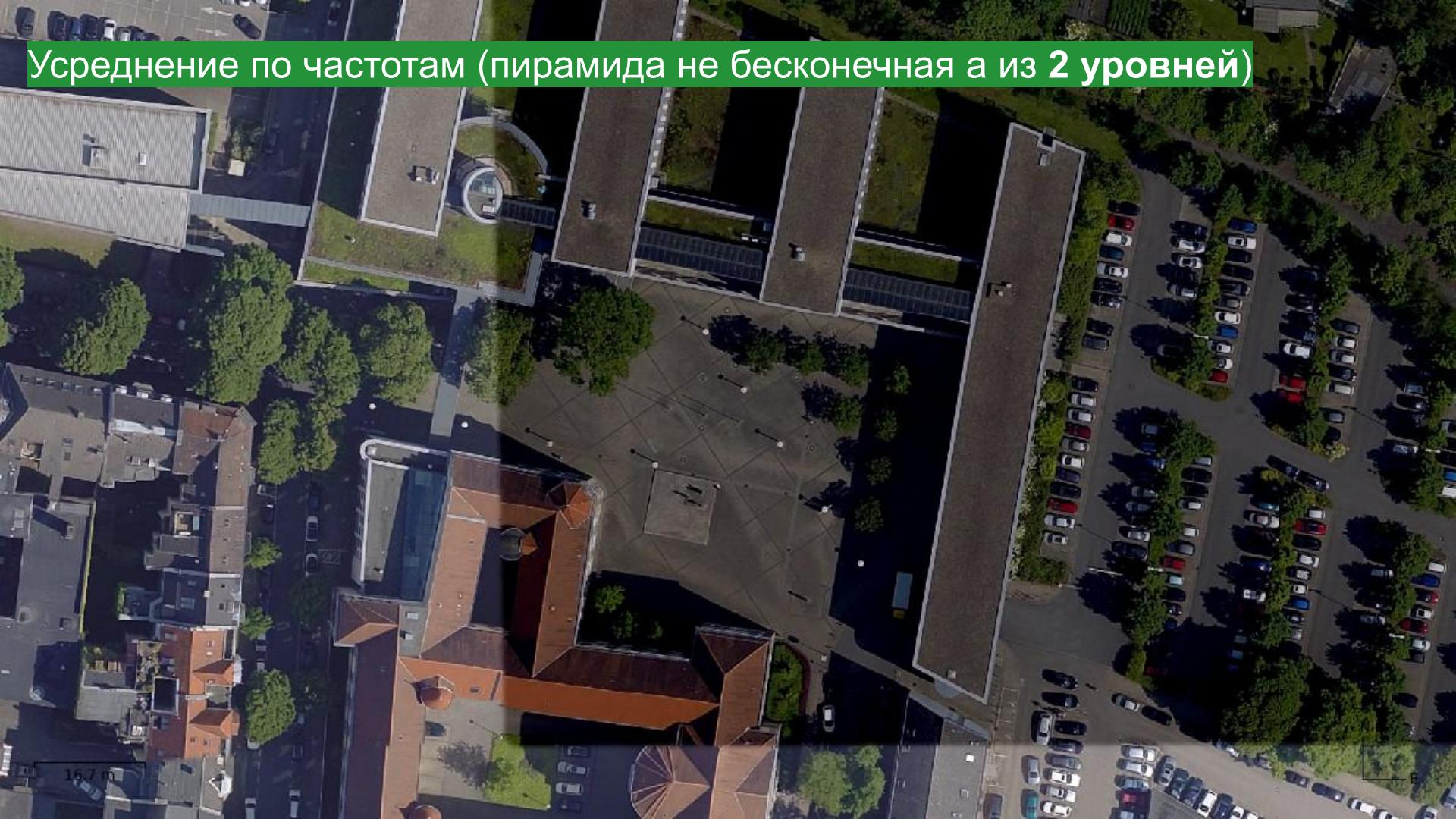
Усреднение по частотам (пирамида не бесконечная а из 4 уровней)



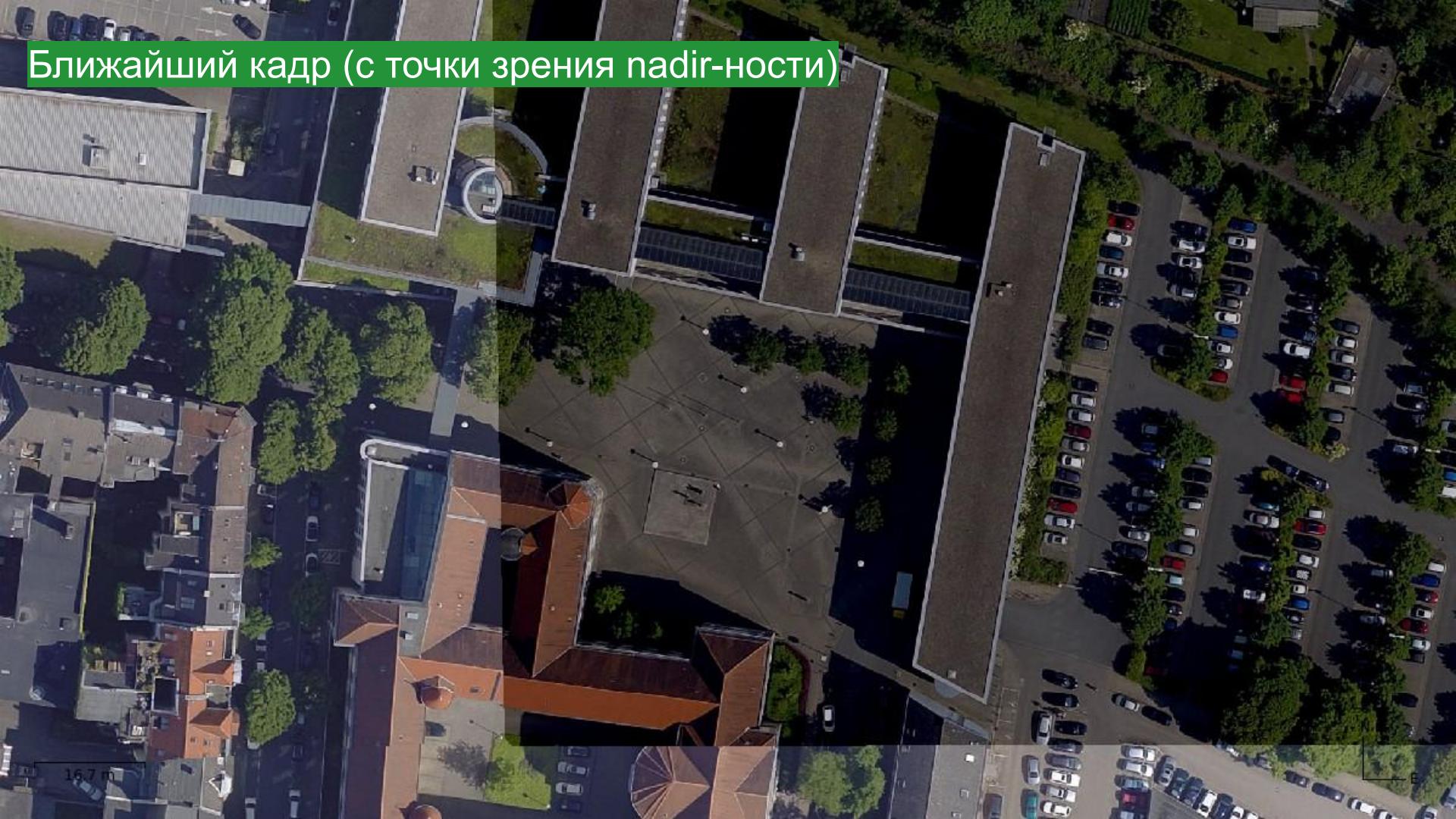
16.7 m

E

Усреднение по частотам (пирамида не бесконечная а из 2 уровней)



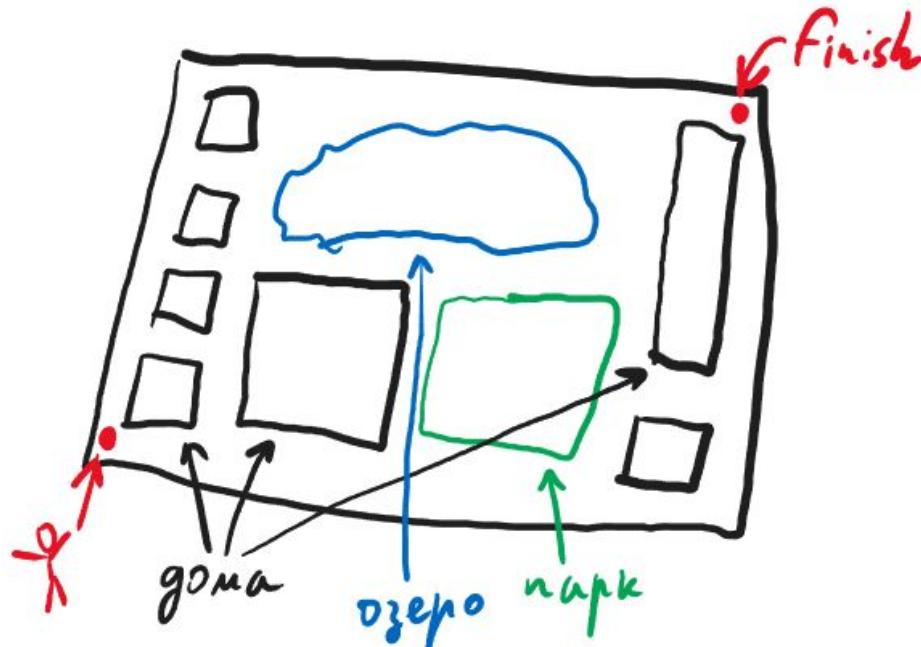
Ближайший кадр (с точки зрения nadir-ности)



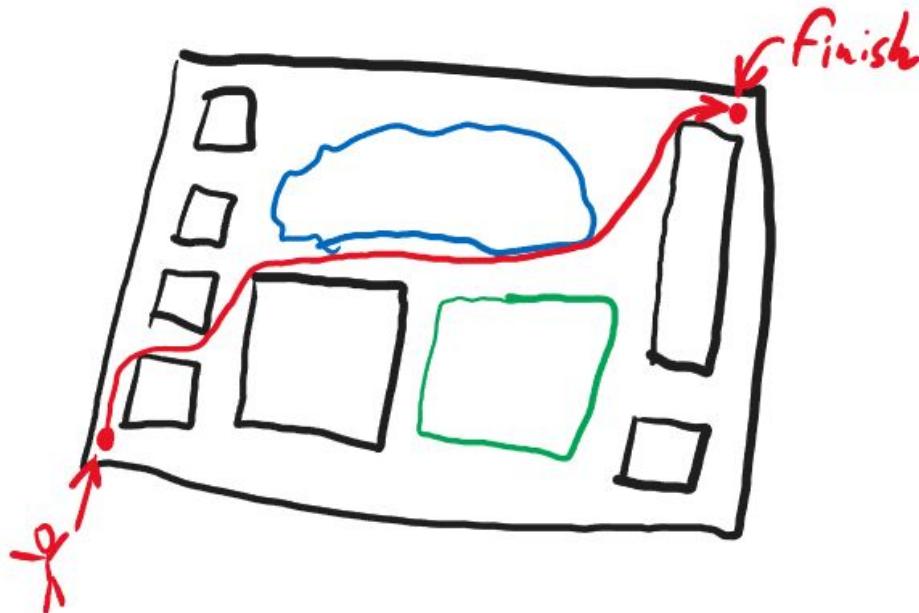
16.7 m

E

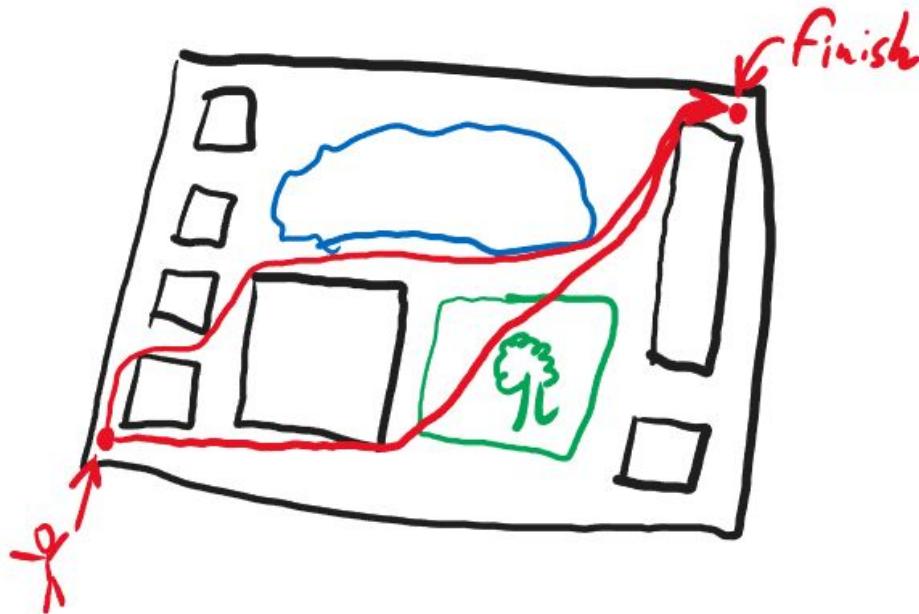
Отвлечемся: найдем кратчайший путь в городе



Отвлечемся: найдем кратчайший путь в городе



Отвлечемся: найдем кратчайший путь в городе



Отвлечемся: найдем кратчайший путь в городе



Если построим граф, то задача решаема.
Например **алгоритмом Дейкстры**.

Отвлечемся: найдем кратчайший путь в городе



Отвлечемся: найдем кратчайший путь в городе



Отвлечемся: найдем кратчайший путь в городе



Отвлечемся: найдем кратчайший путь в городе



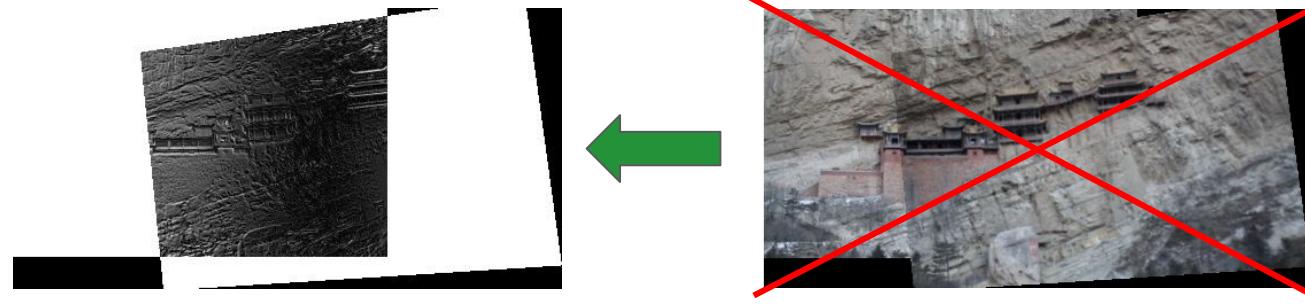
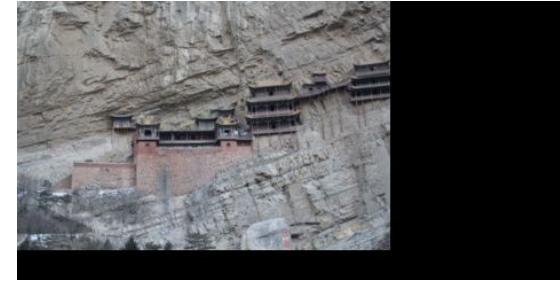
Прокладка умного шва в панораме



Прокладка умного шва в панораме



Прокладка умного шва в панораме

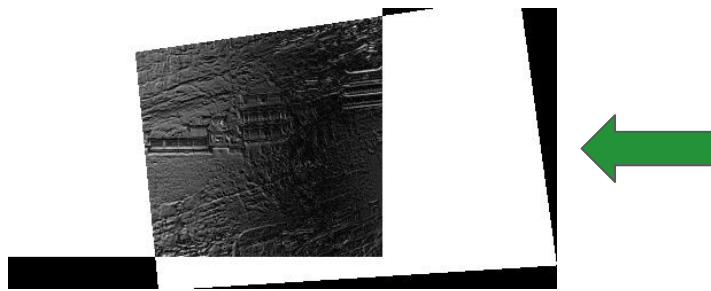


Построим картинку-“панораму” где в каждом пикселе
написано **насколько отличаются** картинки:

Прокладка умного шва в панораме

Где тогда нам бы хотелось провести шов?

Построим картинку-“панораму” где в каждом пикселе написано **насколько отличаются картинки**:



Прокладка умного шва в панораме

Где тогда нам бы хотелось провести шов?

Построим картинку-“панораму” где в каждом пикселе написано **насколько отличаются картинки**:



Прокладка умного шва в панораме



Где тогда нам бы хотелось провести шов?

Построим картинку-“панораму” где в каждом пикселе
написано **насколько отличаются картинки**:



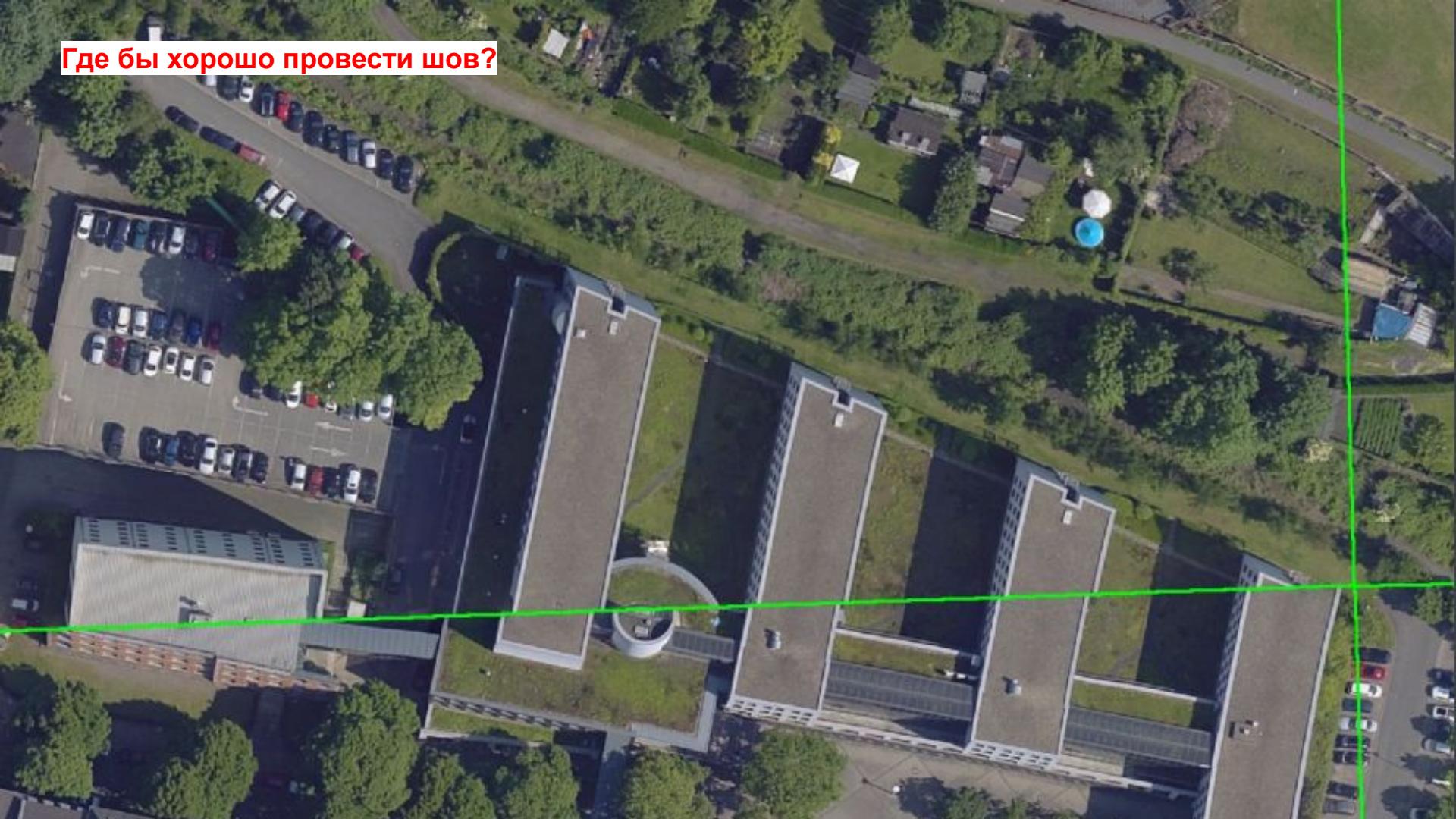
Прокладка умного шва в панораме

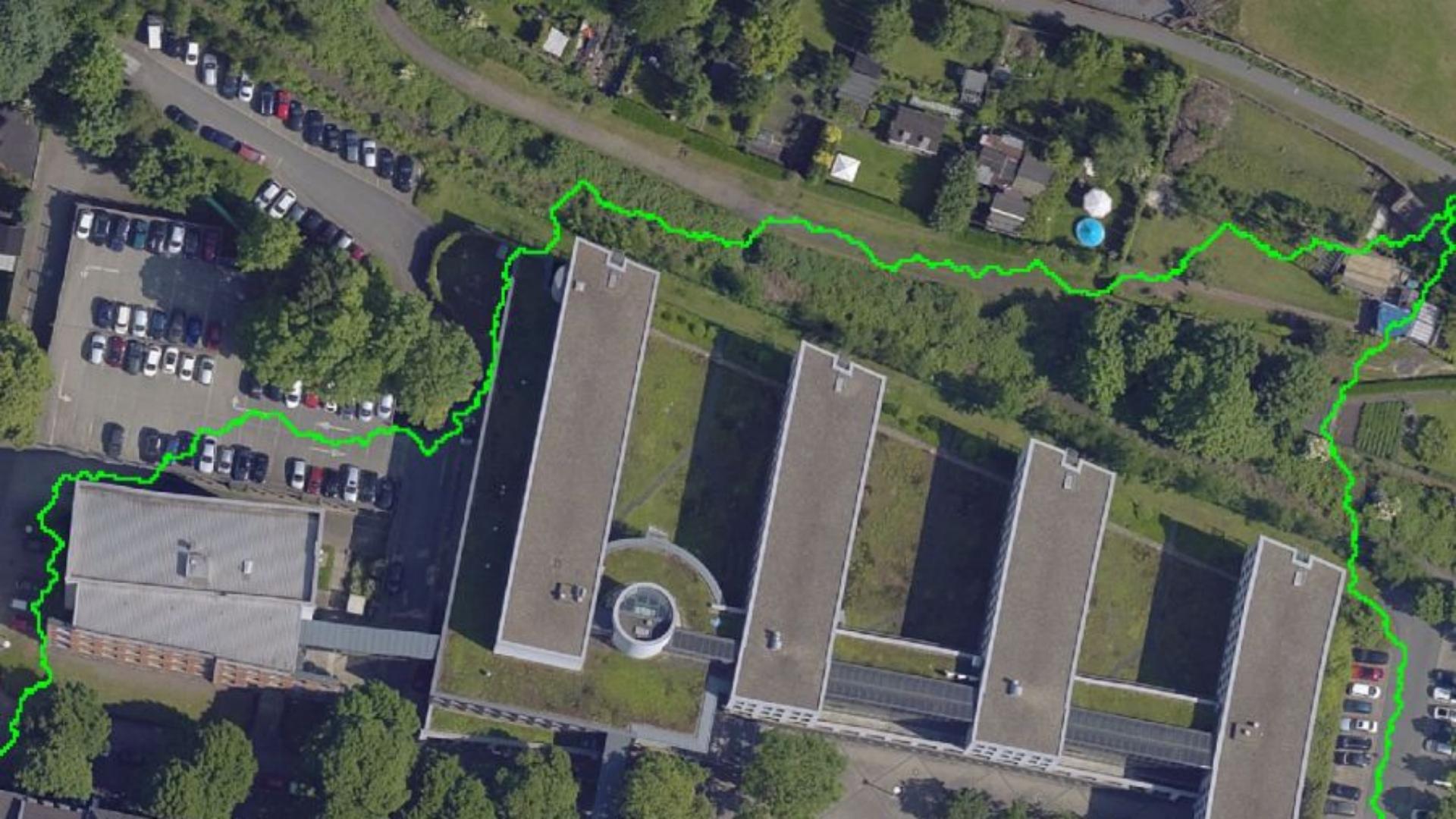
Где тогда нам бы хотелось провести шов?

Построим картинку-“панораму” где в каждом пикселе написано **насколько отличаются картинки**:



Где бы хорошо провести шов?





Если хочется покодить

- 1) [Здесь](#) есть какие-то материалы с заданиями на [github](#):
 - урок 3: OpenCV + простой вариант зеленого экрана для веб-камеры
 - уроки 8-9: преобразование Хафа для поиска прямых
 - урок 15: RANSAC для поиска прямой по точкам
 - урок 16: сшивка картинок на базе SIFT + K-Ratio test + LR check + RANSAC
 - уроки 18-20: прокладывание умного шва в панораме
 - уроки 21-22: ретуширование (залатывание дыр)

Если хочется покодить

- 1) [Здесь](#) есть какие-то материалы с заданиями на [github](#):
 - урок 3: OpenCV + простой вариант зеленого экрана для веб-камеры
 - уроки 8-9: преобразование Хафа для поиска прямых
 - урок 15: RANSAC для поиска прямой по точкам
 - урок 16: сшивка картинок на базе SIFT + K-Ratio test + LR check + RANSAC
 - уроки 18-20: прокладывание умного шва в панораме
 - уроки 21-22: ретуширование (залатывание дыр)
- 2) [Курс для студентов](#) по фотограмметрии с заданиями на [github](#).

Вопросы?



Полярный Николай

polarnick239@gmail.com