

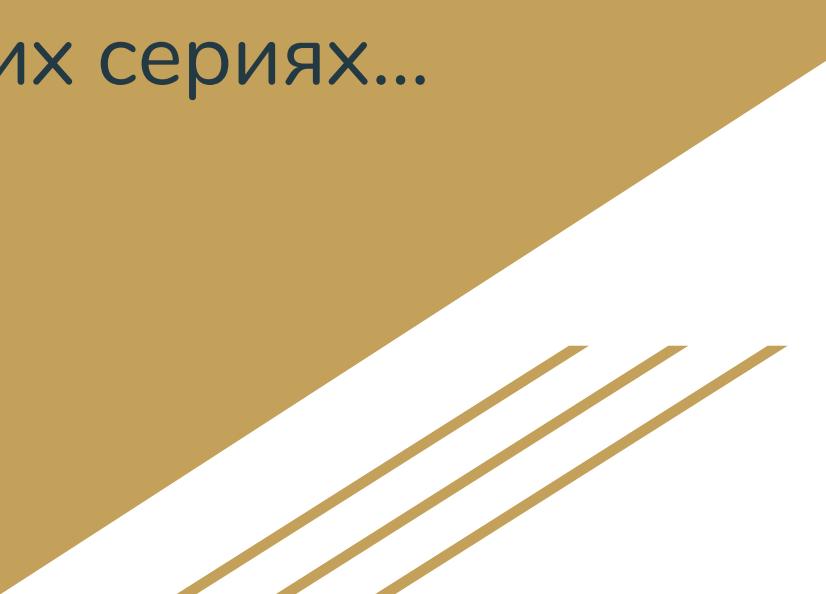
Компьютерное зрение

Лекция 3. Признаки изображения

Детектор Харриса, Panorama Stitching, RANSAC. SIFT. HoG

04.06.2020

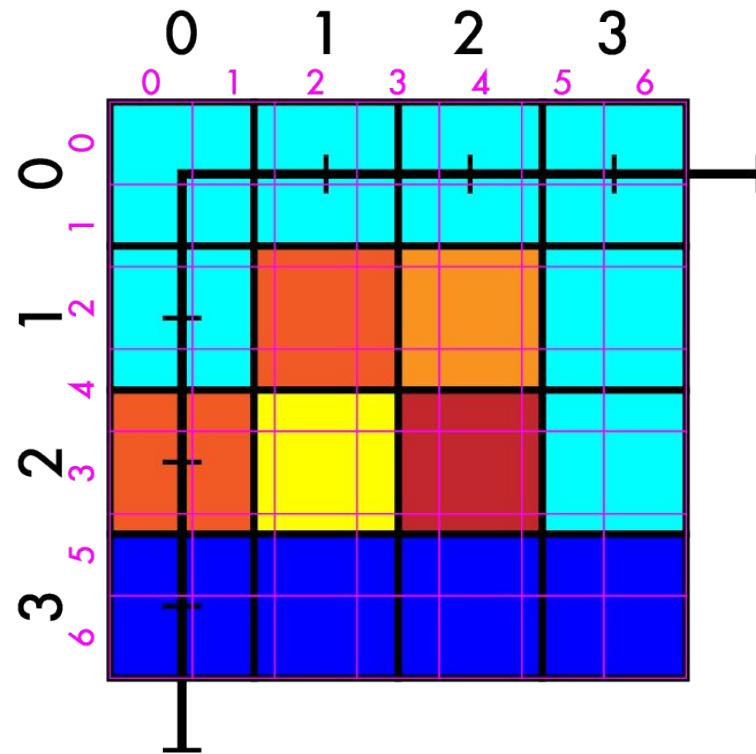
Руслан Алиев



В предыдущих сериях...

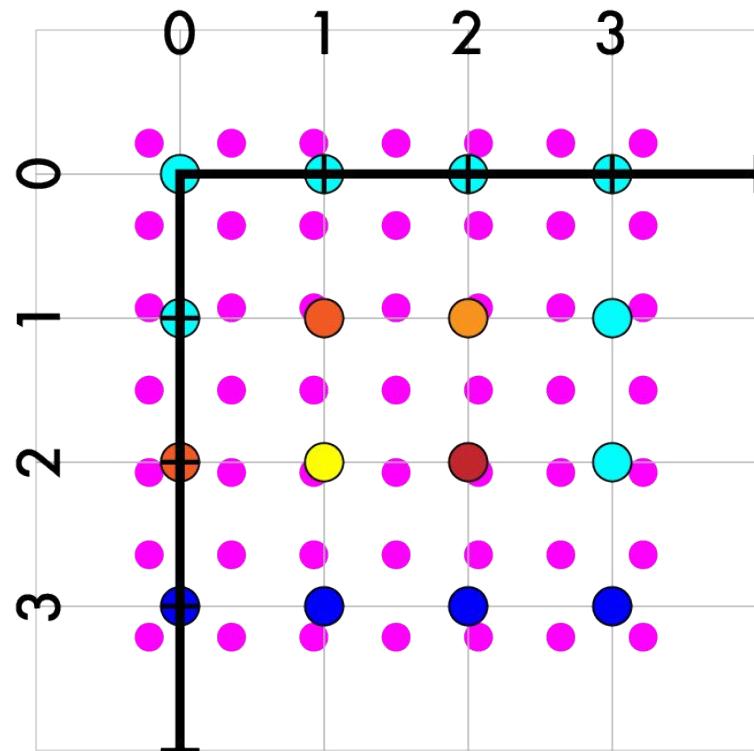
Ресайзинг $4 \times 4 \rightarrow 7 \times 7$

- Создаем новое изображение
- Матчим координаты



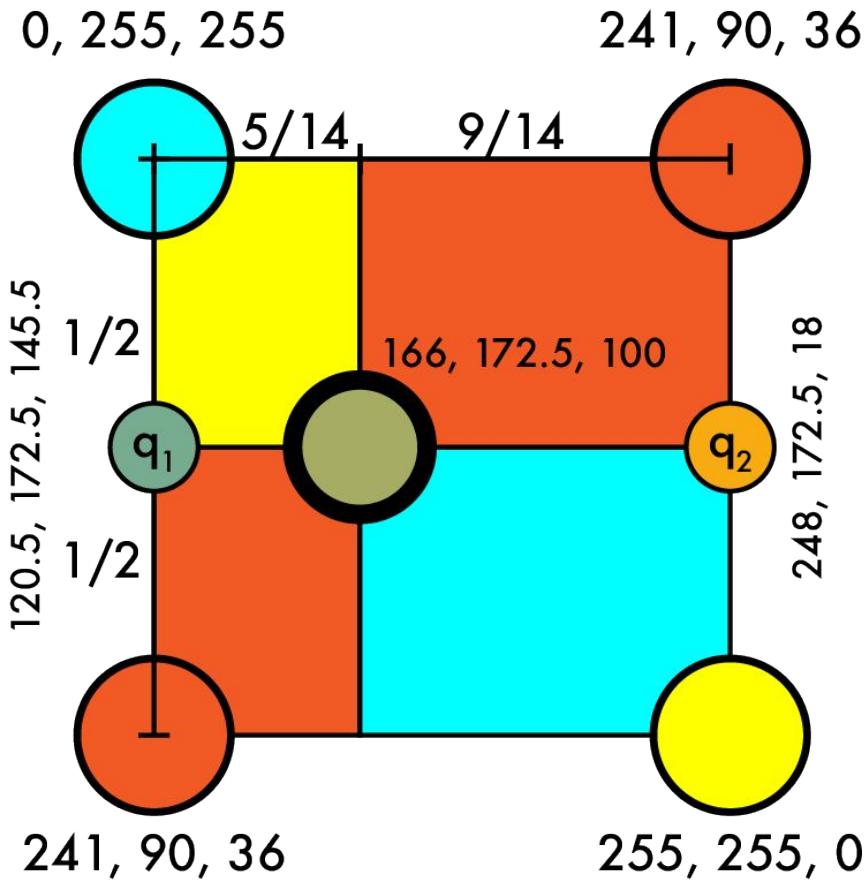
Ресайзинг $4 \times 4 \rightarrow 7 \times 7$

- Создаем новое изображение
- Матчим координаты
 - $4/7 X - 3/14 = Y$
- Итерируемся по всем новым точкам
 - Отображаем в старые координаты



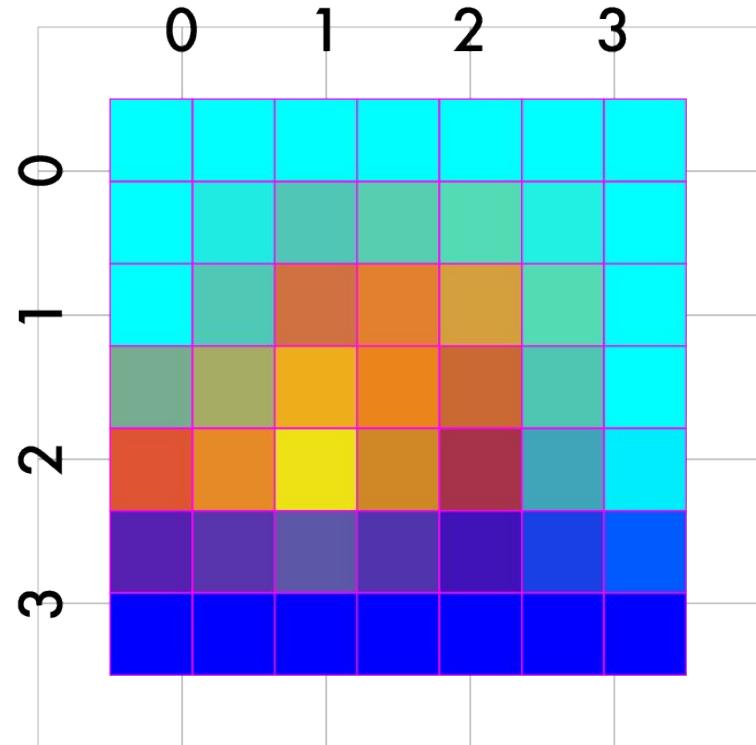
Ресайзинг 4x4 -> 7x7

- Создаем новое изображение
- Матчим координаты
 - $4/7 X - 3/14 = Y$
- Итерируемся по всем новым точкам
 - Отображаем в старые координаты
 - $(1, 3) \rightarrow (5/14, 7/14)$
 - Интерполируем старые значения
 - $q = (166, 172.5, 100)$

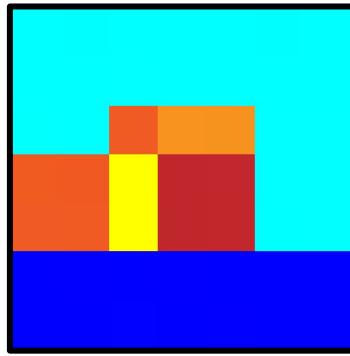
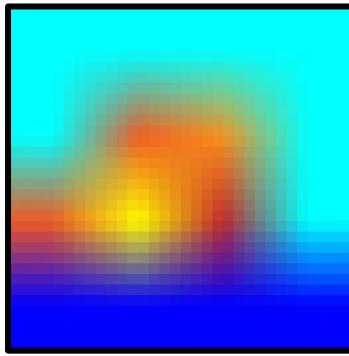
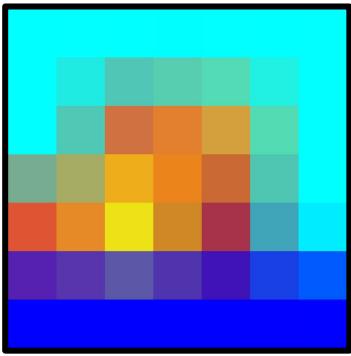
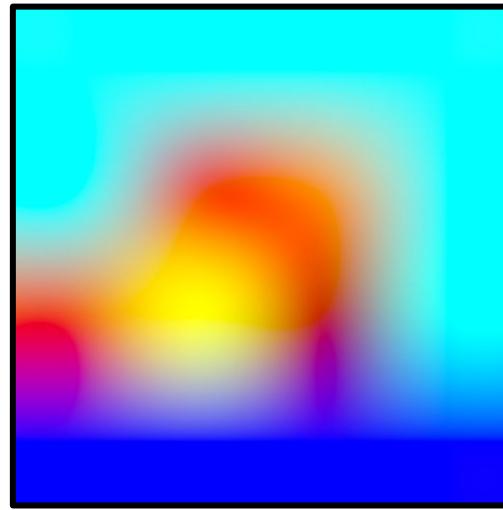
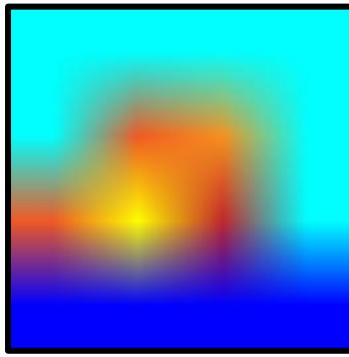
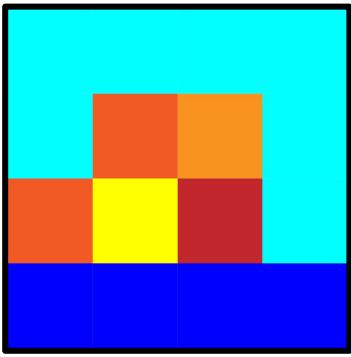


Ресайзинг $4 \times 4 \rightarrow 7 \times 7$

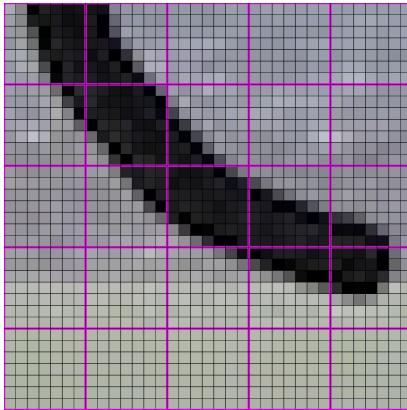
- Создаем новое изображение
- Матчим координаты
 - $4/7 X - 3/14 = Y$
- Итерируемся по всем новым точкам
 - Отображаем в старые координаты
 - $(1, 3) \rightarrow (5/14, 7/14)$
 - Интерполируем старые значения
 - И так далее...



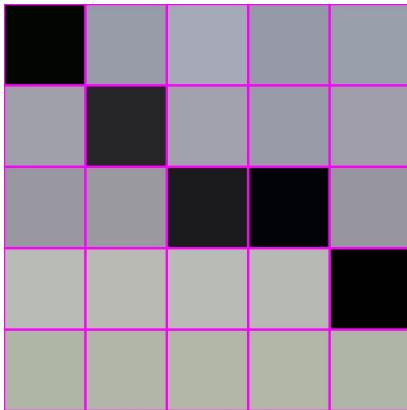
Различные методы



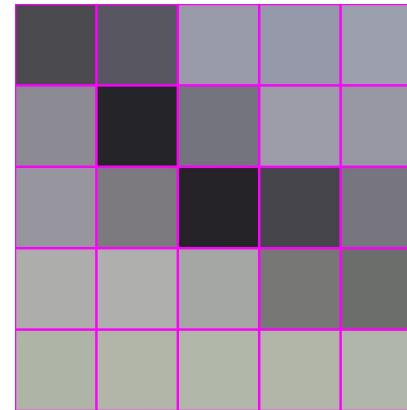
Для уменьшения интерполяция работает плохо



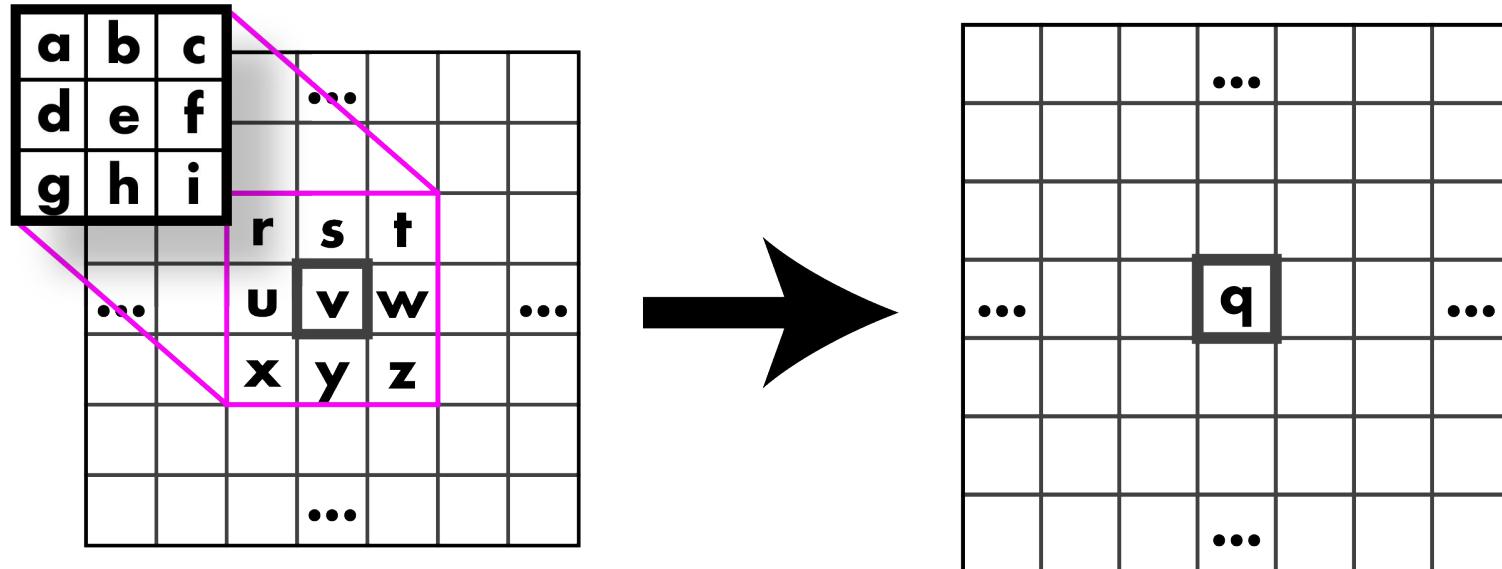
“interpolation”



averaging

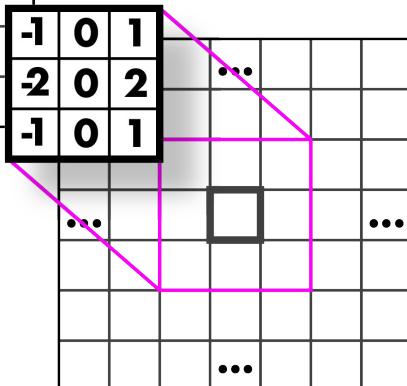
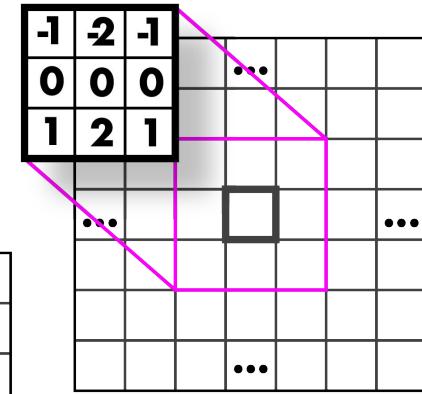
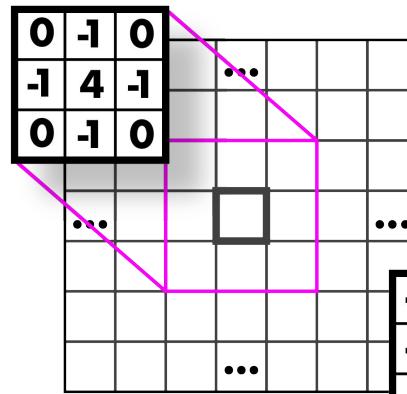
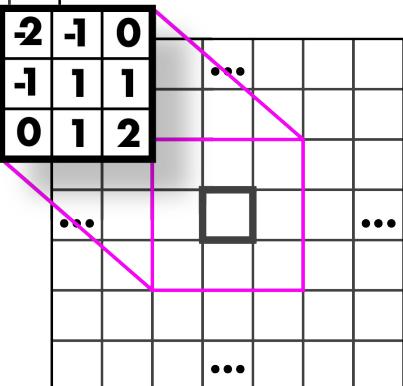
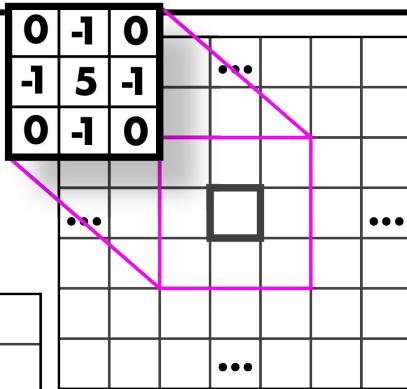
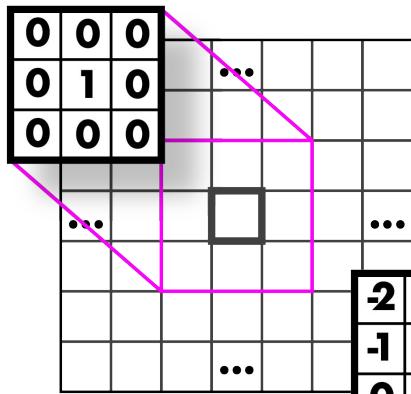


Конволюция: взвешенная сумма соседей пикселя



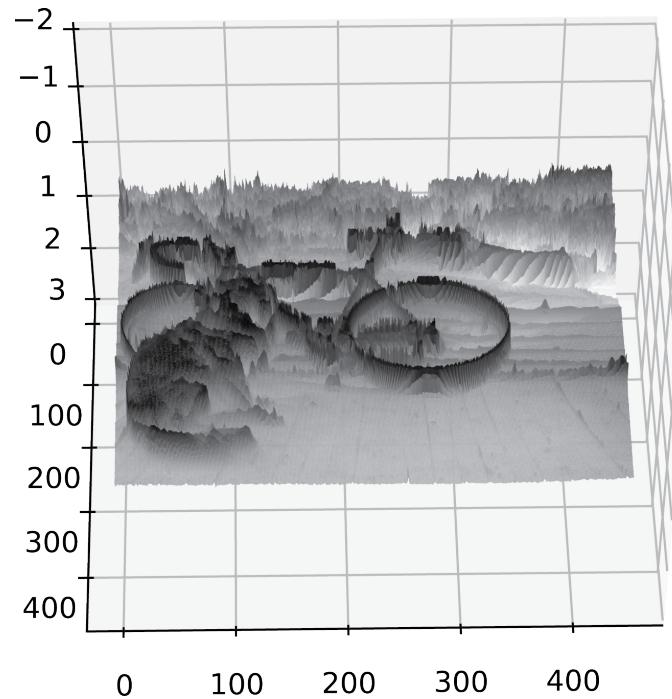
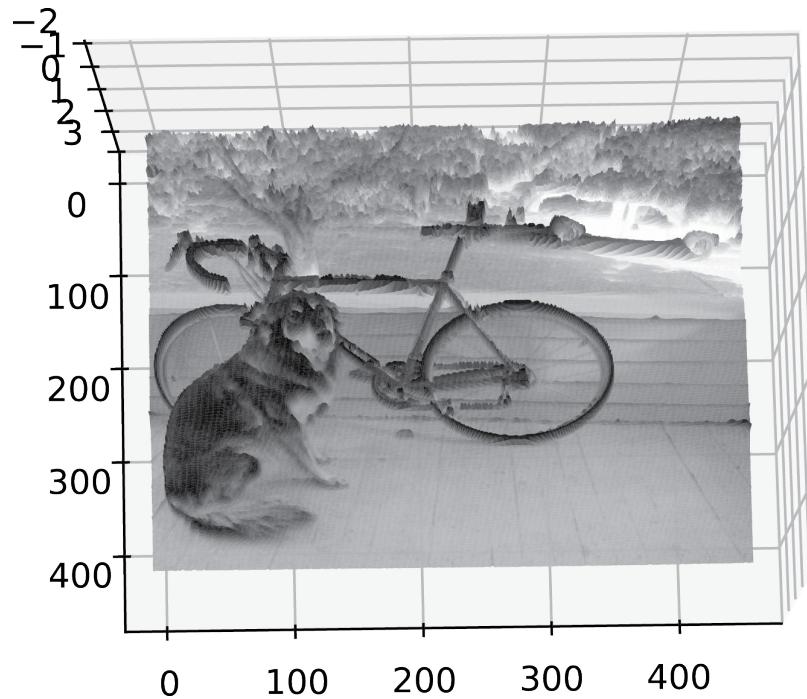
$$q = a \times r + b \times s + c \times t + d \times u + e \times v + f \times w + g \times x + h \times y + i \times z$$

Фильтры



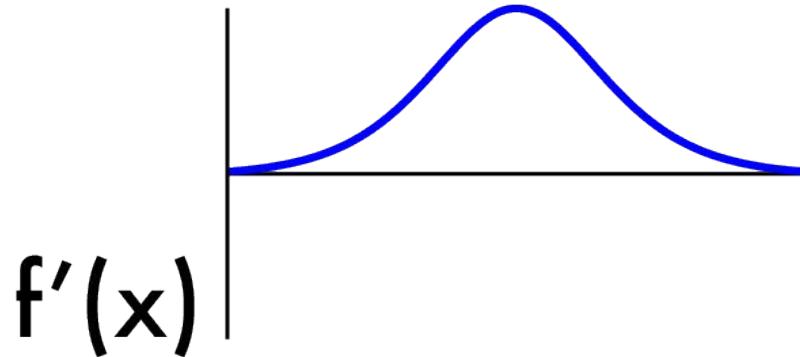
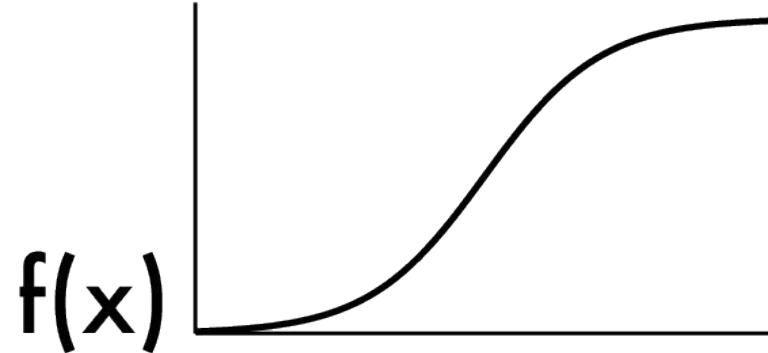
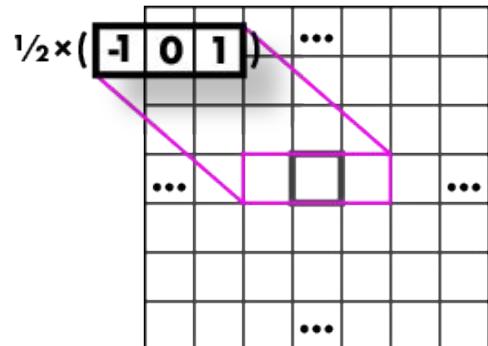
Что такое граница?

- Изображение это функция
- Границы - резкое изменение значения в этой функции



Производные изображения

- Вспомним:
 - $f'(a) = \lim_{h \rightarrow 0} \frac{f(a + h) - f(a)}{h}$.
- У нас нет настоящей функции,
нужно аппроксимировать
- Ставим $h = 2$

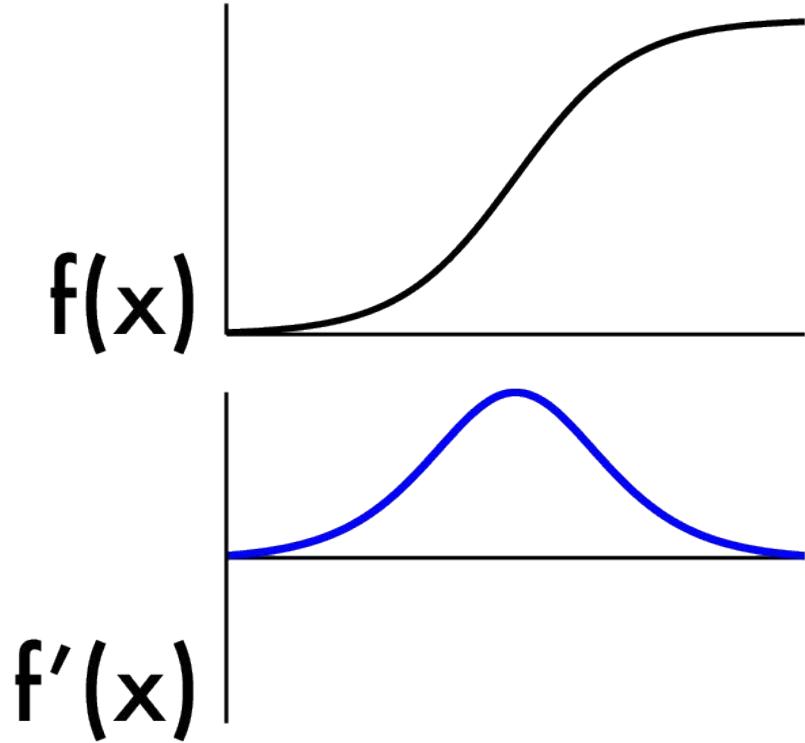


Производные изображения

- Вспомним:
 - $f'(a) = \lim_{h \rightarrow 0} \frac{f(a + h) - f(a)}{h}$.
- Хотим сглаживания!

$$\frac{1}{2} \times \left(\begin{bmatrix} -1 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \right)$$

$\begin{bmatrix} 1 & 2 & 1 \\ -1 & 0 & 1 \\ 1 & 2 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$



Лапласиан (вторая производная)!

- Пересекает ноль в точке экстремума

- Вспомним:

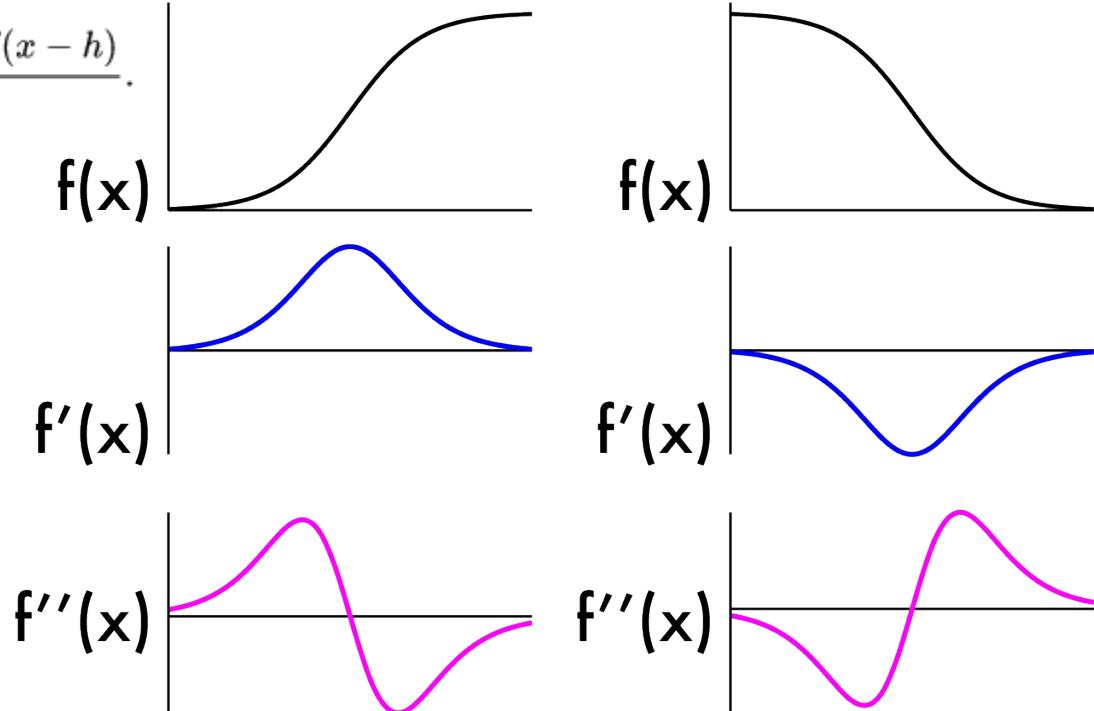
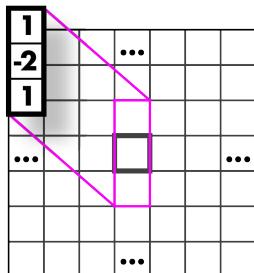
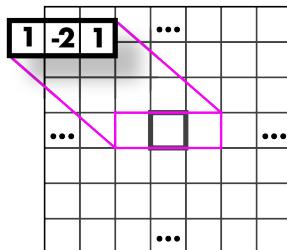
$$- f''(x) = \lim_{h \rightarrow 0} \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}.$$

- Лапласиан:

$$- \Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

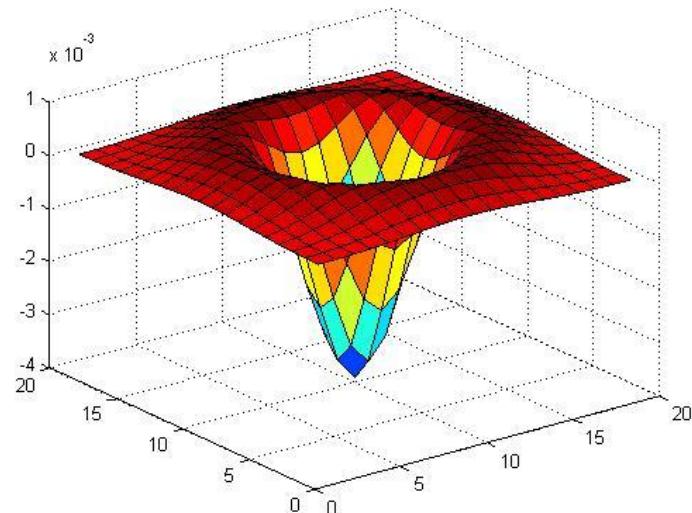
- И снова,

оцениваем $f''(x)$:



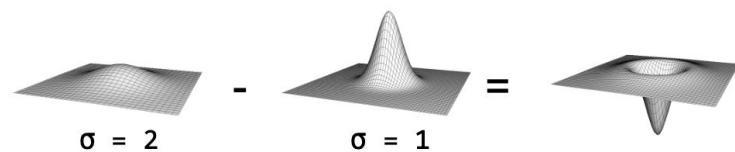
Лапласианы также чувствительны к шуме

- И снова, используем сглаживаем
- Можем использовать один совместный кернел
- Laplacian of Gaussian, LoG
- Хорошая аппроксимация при
5x5 - 9x9 кернелях



Difference of Gaussian (DoG)

- Гауссиана это фильтр низких частот
- Убивает компоненты с частотой $f > 1/\sigma$
- (g^*I) - низкочастотные компоненты
- $I - (g^*I)$ высокочастотные компоненты
- $g(\sigma_1)^*I - g(\sigma_2)^*I$
- Компоненты между этими частотами (“среднечастотные”)
- $g(\sigma_1)^*I - g(\sigma_2)^*I = [g(\sigma_1) - g(\sigma_2)]^*I$

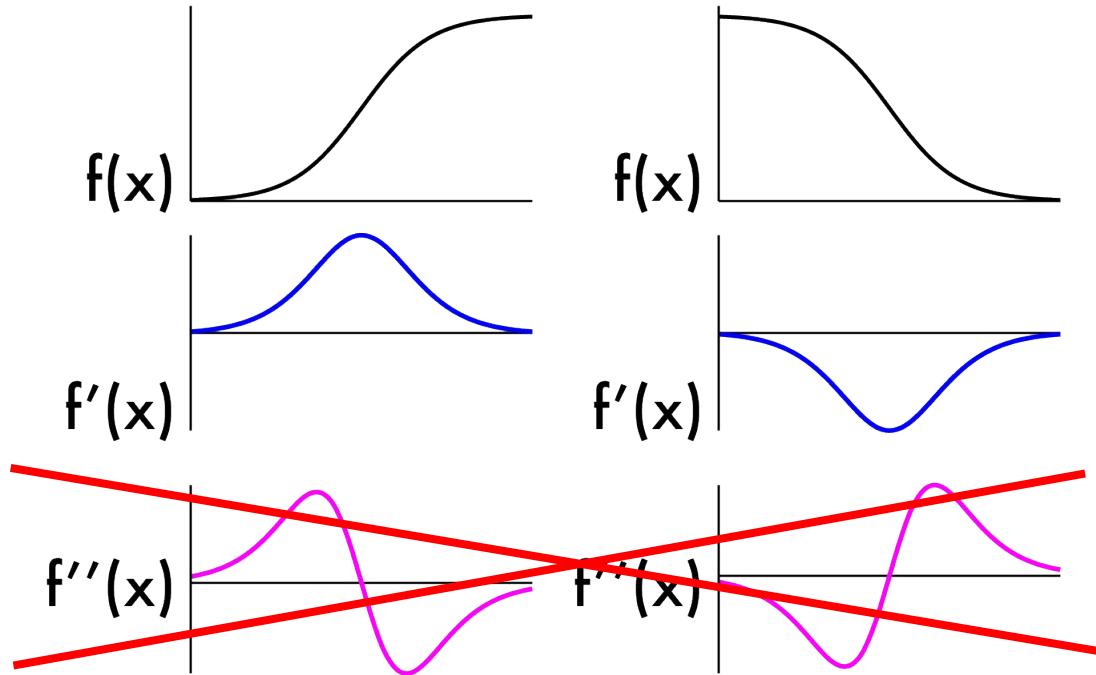


DoGs



Другой подход: магнитуда градиентов

- Не нужны вторые производные
- Просто берем магнитуду градиентов



Детектор границ Кэнни

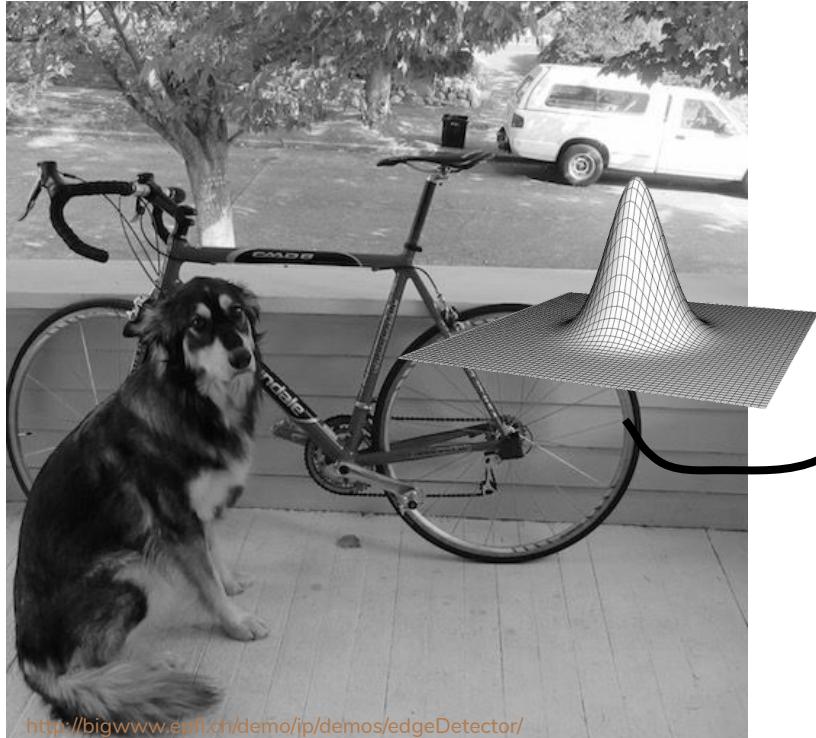
- Ваш первый image processing pipeline!
 - Олдскульное CV целиком состоит из пайплайнов

Алгоритм:

- Сгладить изображение
- Посчитать направление и магнитуду градиентов
- Non-maximum suppression
- Двойной трешхолдинг (strong, weak edges)
- Соединение компонент (гистерезис)

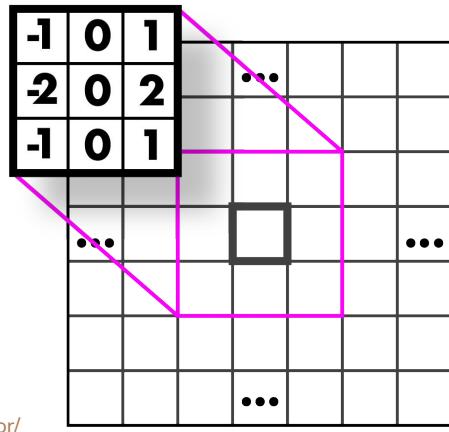
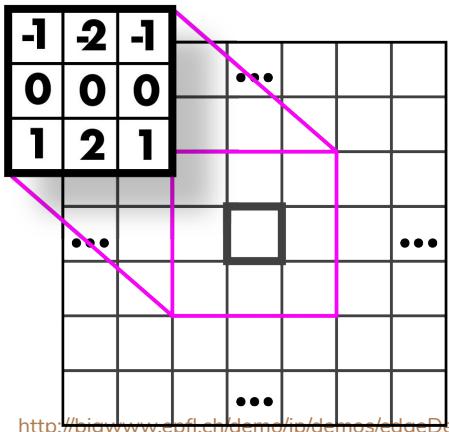
Сглаживание изображений

- Уже знаем как - гауссианы!

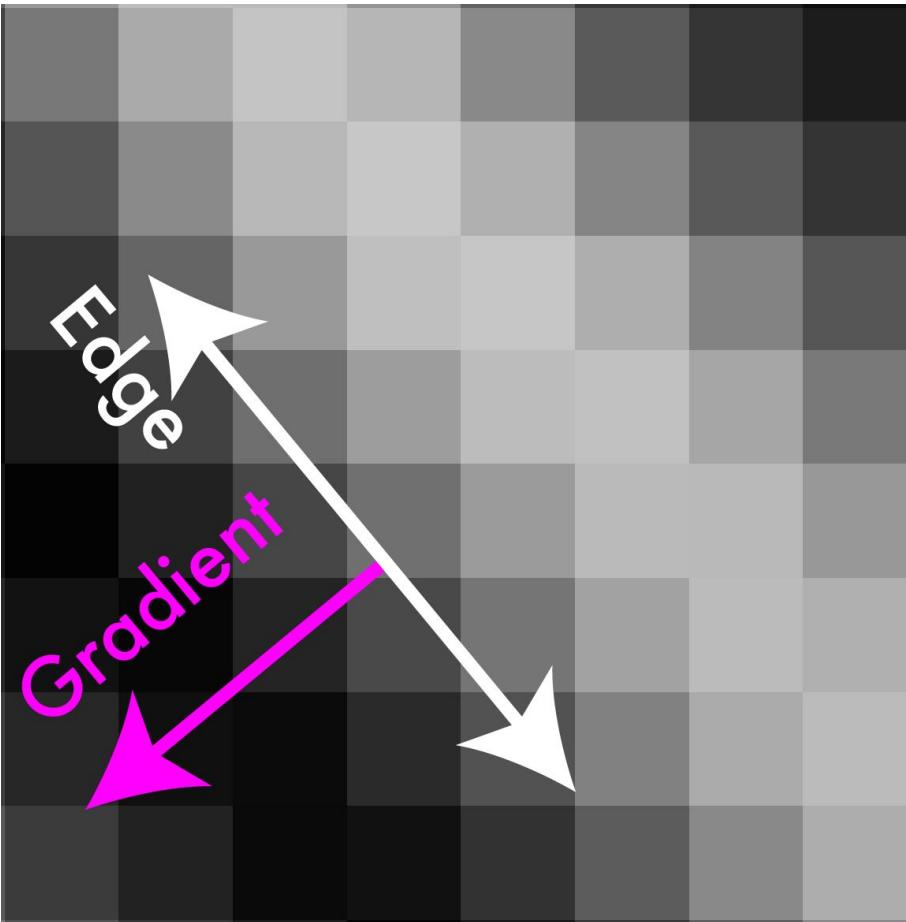


Направление и магнитуда градиентов

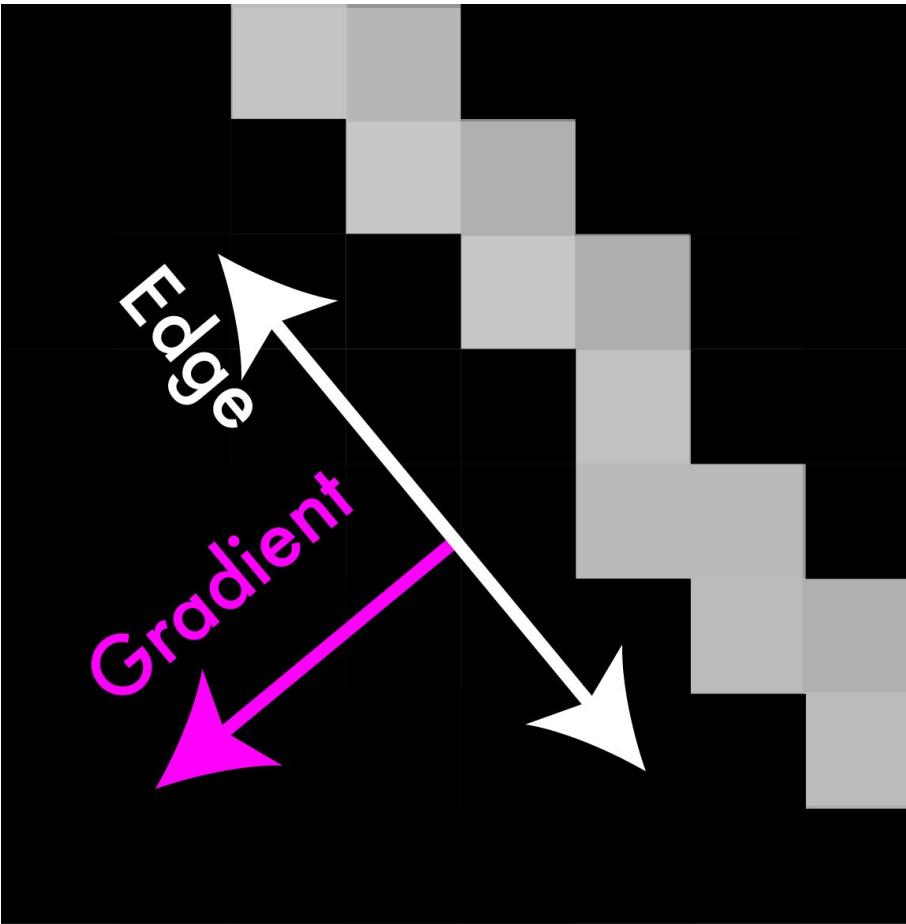
- Фильтр Собеля



Non-maximum suppression



Non-maximum suppression

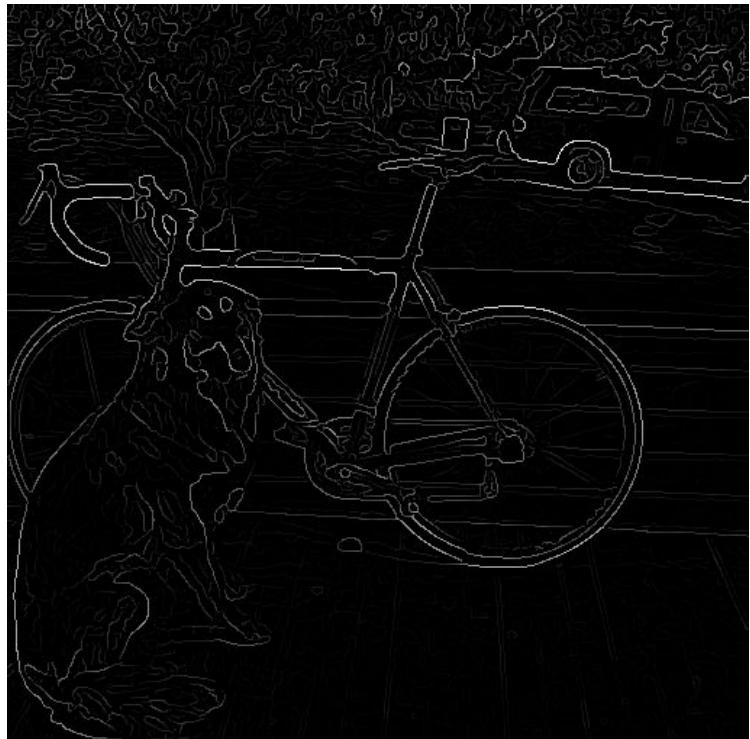


Non-maximum suppression



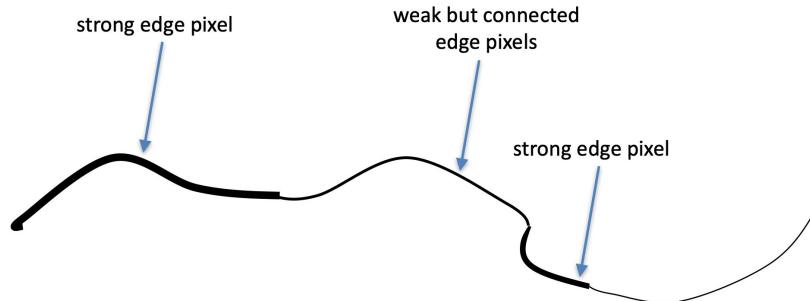
Трешхолдинг границ

- Хотим бинарную маску (граница/не граница)
- Все еще зашумлено
- 2 порога, 3 случая
 - $R > T$: strong edge
 - $R < T$, но $R > t$: weak edge
 - $R < t$: no edge
- Почему 2 порога?



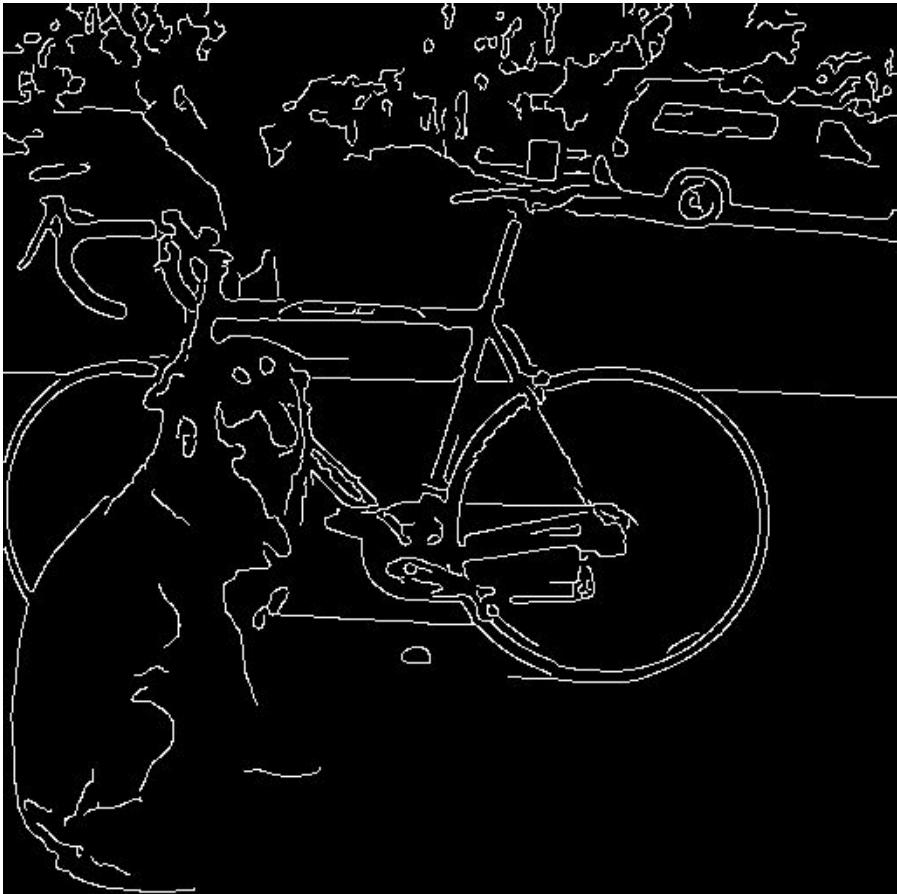
Соединяем их!

- strong edges - точно граница
- weak edges - граница, тогда и только тогда, когда соединен с strong edges
- Смотрим на соседние пиксели (8 ближайших)



Source: S. Seitz

Детектор Границ Кэнни



Признаки изображения

Признаки!

- Наиболее информативные регионы
- Способы описания этих регионов
- Полезно для:
 - Матчинга
 - Распознавания
 - Детектирования

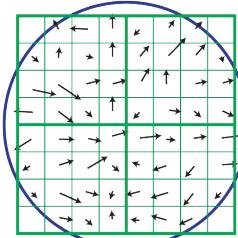
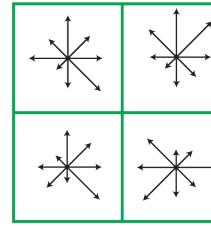
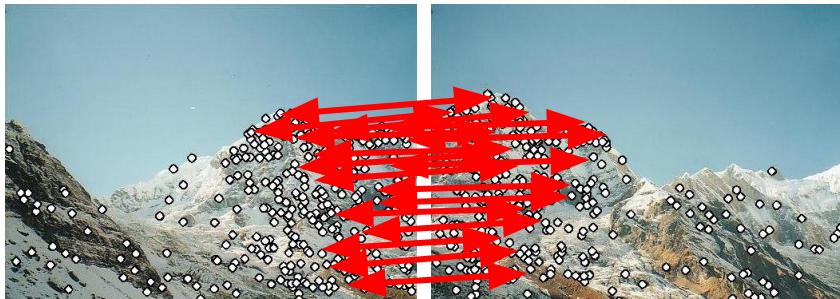
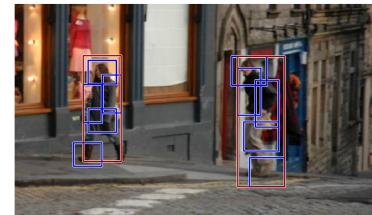


Image gradients



Keypoint descriptor



Что есть хороший признак?

- Хотим найти патчи в изображение, которые имеют какой-то смысл
- Для каждого объекта хотим найти уникальный для этого объекта патч
- Для панорам, хотим находить патчи которые можно легко найти и сопоставить между всеми изображениями
- Хорошие признаки уникальны!
 - Можем легко найти “тот же самый” признак
 - Не перепутать с другими признаками

Насколько близки два патча

- Сумма квадратов разности
- Изображения I, J
- $\sum_{x,y} (I(x,y) - J(x,y))^2$

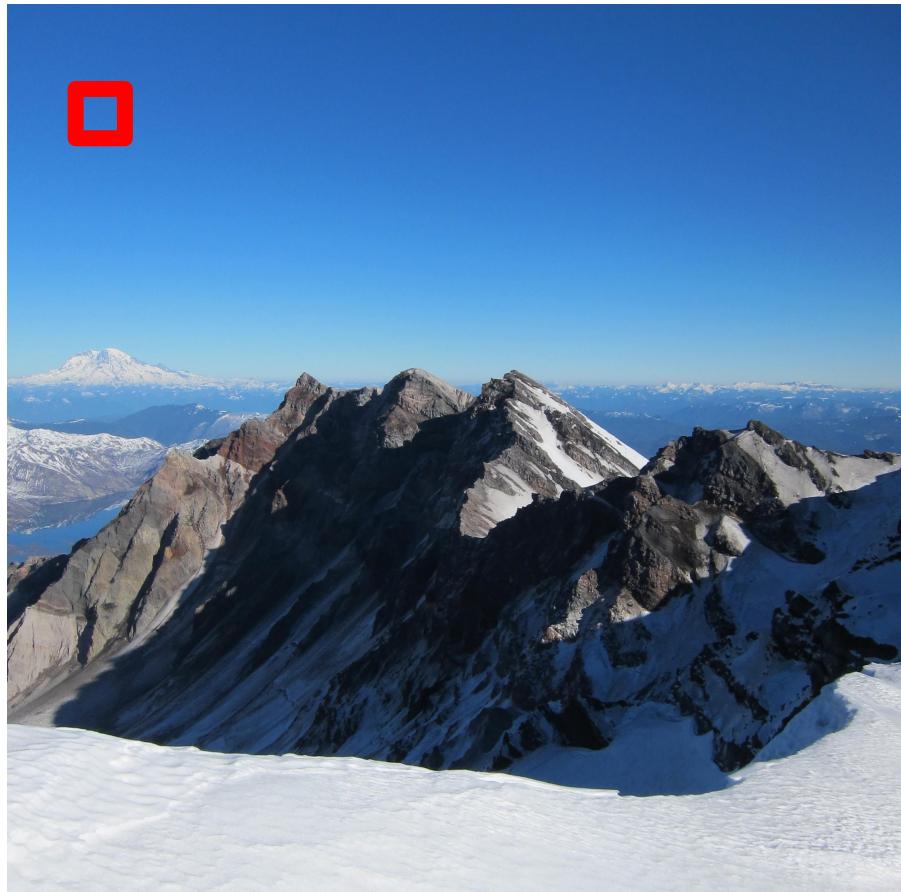
Как найти хороший признак

- Допустим мы делаем панораму
- Хотим патчи в одном изображения сматчить с патчами в другом
- Связь “один к одному”



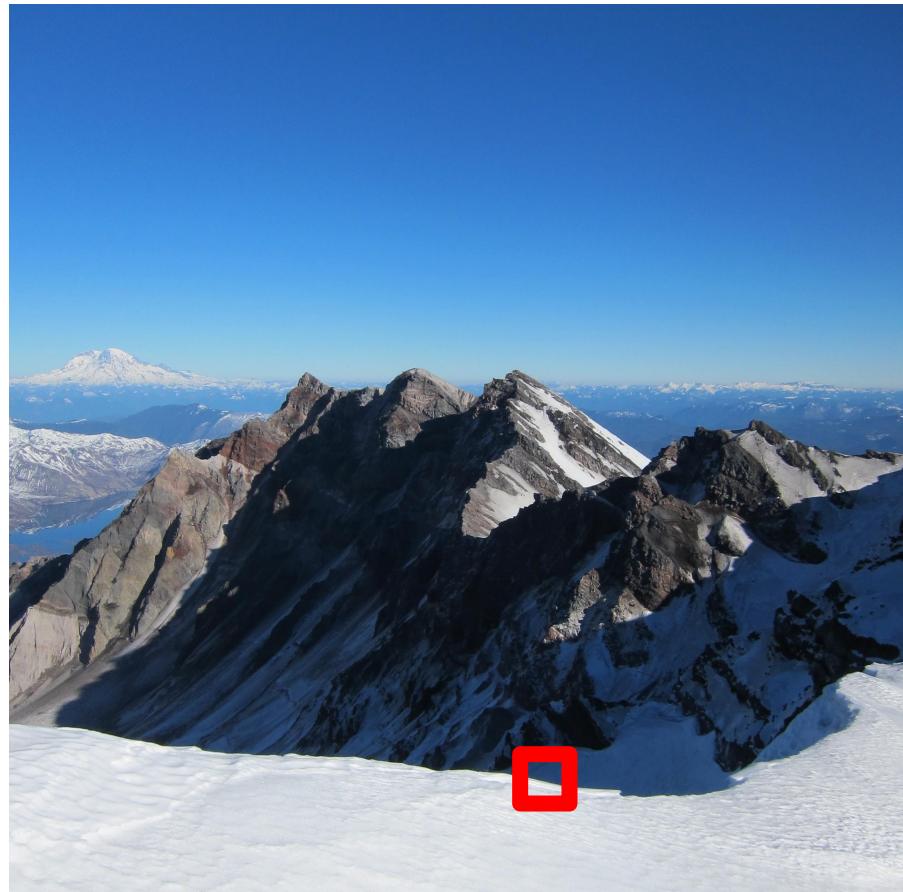
Как найти хороший признак

- Небо: плохо
 - Очень маленькая вариация
 - Неразличимо



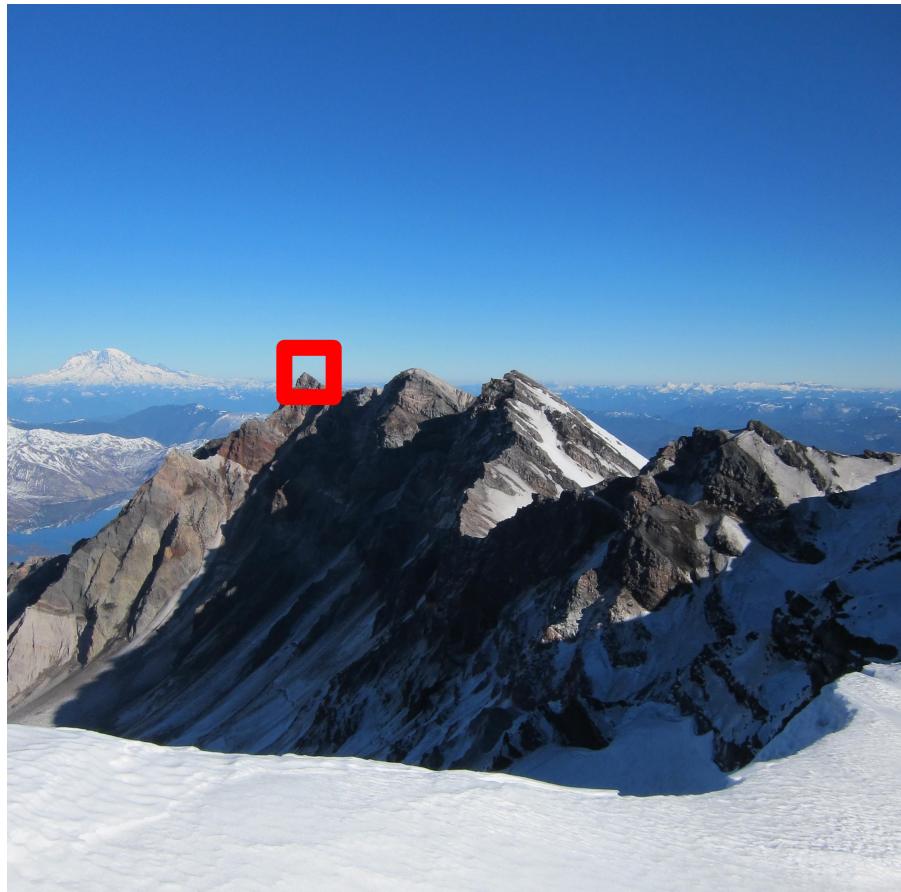
Как найти хороший признак

- Небо: плохо
 - Очень маленькая вариация
 - Неразличимо
- Граница: получше
 - Вариация в одном направлении
 - Может сматчиться с любым другим патчом вдоль этого направления



Как найти хороший признак

- Небо: плохо
 - Очень маленькая вариация
 - Неразличимо
- Граница: получше
 - Вариация в одном направлении
 - Может сматчиться с любым другим патчом вдоль этого направления
- Углы: супер!
 - Углы уникальны



Как найти хороший признак

- Хотим патч уникальный в изображении
- Можем посчитать расстояние между патчем и каждым другим патчем
- Много вычислений!

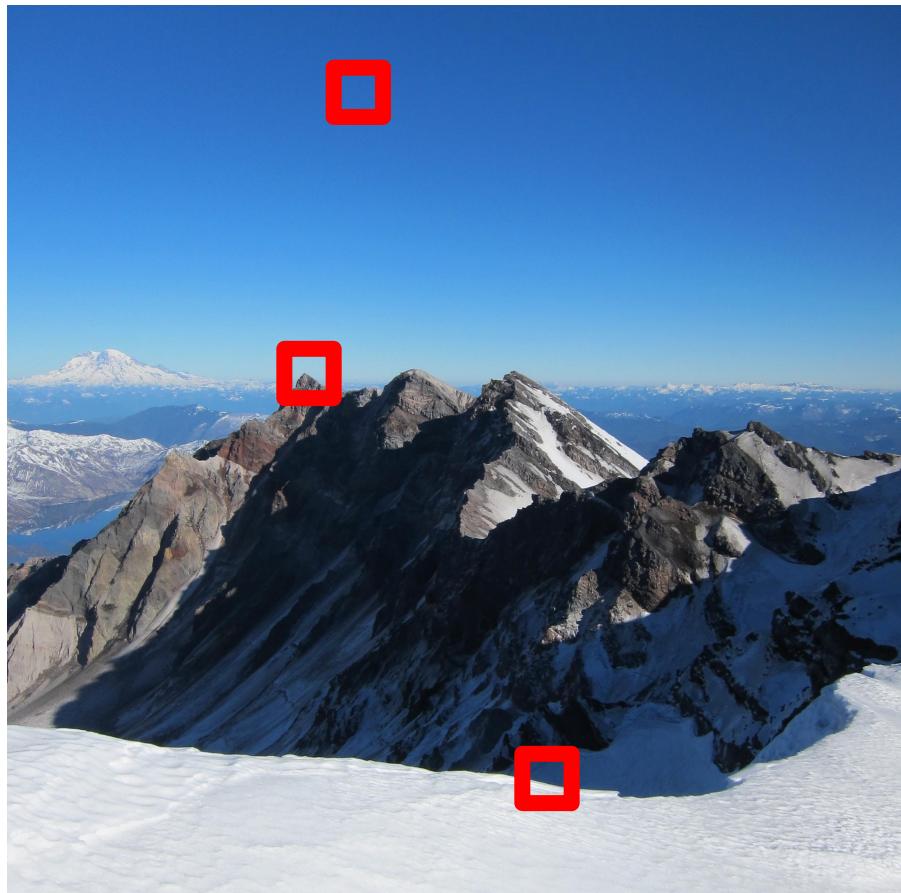
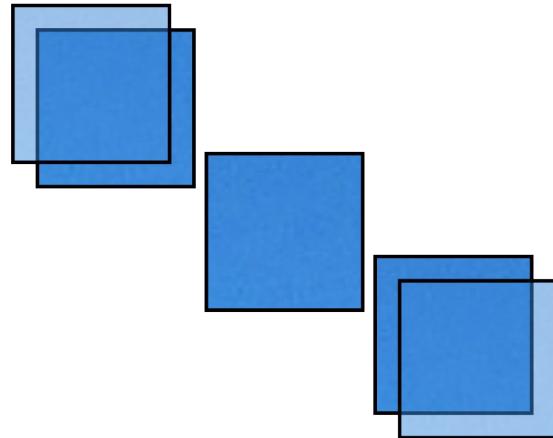
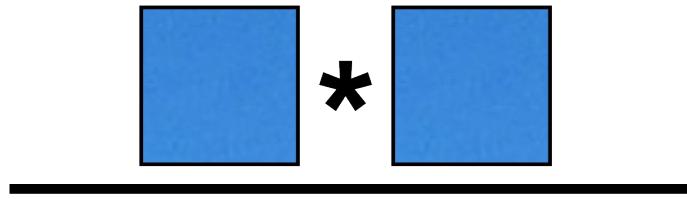


Как найти хороший признак

- Хотим патч уникальный в изображении
- Можем посчитать расстояние между патчем и каждым другим патчем
- Много вычислений!
- Воспользуемся автокорреляцией:
 - Насколько хорошо изображение матчится со смещенной версией самого себя?
- $\sum_d \sum_{x,y} (I(x,y) - I(x+d_x, y+d_y))^2$
- Мера непохожести

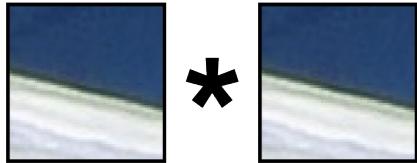
Мера непохожести

Небо: везде низкая

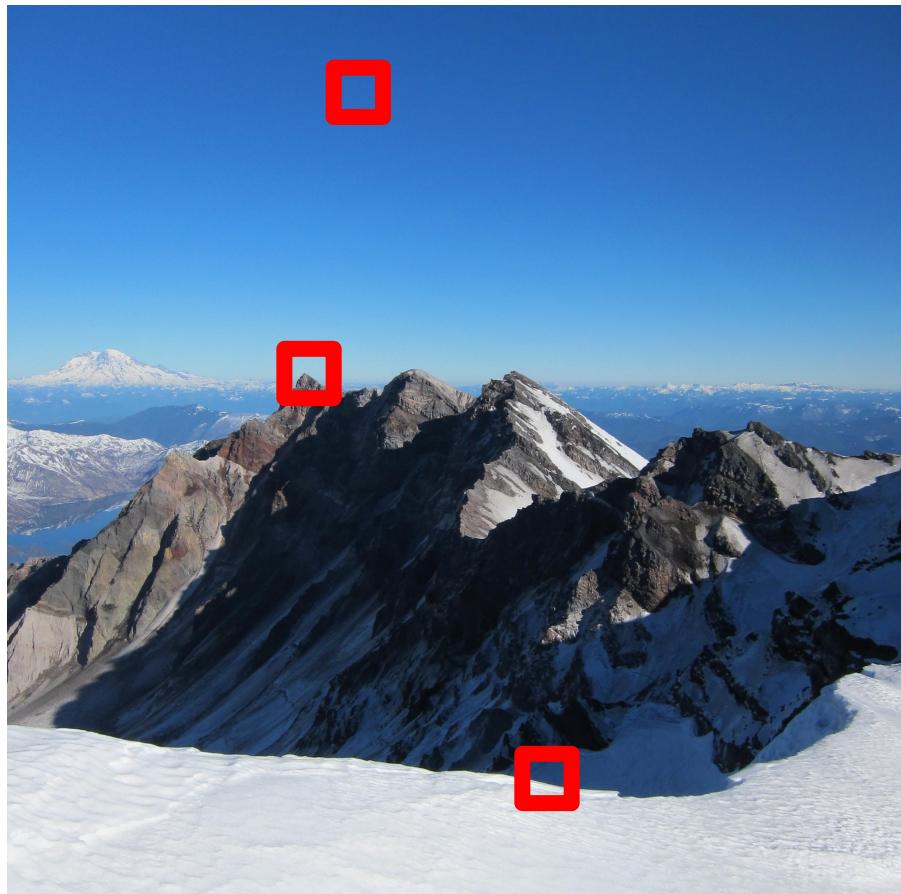
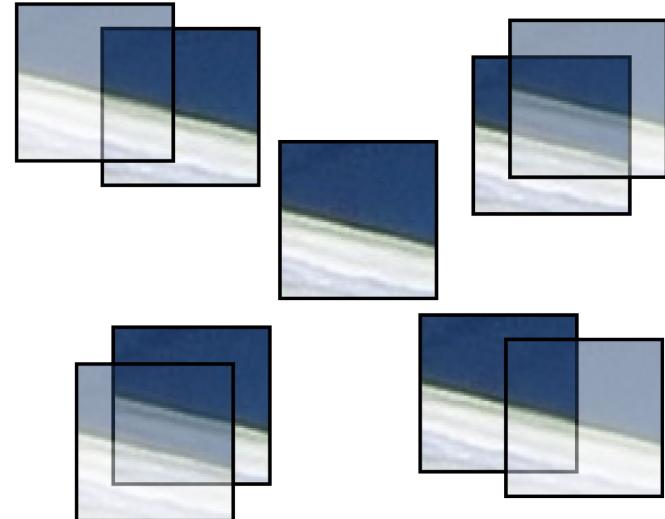


Мера непохожести

Край: низкая вдоль границы



*

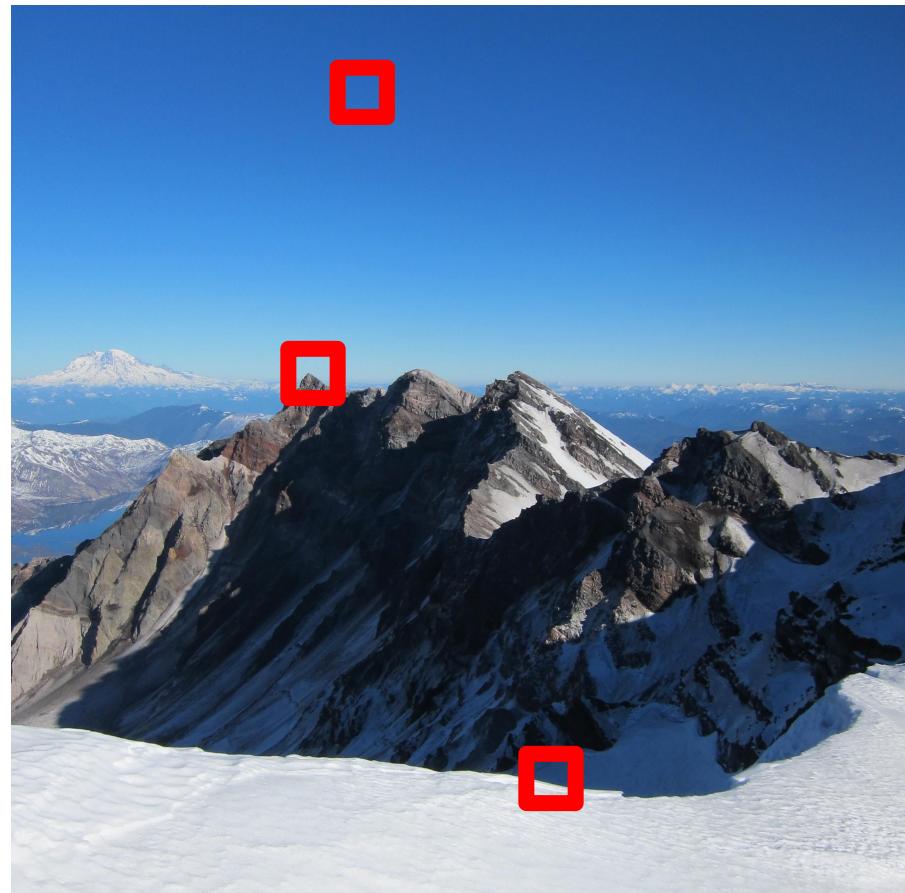
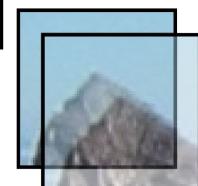
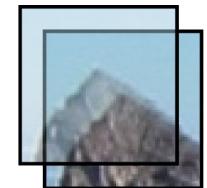


Мера непохожести

Угол: везде высокая



*

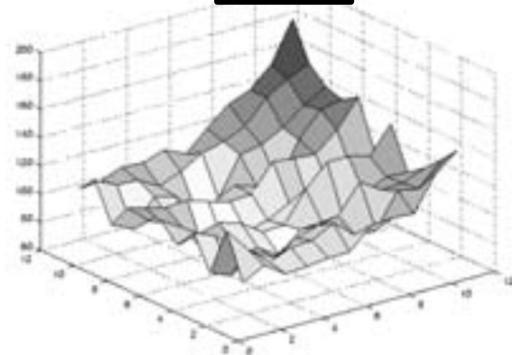
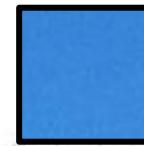
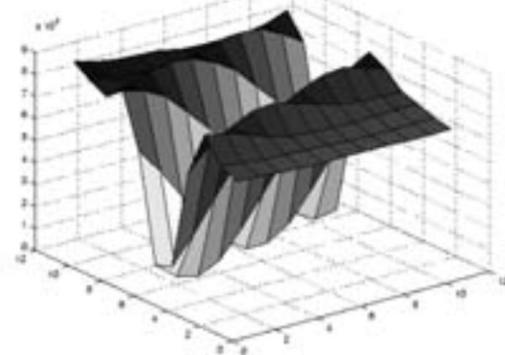
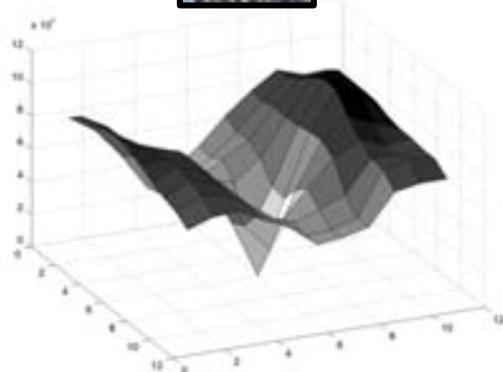


Мера непохожести

Небо: везде низкая

Край: низкая вдоль границы

Угол: везде высокая



Мера непохожести

- $\sum_d \sum_{x,y} (I(x,y) - I(x+d_x, y+d_y))^2$
- Все еще много вычислений
- Нужна аппроксимация!

Аппроксимируем

- Смотрим на градиенты I_x и I_y
- Если они около нуля, ничего интересного нету
 - Небольшие значения меры непохожести
- Если градиенты в одном направлении - граница
 - Все еще небольшие значения меры непохожести
- Если градиенты большие в нескольких направлениях - угол!
 - Большие значения меры непохожести, хороший регион!

Аппроксимируем

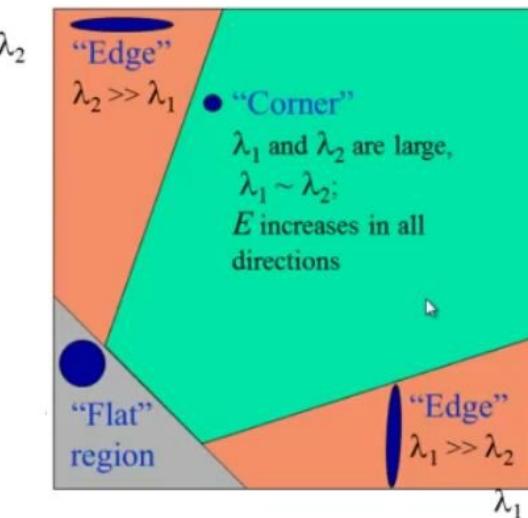
- Как понять, что происходит с градиентами?
- Собственные значения матрицы!
- Определим structure matrix

$$S_w[p] = \begin{bmatrix} \sum_r w[r](I_x[p-r])^2 & \sum_r w[r]I_x[p-r]I_y[p-r] \\ \sum_r w[r]I_x[p-r]I_y[p-r] & \sum_r w[r](I_y[p-r])^2 \end{bmatrix}$$

- Не так сложно как выглядит, просто взвешенная сумма градиентов соседей

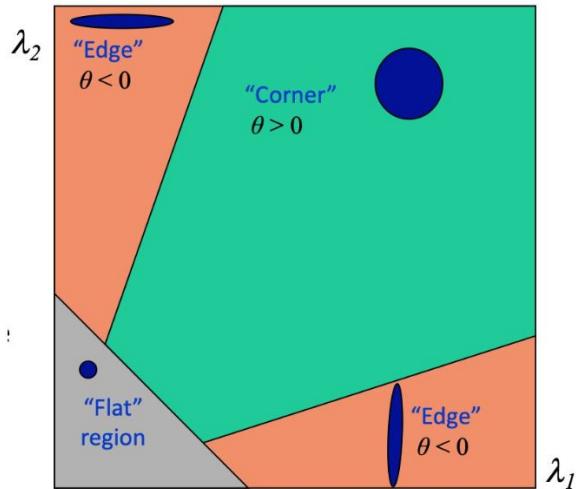
Структурная матрица

- Взвешенная сумма градиентов соседей
 - $\begin{vmatrix} \sum_i w_i |x(i)|_x(i) & \sum_i w_i |x(i)|_y(i) \\ \sum_i w_i |x(i)|_y(i) & \sum_i w_i |y(i)|_y(i) \end{vmatrix}$
 - $\begin{vmatrix} \sum_i w_i |x(i)|_x(i) & \sum_i w_i |y(i)|_y(i) \\ \sum_i w_i |y(i)|_x(i) & \sum_i w_i |y(i)|_y(i) \end{vmatrix}$
- Можем использовать Гауссиану (опять)
- Собственные числа этой матрицы описывают поведение градиентов в окруже
- λ_1 и λ_2 - собственные числа
 - λ_1 и λ_2 оба маленькие: flat region
 - $\lambda_1 \gg \lambda_2, \lambda_2 \gg \lambda_1$: граница
 - λ_1 и λ_2 оба большие: угол



Функция отклика угла

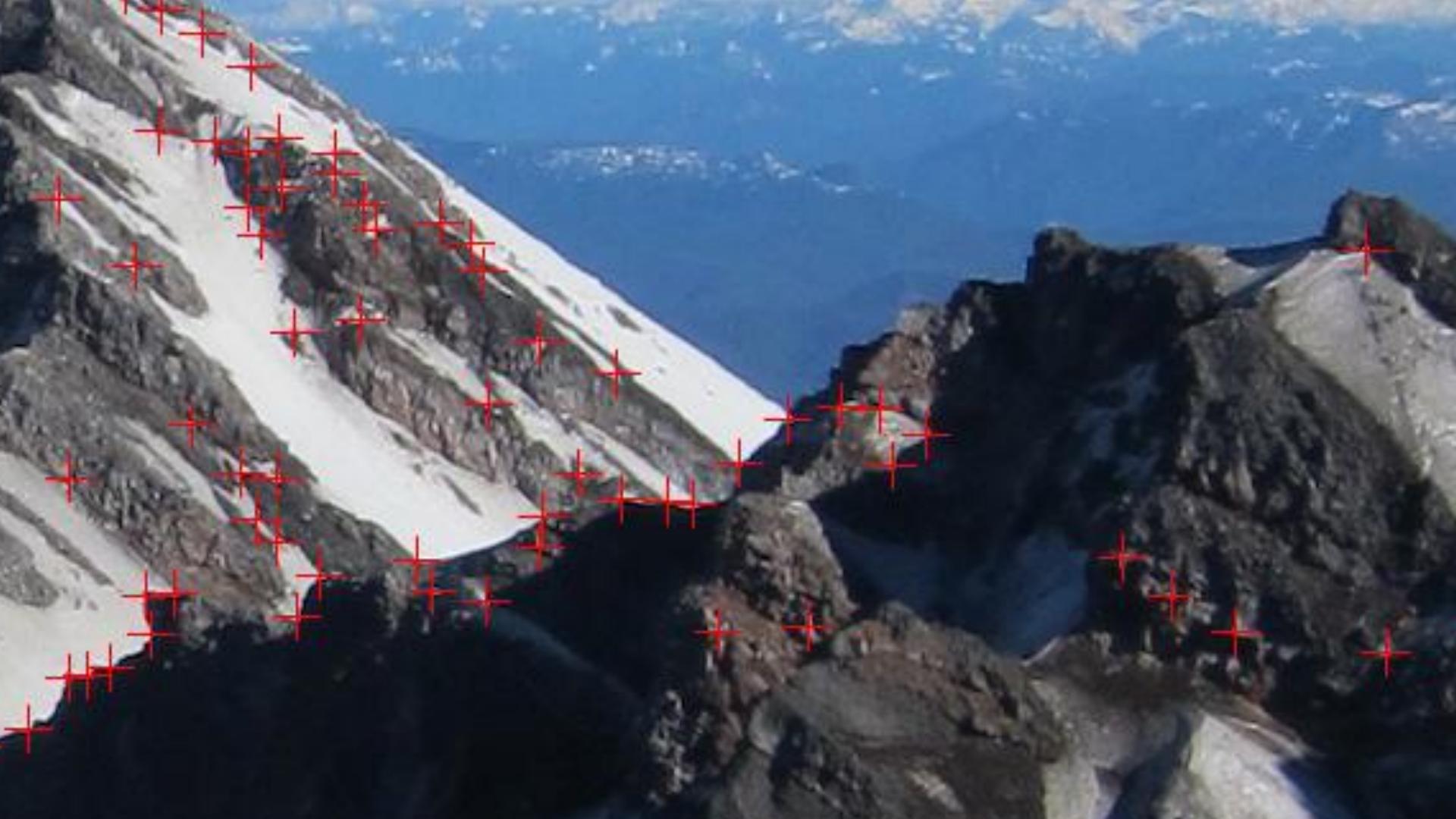
- Можно напрямую посчитать λ_1 и λ_2
 - Нужно брать квадрат числа
- Несколько методов:
 - $\det(S) = \lambda_1 * \lambda_2$
 - $\text{trace}(S) = \lambda_1 + \lambda_2$
- $\Theta = \det(S) - \alpha \text{trace}(S)^2 = \lambda_1 \lambda_2 - \alpha(\lambda_1 + \lambda_2)^2$



Детектор углов Харриса

- Вычислить производные I_x и I_y
- Вычислить 3 вспомогательные матрицы \mathbf{I}_{xx} , \mathbf{I}_{yy} , \mathbf{I}_{xy}
- Посчитать взвешенную сумму
 - Взвешенная?
 - Фильтр Гаусса!
- Посчитать функцию отклика угла
- Non-max suppression (как в детекторе границ Кэнни)





Нашли углы, что дальше?

- Нужно сматчить их друг с другом
- Найти преобразование одного изображения в пространство другого изображения



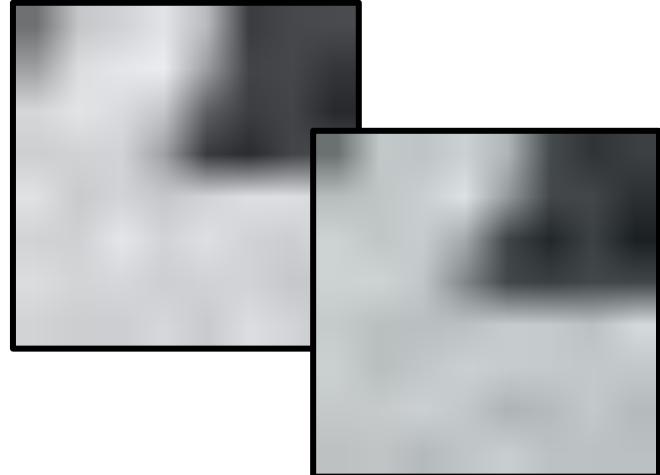
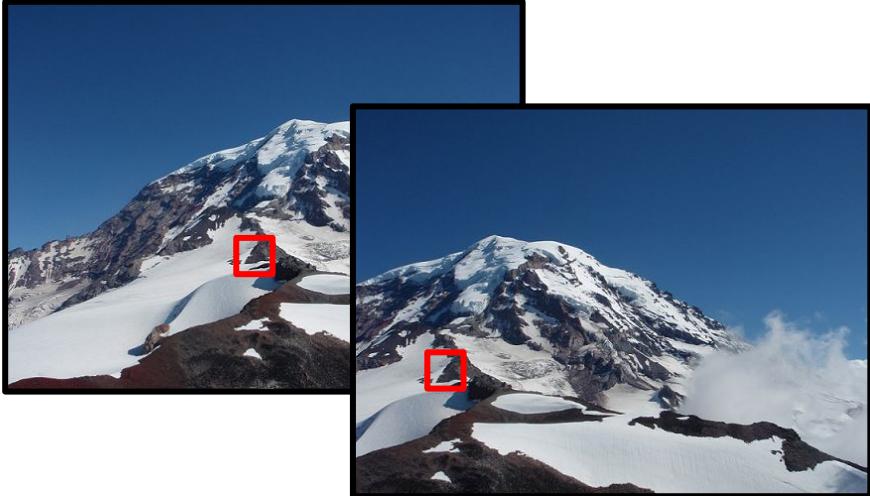
Нашли углы, что дальше?

- Нужно сматчить их друг с другом
 - Как их матчить? Как описывать регионы?
- Найти преобразование одного изображения в пространство другого изображения
 - Как преобразовывать изображение?
 - Как имея матчи найти преобразование?



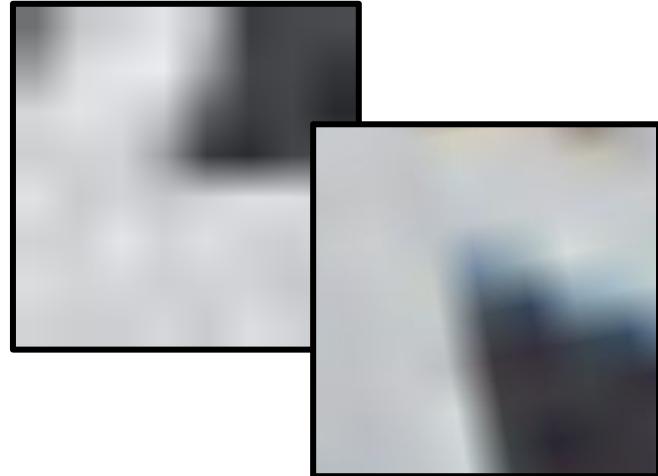
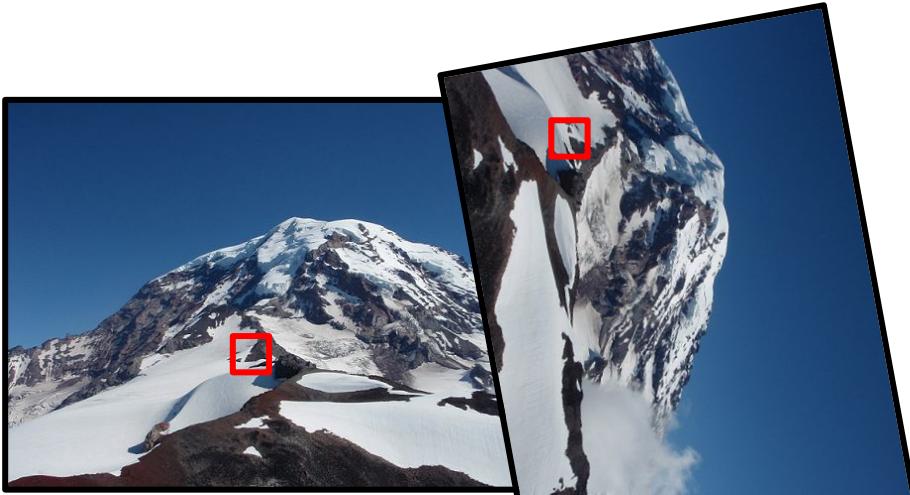
Дескрипторы

- Мы хотим как-то описывать регионы изображения
- Можно просто брать значения пикселей в этом регионе
- Матчить просто, метрика расстояния:
 - $\sum_{x,y} (I(x,y) - J(x,y))^2$
 - Какие тут возникают проблемы?



Дескрипторы

- Мы хотим как-то описывать регионы изображения
- Можно просто брать значения пикселей в этом регионе
- Матчить просто, метрика расстояния:
 - $\sum_{x,y} (I(x,y) - J(x,y))^2$
 - Какие тут возникают проблемы?

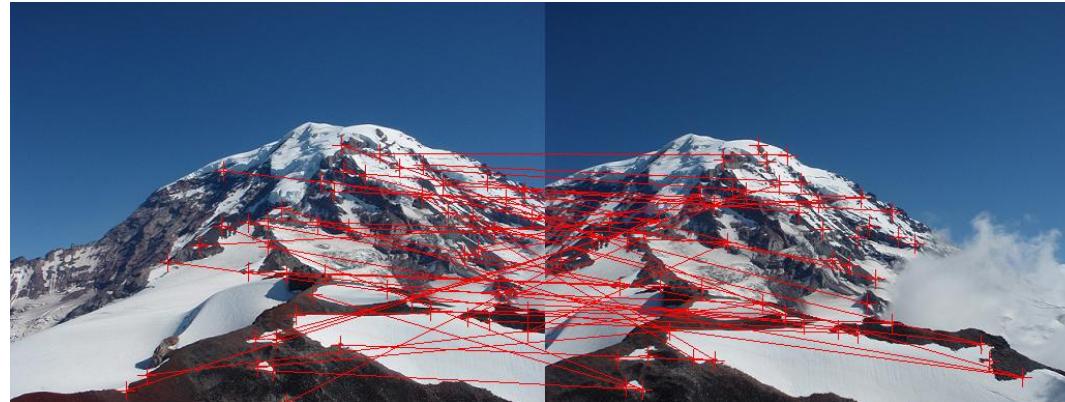


Дескрипторы

- Мы хотим как-то описывать регионы изображения
- Можно просто брать значения пикселей в этом регионе
- Матчить просто, метрика расстояния:
 - $\sum_{x,y} (I(x,y) - J(x,y))^2$
 - Какие тут возникают проблемы?
- Дескрипторы могут быть сложнее
 - Использовать градиенты
 - Нормализовывать
 - HoG и SIFT

Дескрипторы

- Допустим мы уже построили хороший дескриптор
- Проходим по регионам в одном изображении
 - Находим лучший матч в другом изображении
- Делаем что-то с этими матчами
 - ???



Как преобразовывать изображения?

- Хотим преобразовать одно изображение на другое
- Много различных типов преобразования изображения
 - Вложенная иерархия преобразования
- Нужно немножко вспомнить линейную алгебру

Как преобразовывать изображения?

- **x** - это пиксель (его координата в изображении):
 - В матричных терминах:

$$\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$$

Новая система координат

- Отобразить точки с одного изображения на другое
- Мы можем использовать матричные операции
- Имея точку x , получить точку x' используя M

$$\mathbf{x}' = \mathbf{M} \mathbf{x}$$

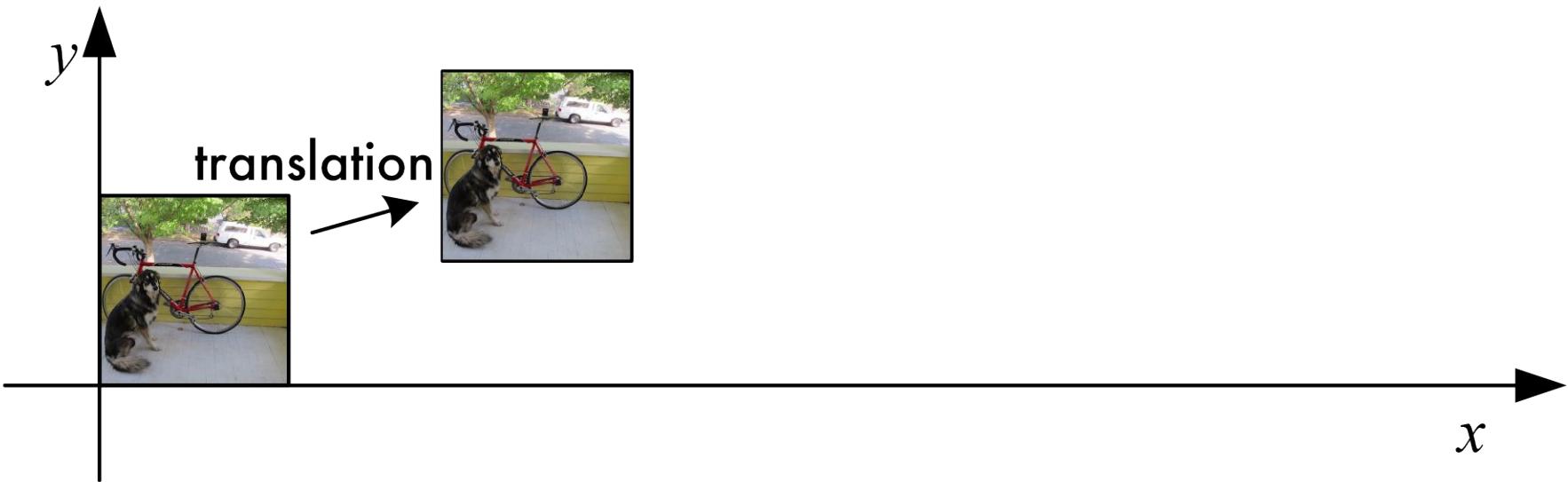
Масштабирование как матричная операция

- Отобразить точки с одного изображения на другое
- Мы можем использовать матричные операции
- Имея точку x , получить точку x' -> используем M

$$\mathbf{x}' = S \mathbf{x} \quad \mathbf{x}' = \begin{bmatrix} s & 0 \\ 0 & s \end{bmatrix} \mathbf{x}$$

Сдвиг

- $x' = M x$
 - Хотим сдвинуть на dx и на dy
 - Как выбрать M ?
 - Матрицей 2×2 этого не сделать



Сдвиг

- \bar{x} это x с добавленной точкой 1
- *Дополненный вектор*

$$\bar{x} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Сдвиг

- $\bar{\mathbf{x}}$ это \mathbf{x} с добавленной точкой 1
- *Дополненный вектор*
- Теперь сдвиг - это просто

$$\bar{\mathbf{x}} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\mathbf{x}' = [\mathbf{I} \quad \mathbf{t}] \bar{\mathbf{x}}$$

I - единичная матрица

Может быть любого размера

$$I = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ & \dots & & & \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix} \quad I_{2 \times 2} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad I_{3 \times 3} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Сдвиг

- $\bar{\mathbf{x}}$ это \mathbf{x} с добавленной точкой 1
- *Дополненный вектор*
- Теперь сдвиг - это просто

$$\bar{\mathbf{x}} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\mathbf{x}' = \begin{bmatrix} 1 & 0 & dx \\ 0 & 1 & dy \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\mathbf{x}' = [I \ t] \bar{\mathbf{x}}$$

Сдвиг

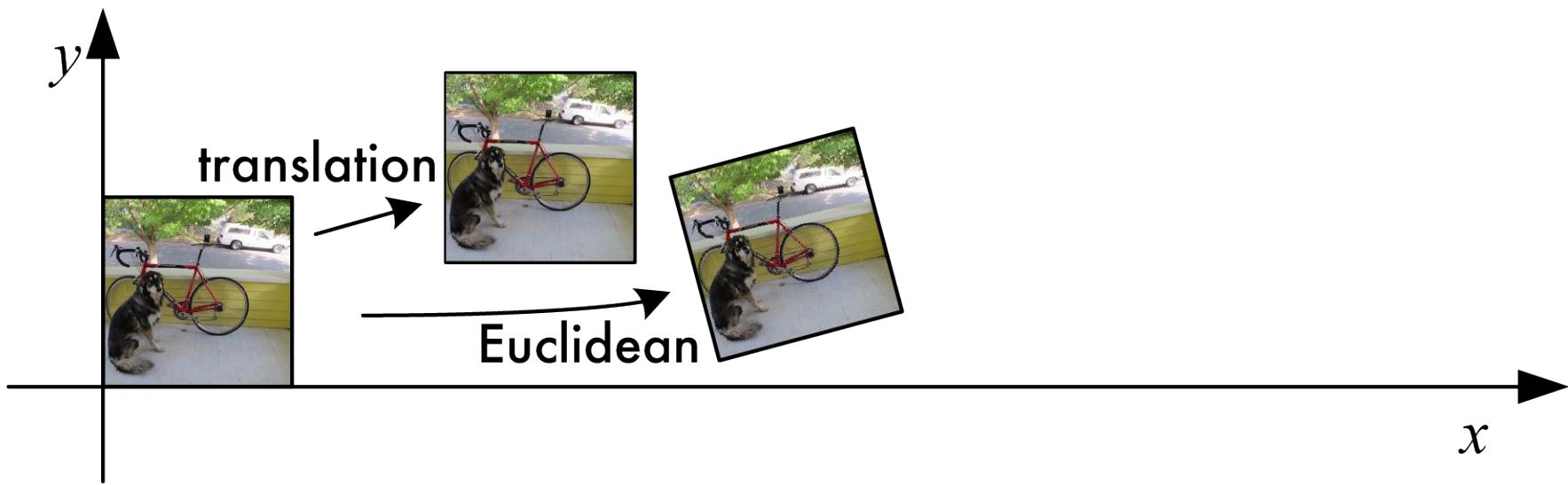
- $\bar{\mathbf{x}}$ это \mathbf{x} с добавленной точкой 1
- *Дополненный вектор*
- Теперь сдвиг - это просто
- $x' = 1*x + 0*y + dx*1$
- $y' = 0*x + 1*y + dy*1$

$$\bar{\mathbf{x}} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\mathbf{x}' = \begin{bmatrix} 1 & 0 & dx \\ 0 & 1 & dy \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\mathbf{x}' = [I \ t] \bar{\mathbf{x}}$$

Ортогональное (euclidean) преобразование



Евклидово преобразование: поворот + сдвиг

- Хотим одновременно повернуть и сдвинуть
- Все еще матричная операция

$$\mathbf{x}' = [\mathbf{R} \ \mathbf{t}] \bar{\mathbf{x}}$$

Евклидово преобразование: поворот + сдвиг

- Хотим одновременно повернуть и сдвинуть
- Все еще матричная операция
- R матрица, t вектор сдвига

$$\mathbf{x}' = [\mathbf{R} \quad \mathbf{t}] \bar{\mathbf{x}}$$

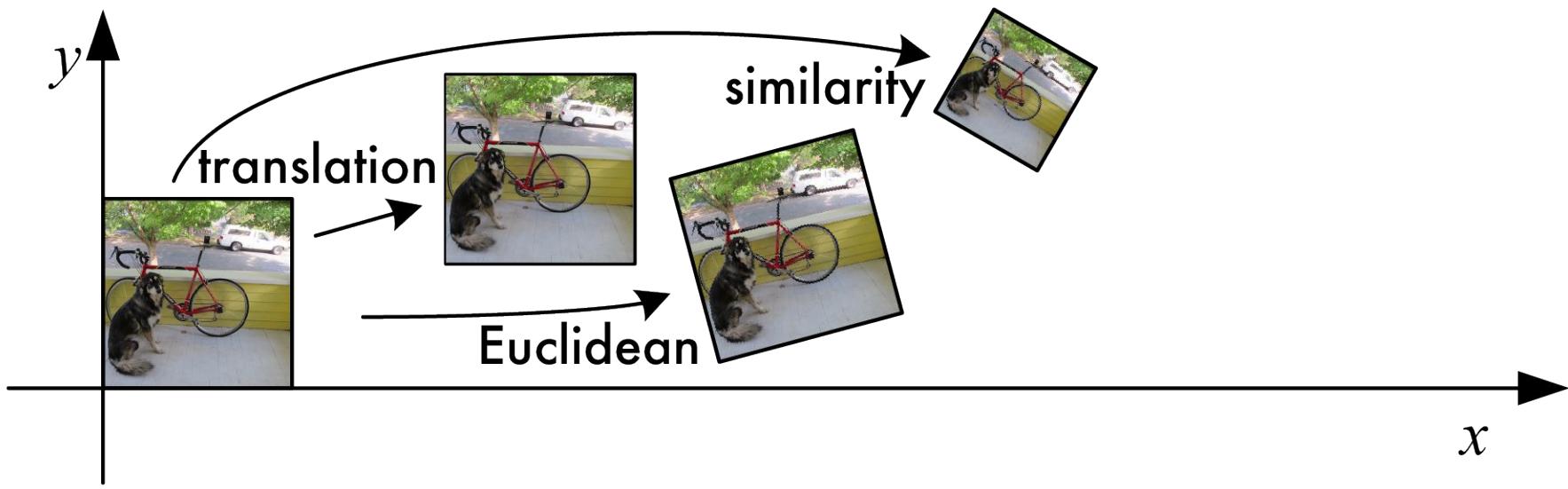
$$\mathbf{R} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

Евклидово преобразование: поворот + сдвиг

- Хотим одновременно повернуть и сдвинуть
- Все еще матричная операция
- R матрица, t вектор сдвига

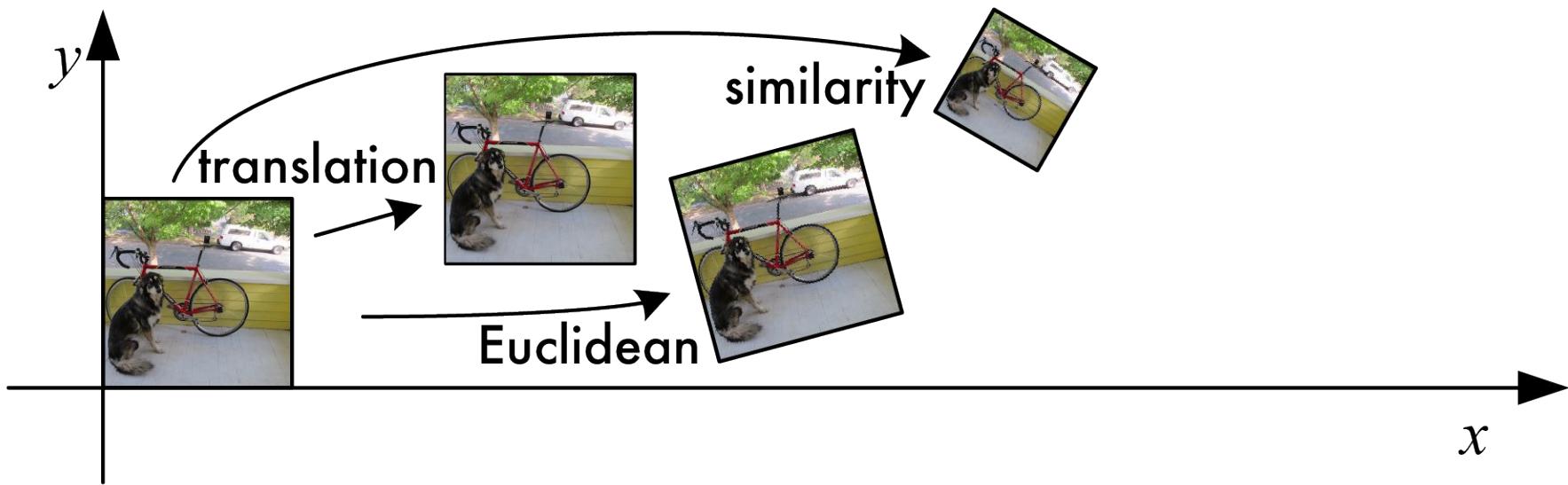
$$\mathbf{x}' = \begin{bmatrix} \cos\theta & -\sin\theta & dx \\ \sin\theta & \cos\theta & dy \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad \mathbf{x}' = [R \ t] \bar{\mathbf{x}}$$
$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

Подобие (similarity): масштабирование, поворот, сдвиг



Подобие (similarity): масштабирование, поворот, сдвиг

$$\mathbf{x}' = [s\mathbf{R} \quad \mathbf{t}] \bar{\mathbf{x}}$$

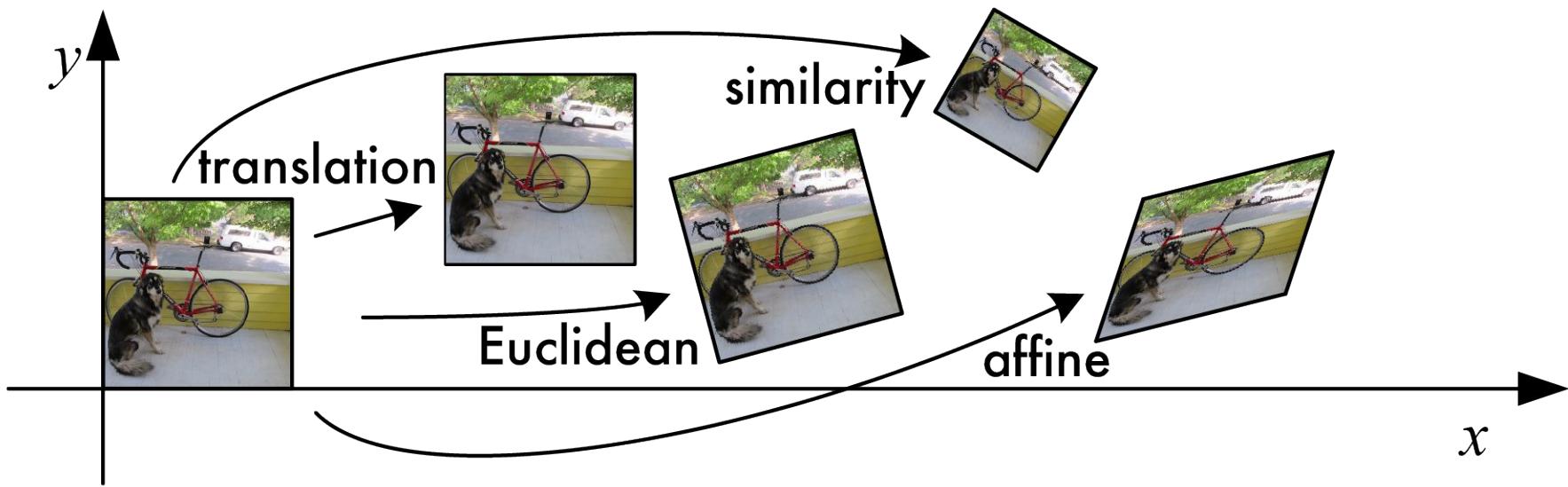


Подобие (similarity): масштабирование, поворот, сдвиг

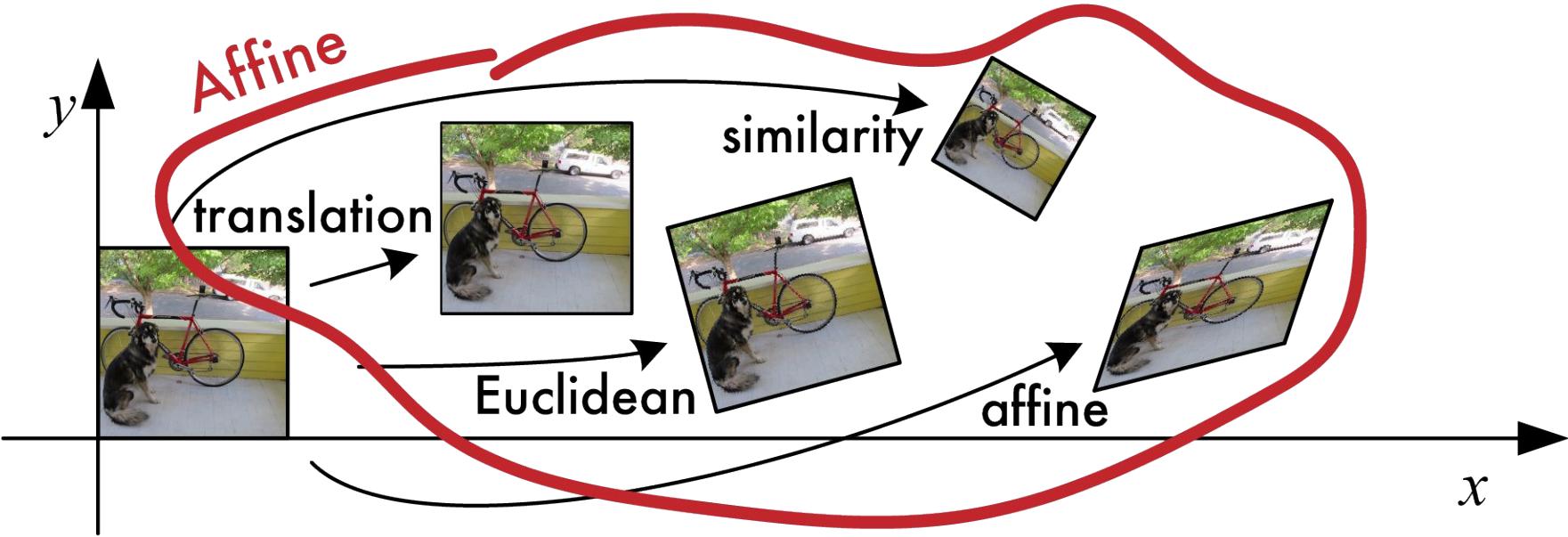
$$\mathbf{x}' = [s\mathbf{R} \quad \mathbf{t}] \bar{\mathbf{x}}$$

$$\mathbf{x}' = \begin{bmatrix} a & -b & dx \\ b & a & dy \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Аффинное: масштабирование, поворот, сдвиг, искажение

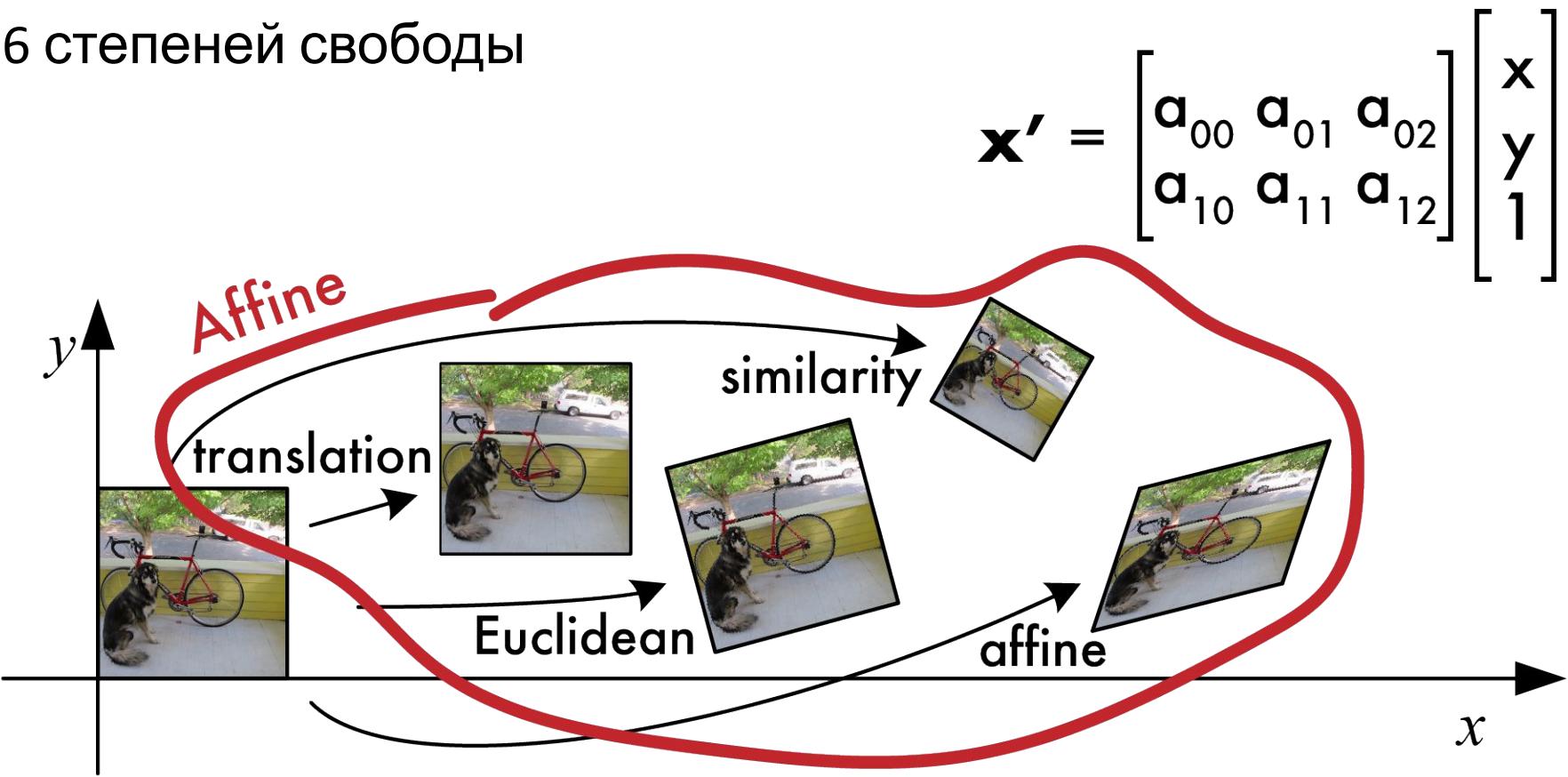


Аффинное: масштабирование, поворот, сдвиг, искажение



Аффинное: масштабирование, поворот, сдвиг, искажение

6 степеней свободы



Композиция - тоже аффинное преобразование

Допустим мы хотим сдвинуть, затем повернуть, затем снова сдвинуть, затем масштабировать

$$\mathbf{x}' = \mathbf{S} \mathbf{t} \mathbf{R} \mathbf{t} \bar{\mathbf{x}} = \mathbf{M} \bar{\mathbf{x}},$$

Если $\mathbf{M} = (\mathbf{S} \mathbf{t} \mathbf{R} \mathbf{t})$

\mathbf{M} - это все еще аффинное преобразование

Но это матрицы 2×3 , как мы можем их перемножать?

Добавим строку к матрицам

$$\bar{\mathbf{x}}' = \begin{bmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

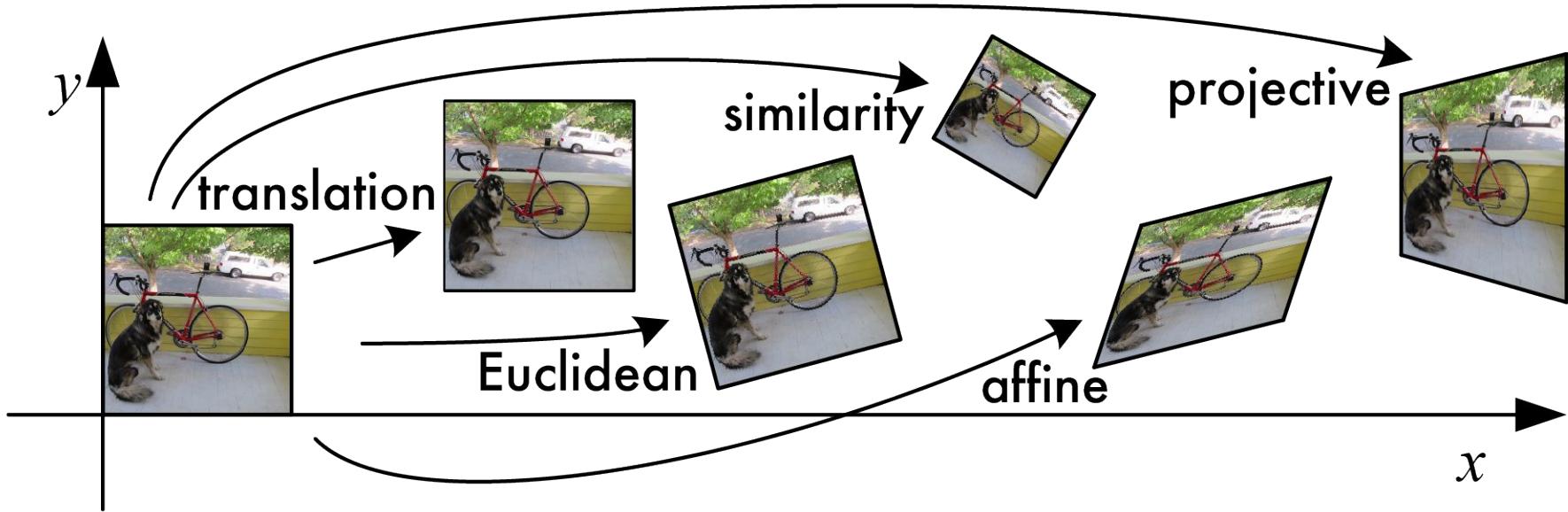
$$\bar{\mathbf{x}}' = \begin{bmatrix} \cos\theta & -\sin\theta & dx \\ \sin\theta & \cos\theta & dy \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\bar{\mathbf{x}}' = \begin{bmatrix} a & -b & dx \\ b & a & dy \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\bar{\mathbf{x}}' = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Проективное преобразование

- АКА гомология
- Аффинное - 2×3 матрицы, а проективное - ...?



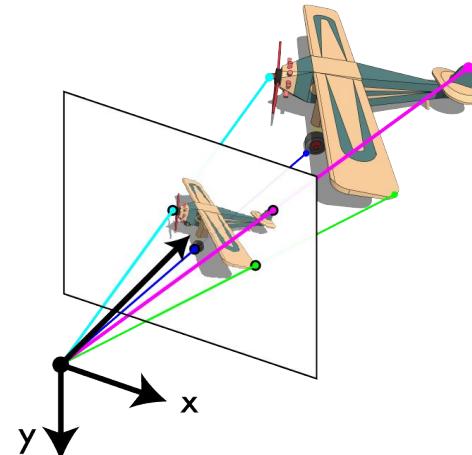
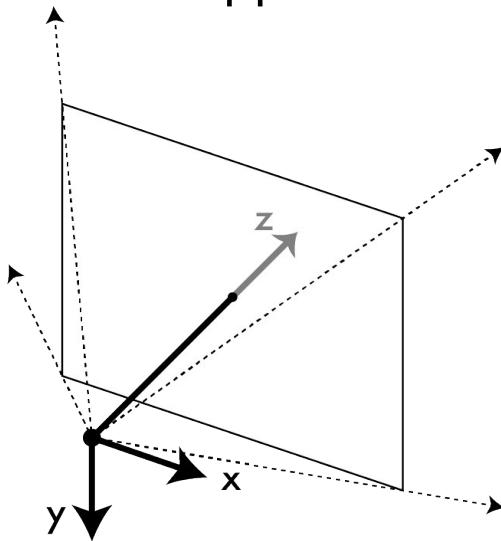
Нужна новая система координат

- Однородная система координат
 - Полезно поскольку соответствует модели камере
- Каждая точка в 2d - это вектор в 3d
- Они одинаковы с точностью до множителя
- Нужно нормализовать, чтобы вернуться в 2d

$$\tilde{\mathbf{x}} = \begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{w} \end{bmatrix} \quad \bar{\mathbf{x}} = \tilde{\mathbf{x}} / \tilde{w}$$

Модель камеры

- Снова вернемся к модели камеры
- Каждая точка в 3d проецируется на плоскость наблюдения через апертуру
- Точки вдоль вектора неразличимы



Проективное преобразование

- АКА гомология
- Аффинное - любая 2×3 матрица
- Гомология - любая 3×3 матрица
- Умножение на скаляр ничего не меняет

$$\tilde{\mathbf{x}}' = \tilde{\mathbf{H}} \tilde{\mathbf{x}}$$

Проективное преобразование

- АКА гомология
- Аффинное - любая 2×3 матрица
- Гомология - любая 3×3 матрица
- Умножение на скаляр ничего не меняет
 - $3^* H \sim H$

$$\tilde{\mathbf{x}}' = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{w} \end{bmatrix} \quad \tilde{\mathbf{x}}' = \tilde{H} \tilde{\mathbf{x}}$$

Проективное преобразование

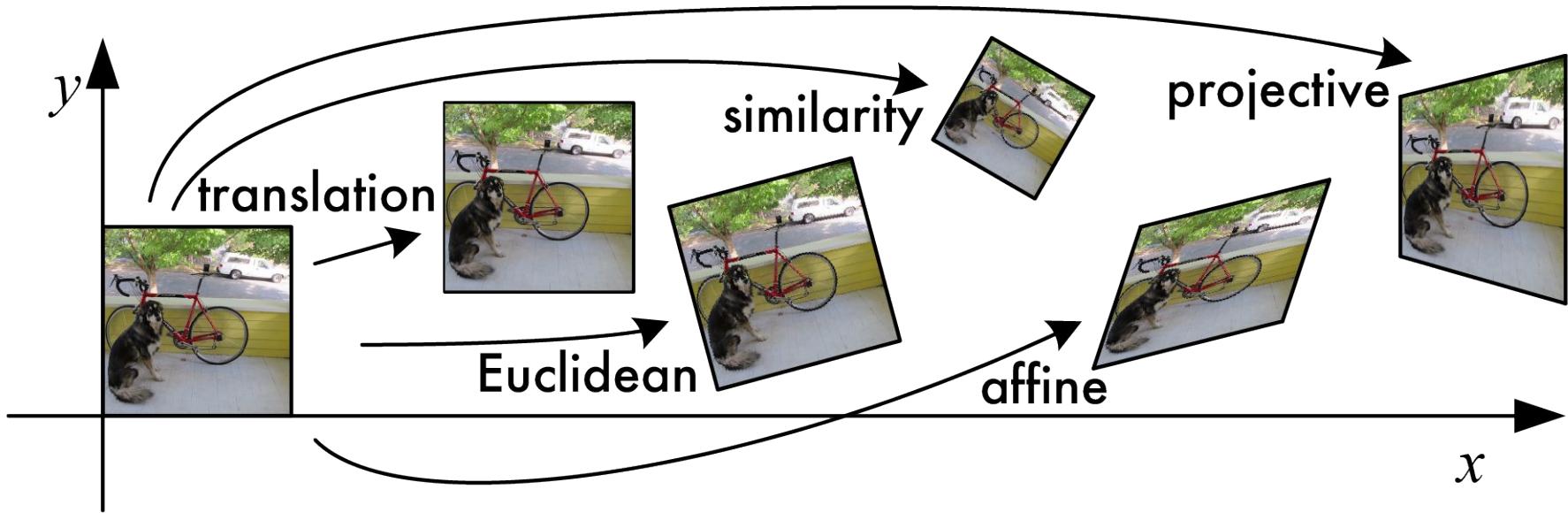
- Умножить $\tilde{\mathbf{x}}^{\sim}$ на \mathbf{H}^{\sim} , чтобы получить $\tilde{\mathbf{x}}'^{\sim}$
- Преобразуем в $\tilde{\mathbf{x}}'$ просто деля w'^{\sim}

$$\tilde{\mathbf{x}}' = \tilde{\mathbf{H}} \tilde{\mathbf{x}}$$

$$\begin{bmatrix} \tilde{x}' \\ \tilde{y}' \\ \tilde{w}' \end{bmatrix} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{w} \end{bmatrix} \quad \bar{\mathbf{x}} = \tilde{\mathbf{x}} / \tilde{w}$$

Какое выбрать?

- Что делает каждый из них?
- Какой лучше подходит для панорамы?



Какое выбрать?

$$\bar{\mathbf{x}}' = \begin{bmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

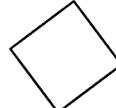
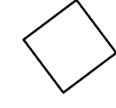
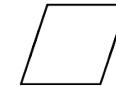
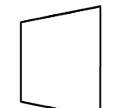
$$\bar{\mathbf{x}}' = \begin{bmatrix} \cos\theta & -\sin\theta & dx \\ \sin\theta & \cos\theta & dy \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\bar{\mathbf{x}}' = \begin{bmatrix} a & -b & dx \\ b & a & dy \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\bar{\mathbf{x}}' = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

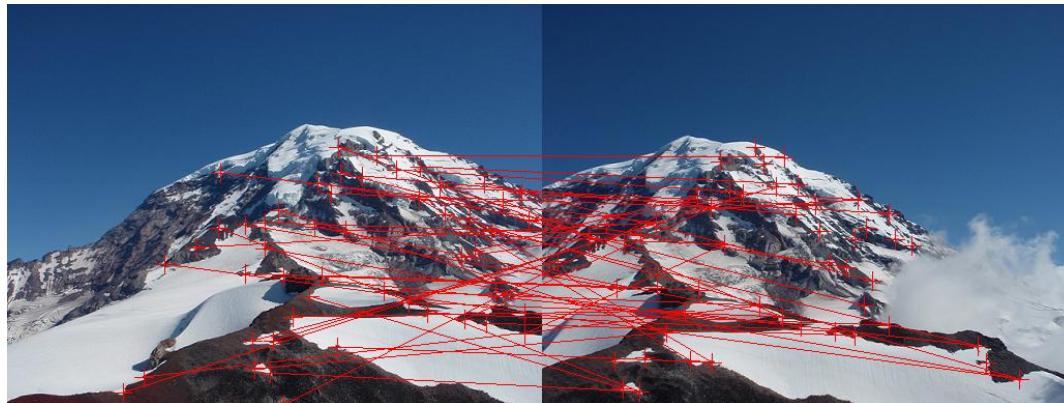
$$\tilde{\mathbf{x}}' = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{w} \end{bmatrix}$$

Какое выбрать?

Transformation	Matrix	# DoF	Preserves	Icon
translation	$\begin{bmatrix} \mathbf{I} & \mathbf{t} \end{bmatrix}_{2 \times 3}$	2	orientation	
rigid (Euclidean)	$\begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix}_{2 \times 3}$	3	lengths	
similarity	$\begin{bmatrix} s\mathbf{R} & \mathbf{t} \end{bmatrix}_{2 \times 3}$	4	angles	
affine	$\begin{bmatrix} \mathbf{A} \end{bmatrix}_{2 \times 3}$	6	parallelism	
projective	$\begin{bmatrix} \tilde{\mathbf{H}} \end{bmatrix}_{3 \times 3}$	8	straight lines	

Аффинное преобразование

- Есть уже сматченные точки
- Хотим оценить матрицу A , преобразующую x в x'
- $xA = x'$



Аффинное преобразование для матчинга

- Есть уже сматченные точки
- Хотим оценить матрицу A , преобразующую x в x'
- $xA = x'$
- Сколько степеней свободы?

Аффинное преобразование для матчинга

- Есть уже сматченные точки
- Хотим оценить матрицу A , преобразующую x в x'
- $xA = x'$
- Сколько степеней свободы?
 - 6
- Сколько уравнений задает одна пара точек? $mX = n$

$$x' = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Аффинное преобразование для матчинга

- Есть уже сматченные точки
- Хотим оценить матрицу A , преобразующую x в x'
- $xA = x'$
- Сколько степеней свободы?
 - 6
- Сколько уравнений задает одна пара точек? $mX = n$

- 2

- $n_x = a_{00} * m_x + a_{01} * m_y + a_{02} * 1$

- $n_y = a_{10} * m_x + a_{11} * m_y + a_{12} * 1$

$$x' = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Аффинное преобразование для матчинга

- Сколько уравнений задает одна пара точек? $m \mathbf{A} = \mathbf{n}$

- $n_x = a_{00} * m_x + a_{01} * m_y + a_{02} * 1$
- $n_y = a_{10} * m_x + a_{11} * m_y + a_{12} * 1$
- Решим уравнение $\mathbf{M} \mathbf{a} = \mathbf{b}$

- $\mathbf{M}^T \mathbf{M} \mathbf{a} = \mathbf{M}^T \mathbf{b}$
- $\mathbf{a} = (\mathbf{M}^T \mathbf{M})^{-1} \mathbf{M}^T \mathbf{b}$

- Работает даже если overdetermined
 - Почему???

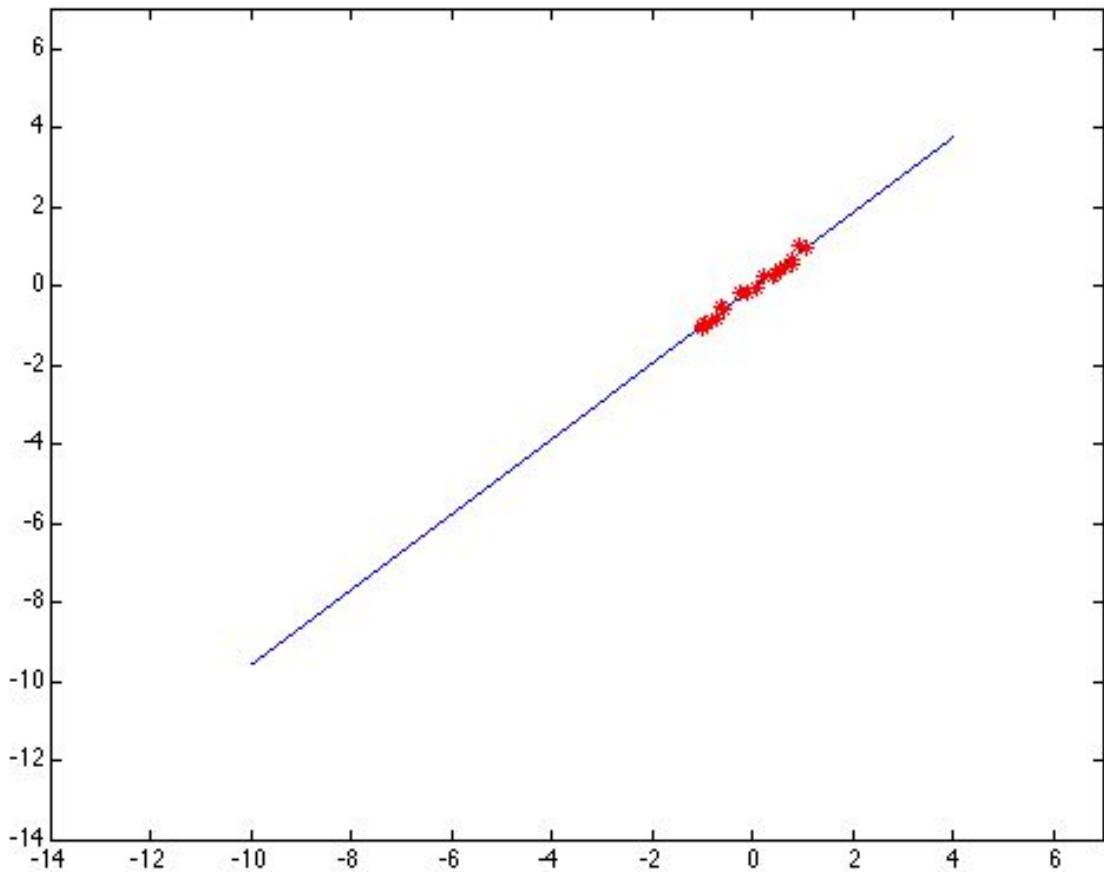
$$\begin{bmatrix} \mathbf{M} & \mathbf{a} & \mathbf{b} \end{bmatrix} = \begin{bmatrix} m_{x1} & m_{y1} & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & m_{x1} & m_{y1} & 1 \\ m_{x2} & m_{y2} & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & m_{x2} & m_{y2} & 1 \\ m_{x3} & m_{y3} & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & m_{x3} & m_{y3} & 1 \\ \dots & & & & & \end{bmatrix} \begin{bmatrix} a_{00} \\ a_{01} \\ a_{02} \\ a_{10} \\ a_{11} \\ a_{12} \end{bmatrix} = \begin{bmatrix} n_{x1} \\ n_{y1} \\ n_{x2} \\ n_{y2} \\ n_{x3} \\ n_{y3} \end{bmatrix}$$

Linear least squares

Хотим решить овердeterminированную линейную систему: $M a = b$

Минимизируем квадрат ошибок: $|| b - M a ||^2 =$

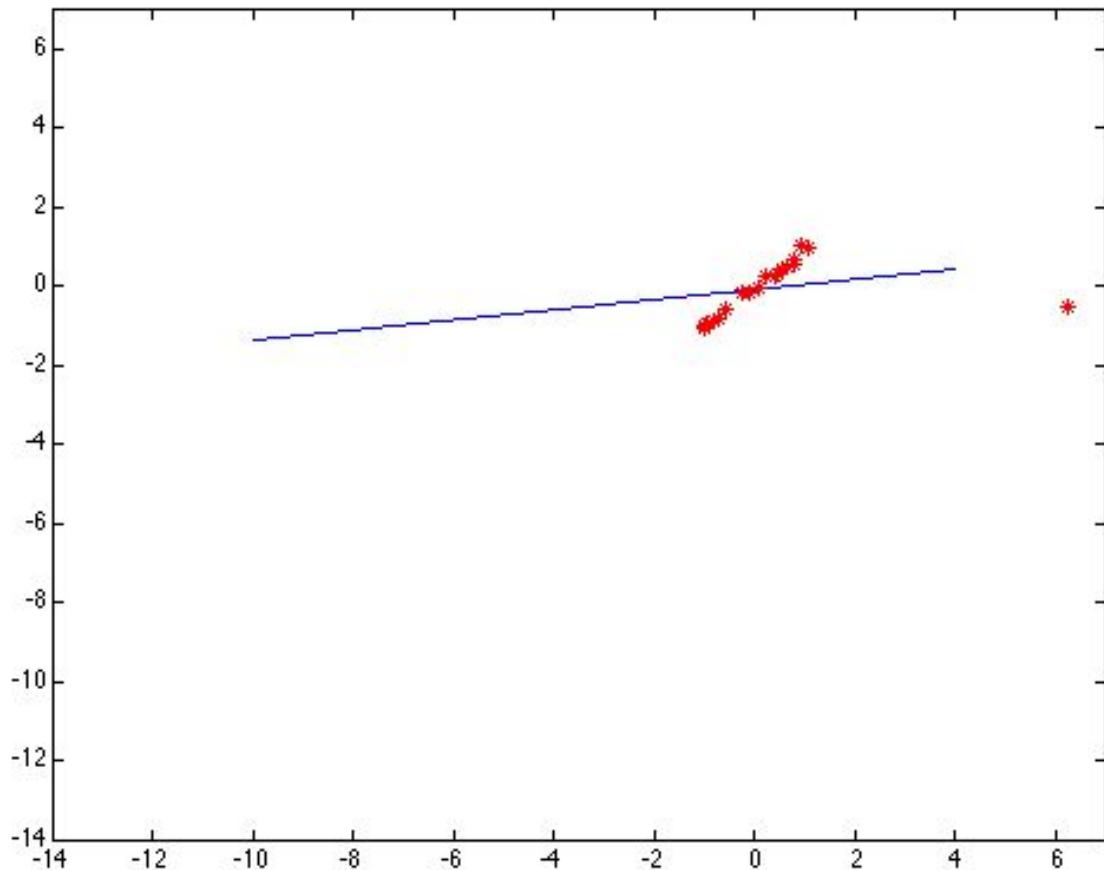
Геометрический смысл



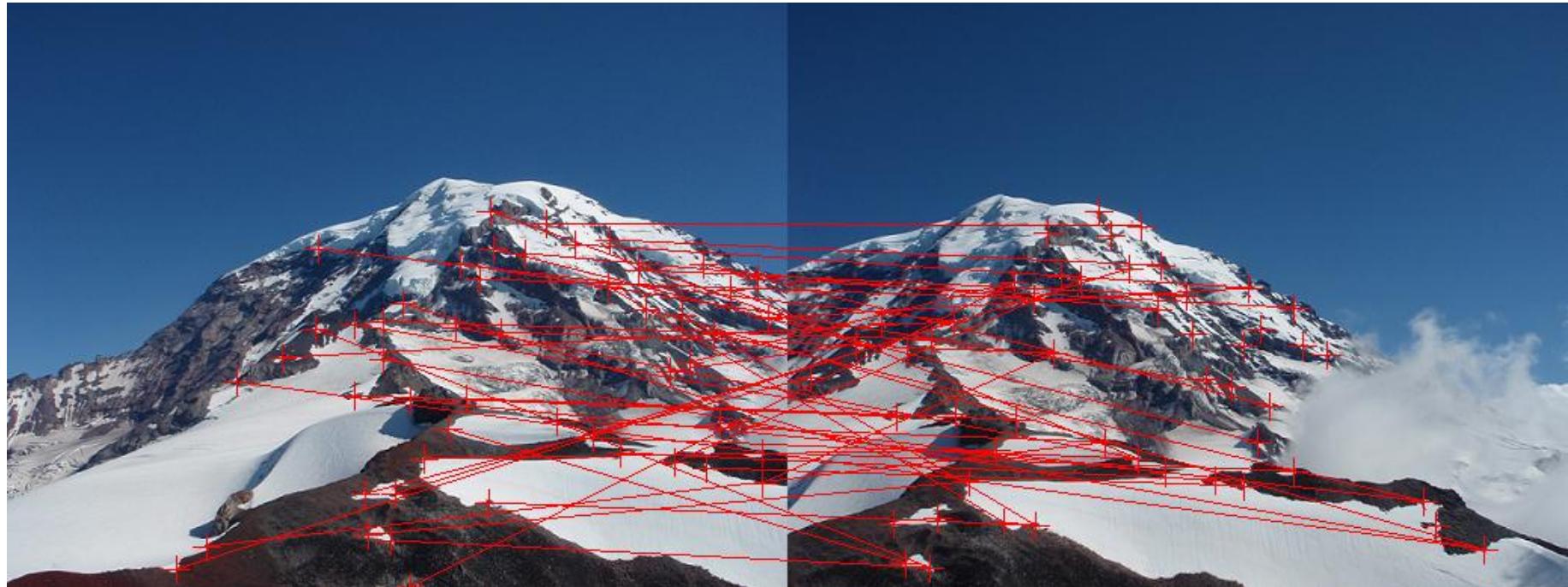
Геометрический смысл

Ошибка квадратичная!

Очень плохо справляется с
выбросами (outliers) в данных



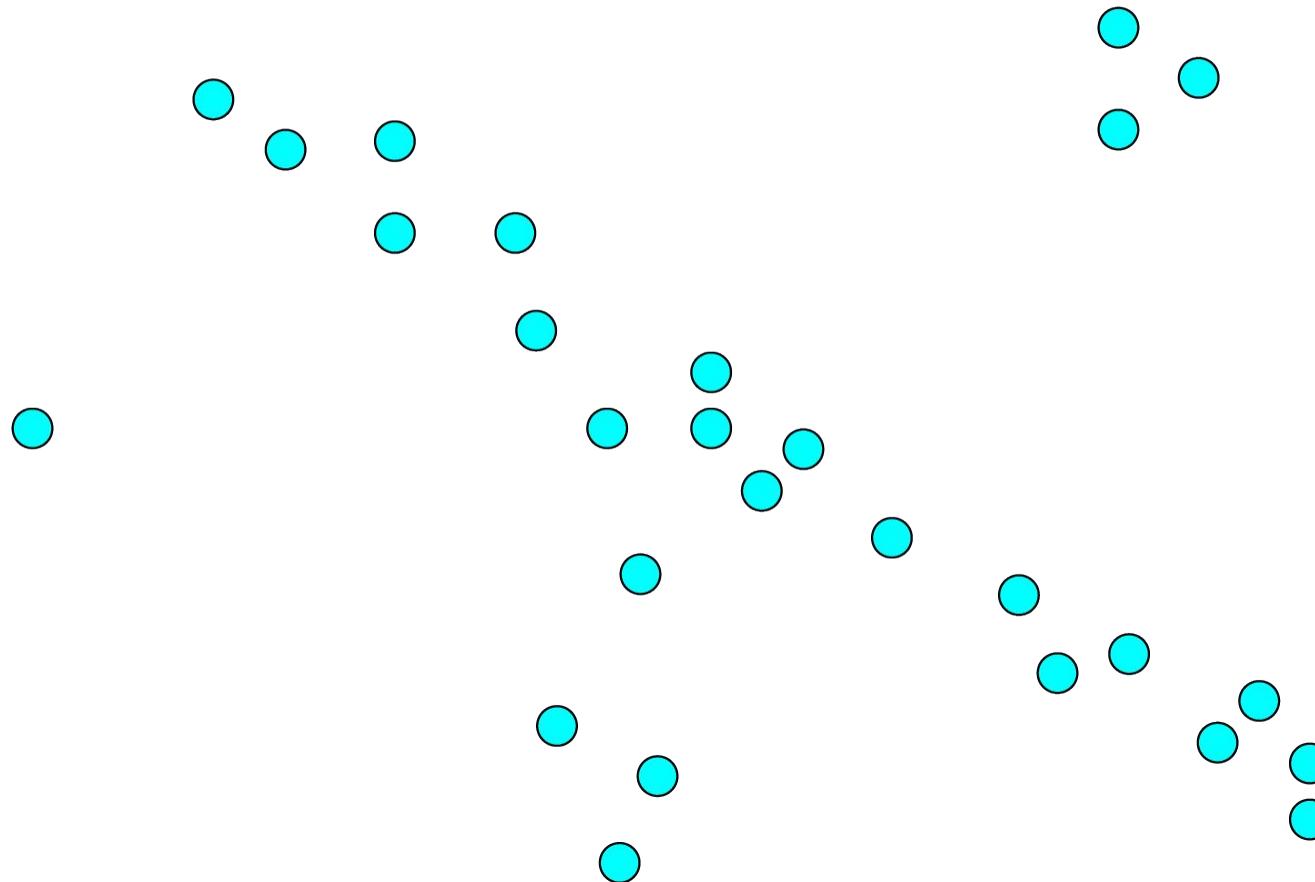
А у нас много выбросов...



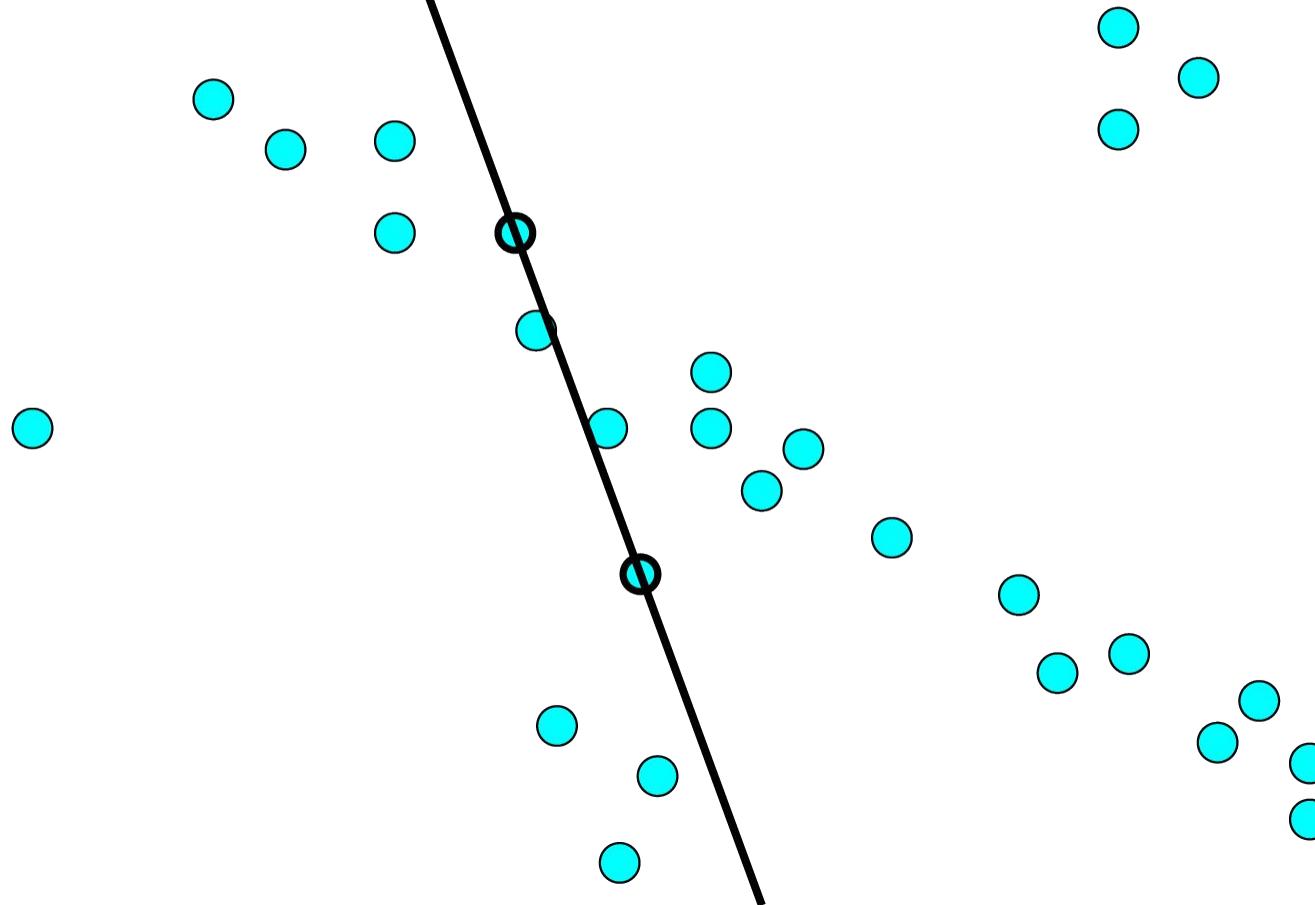
RANSAC: RANdom SAmple Consensus

- Как обучить модель по инлаерам, игнорируя аутлаеры?
- Обучить несколько моделей, посмотреть какая лучшая
- Инлаеры будут чаще согласовываться друг с другом
- Аутлаеры случайны, не будут согласовываться

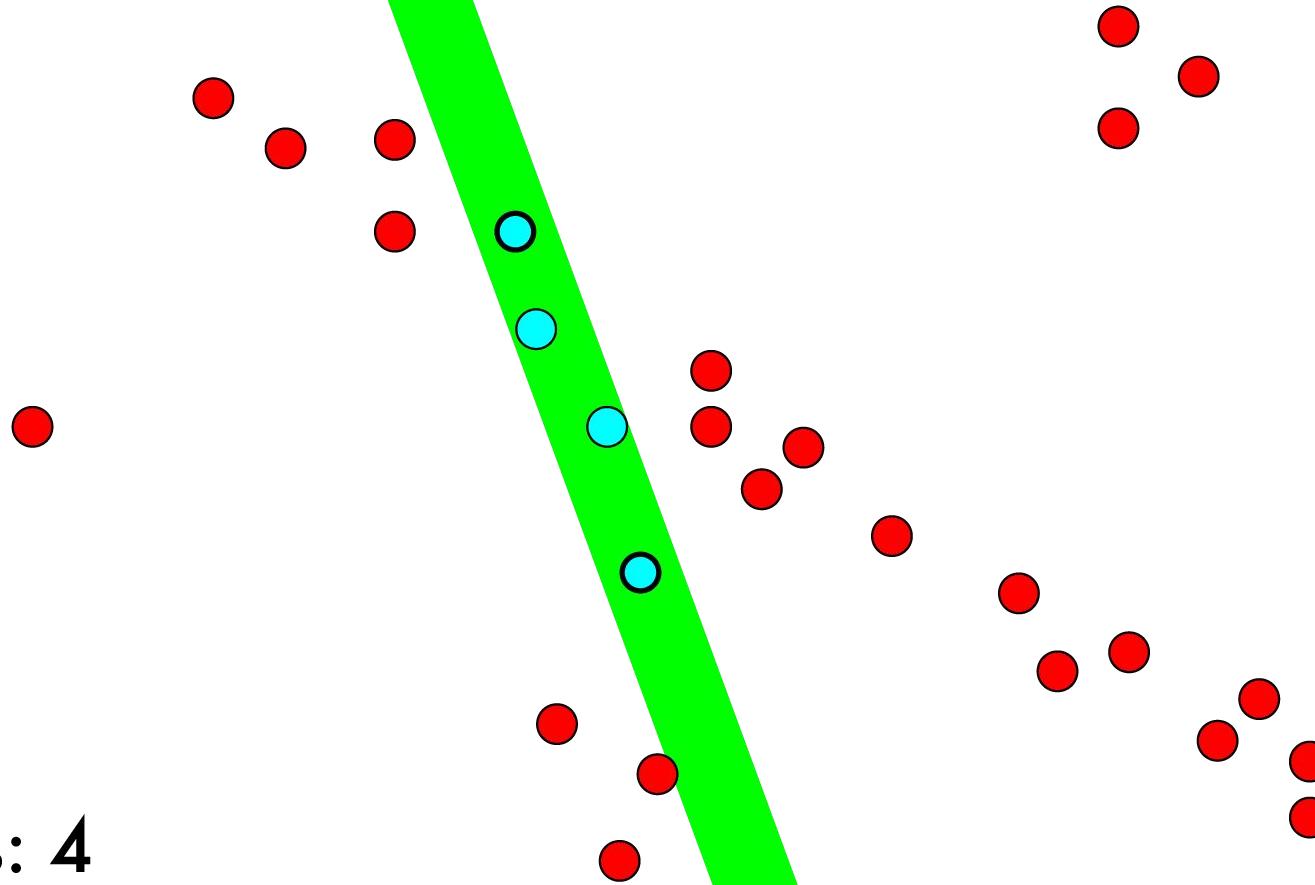
RANSAC: RANdom SAmple Consensus



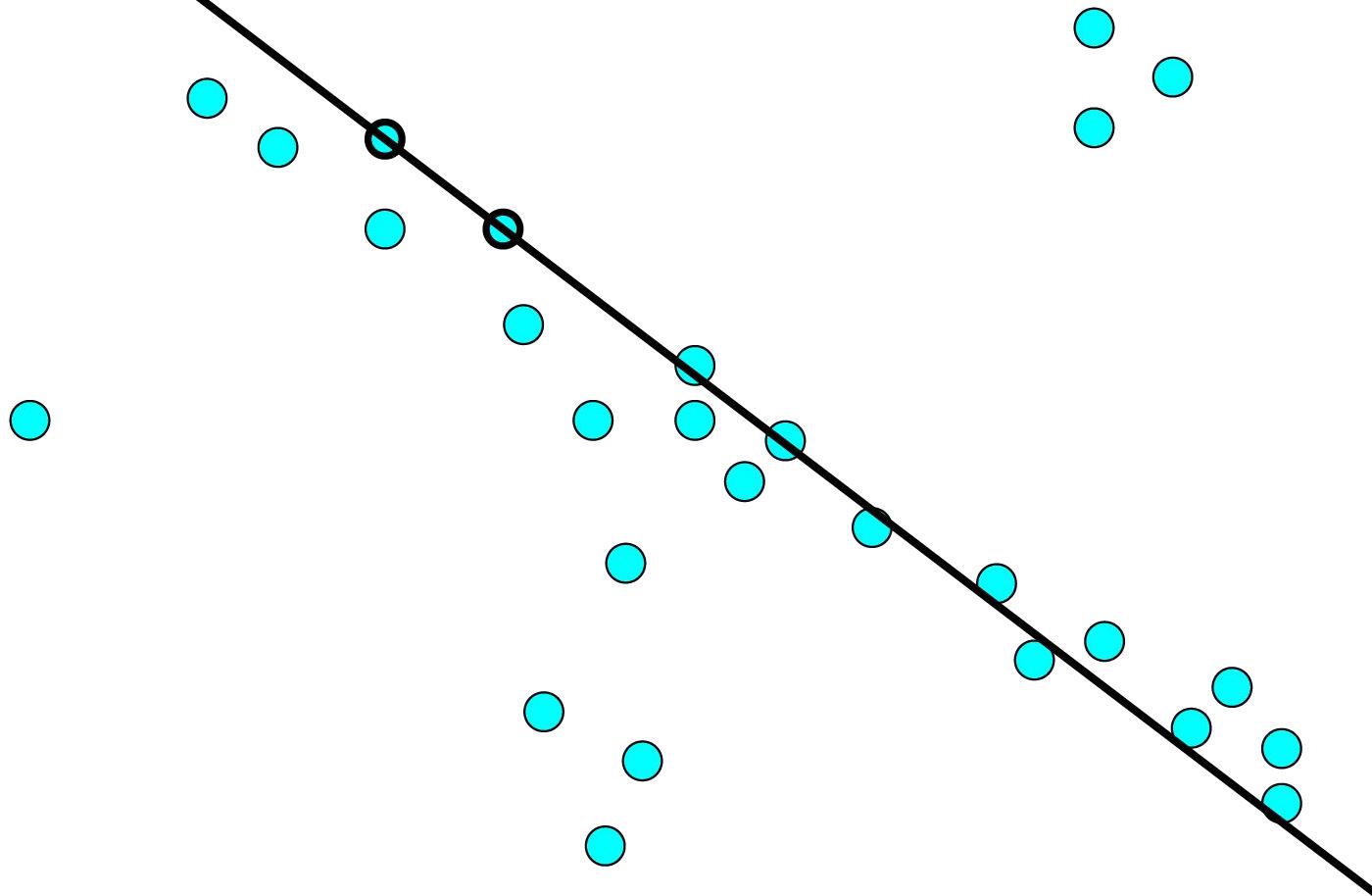
RANSAC: RANdom SAmple Consensus



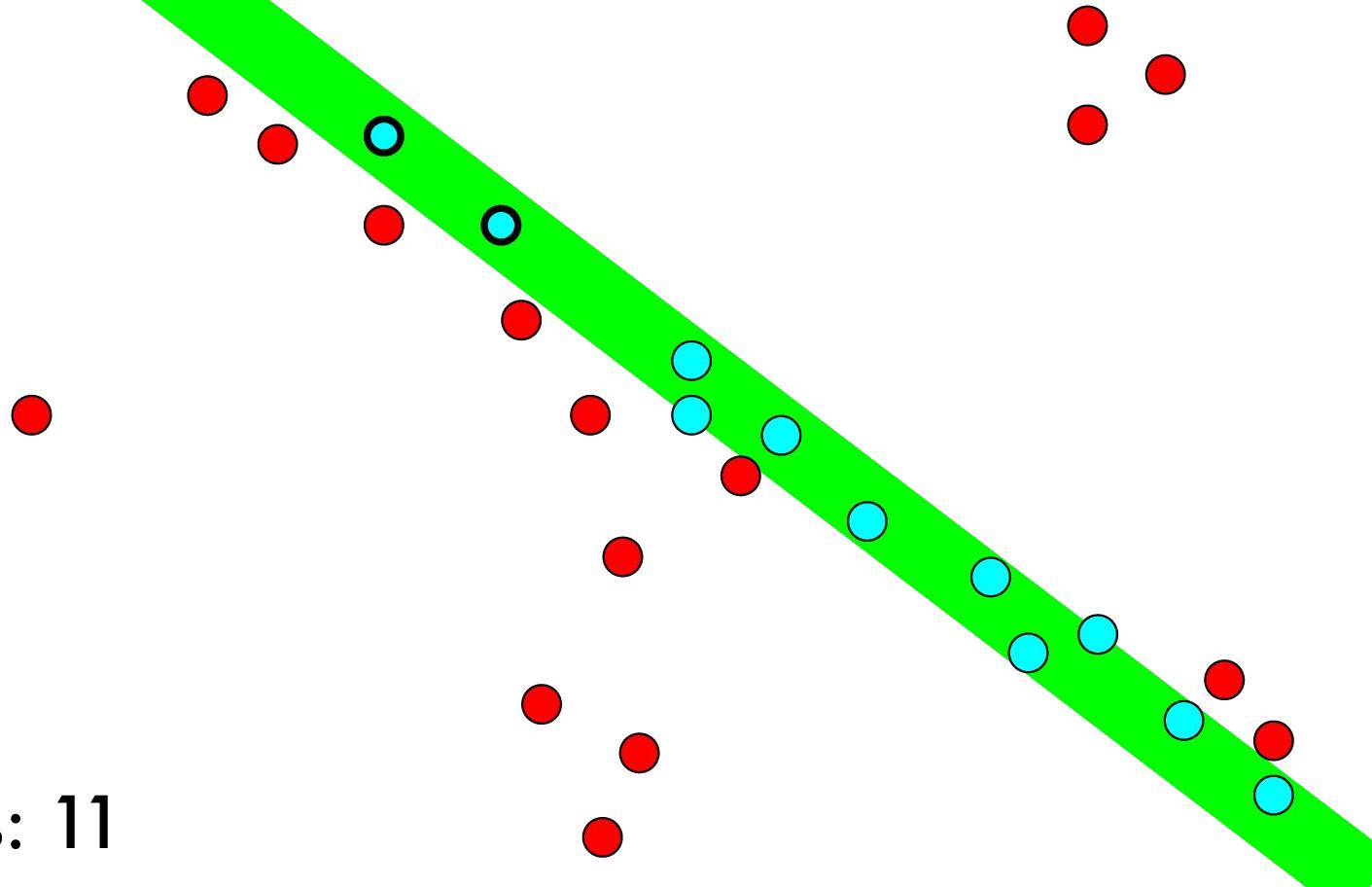
RANSAC: RANdom SAmple Consensus



RANSAC: RANdom SAmple Consensus

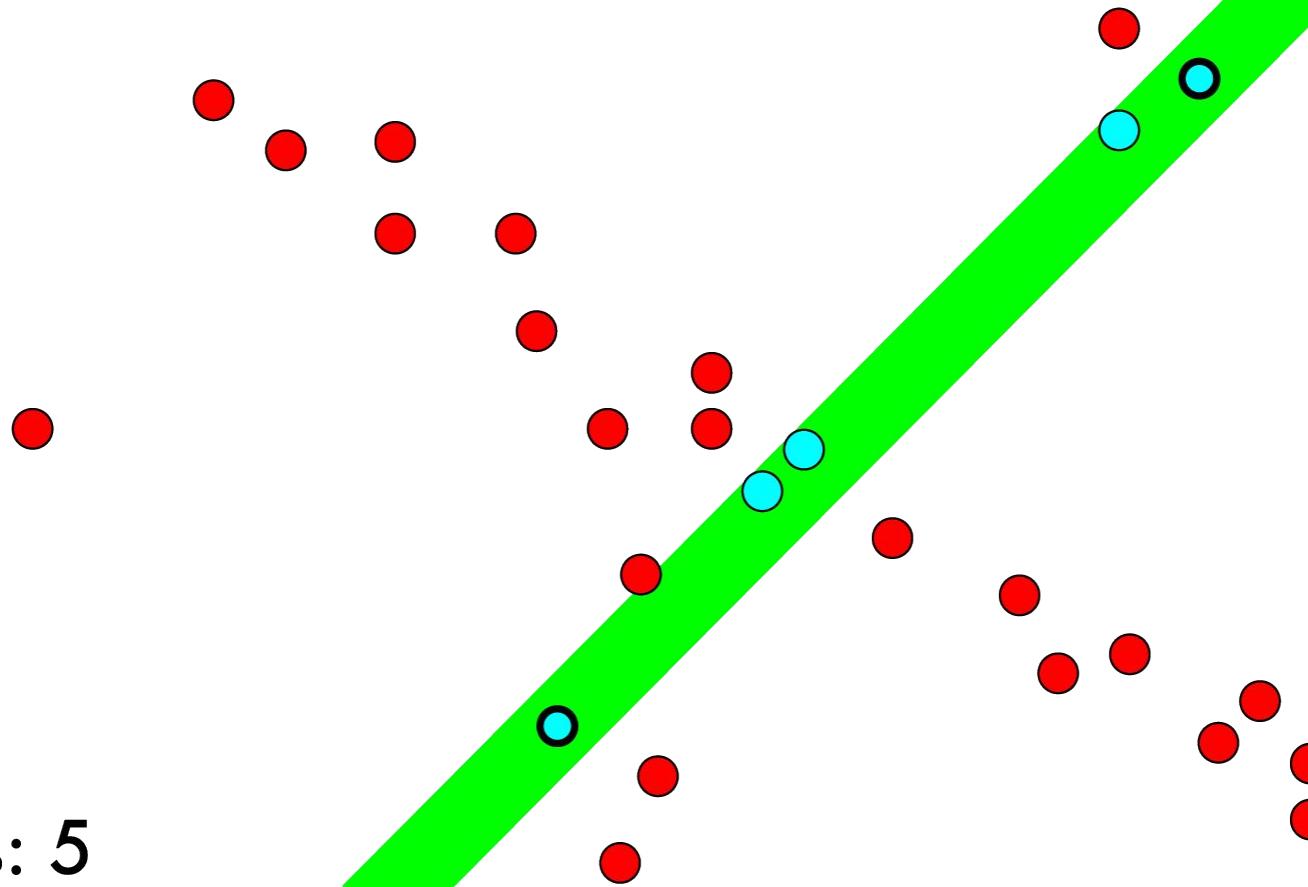


RANSAC: RANdom SAmple Consensus

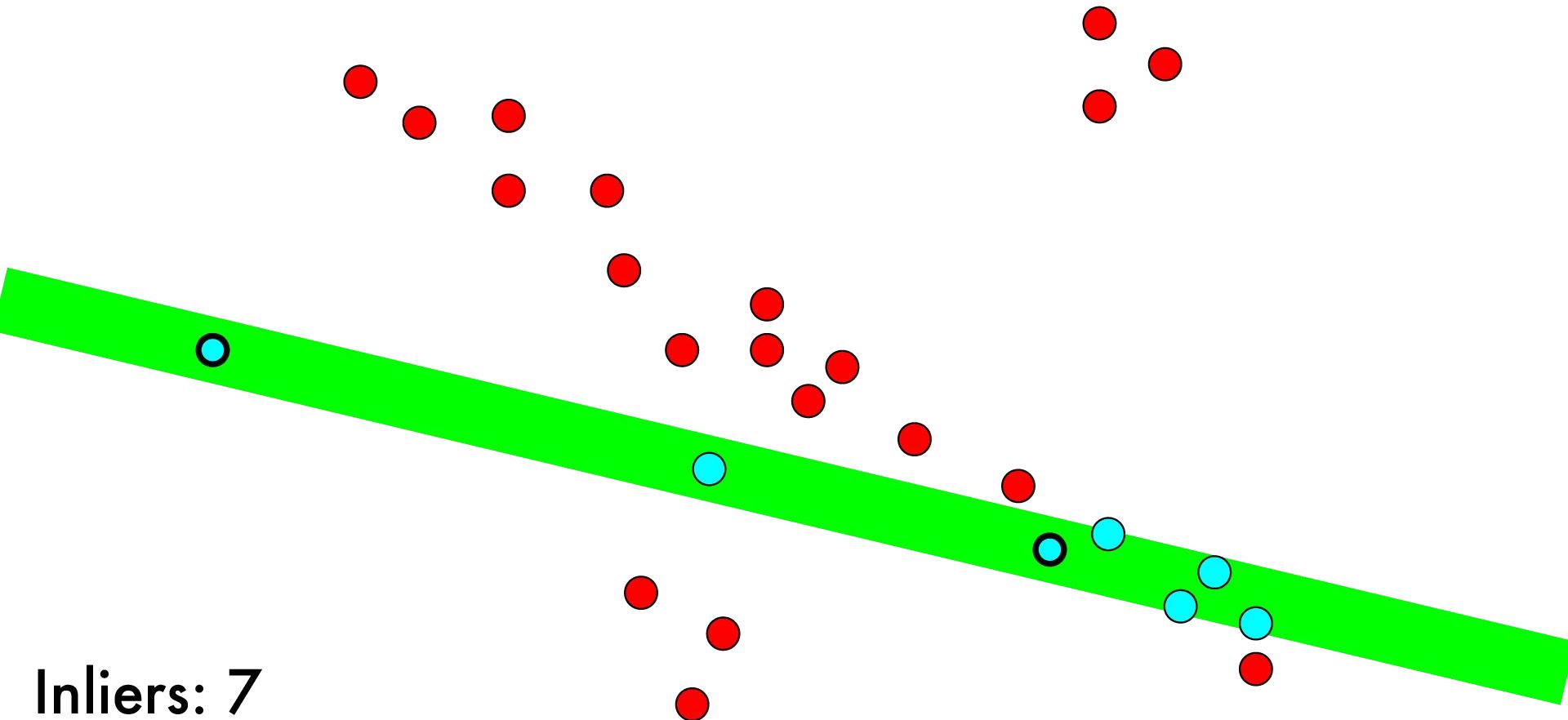


Inliers: 11

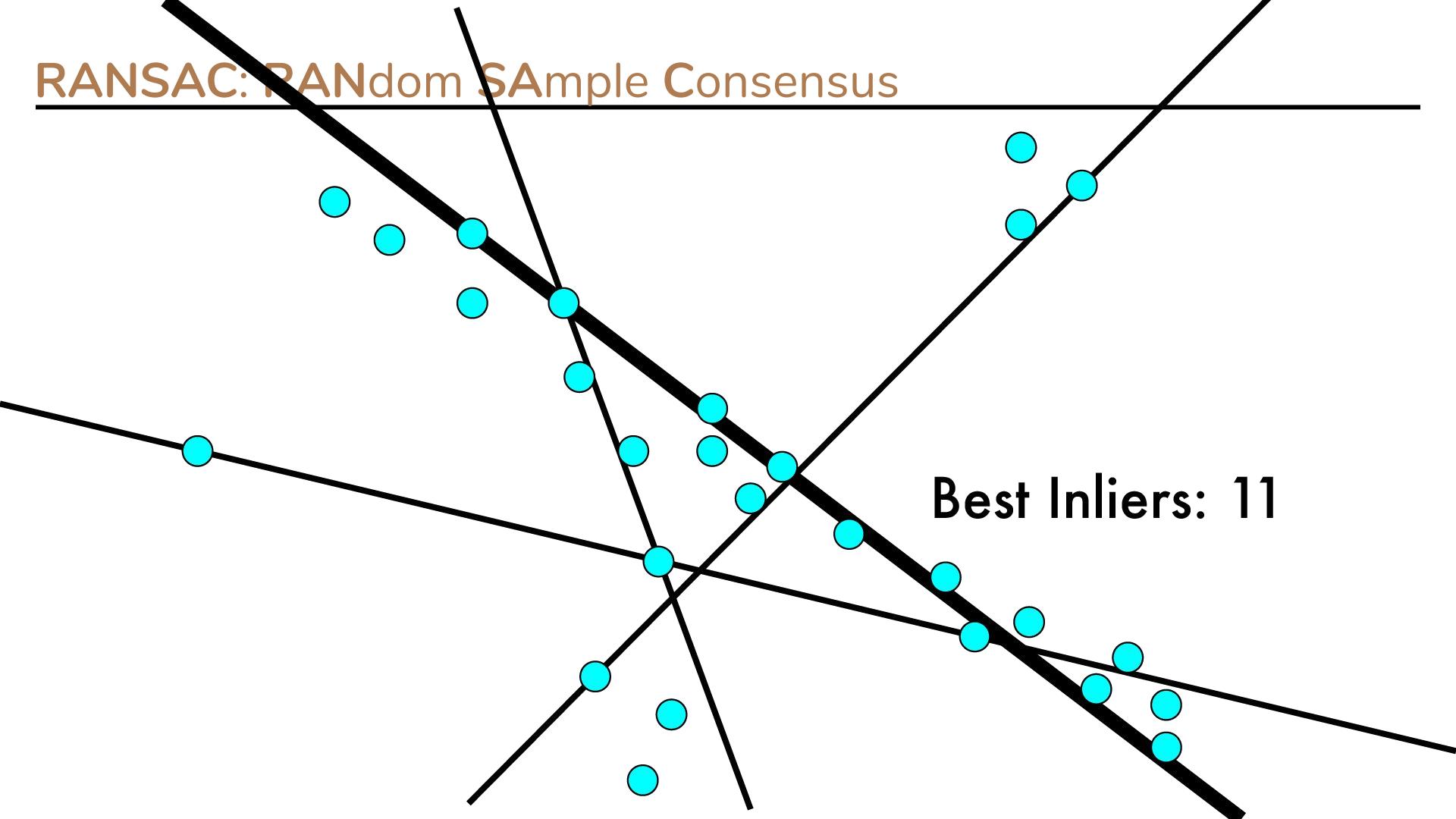
RANSAC: RANdom SAmple Consensus



RANSAC: RANdom SAmple Consensus



RANSAC: RANdom SAmple Consensus



RANSAC: RANdom SAmple Consensus

- Параметры: данные, модель, n - количество точек для обучения, k - количество итераций, t - порог, d - порог “хорошой модели”

```
bestmodel = None
```

```
bestfit = INF
```

```
While i < k:
```

```
    sample = draw n random points from data
```

```
    Fit model to sample
```

```
    inliers = data within t of model
```

```
    if number of inliers > bestfit:
```

```
        bestfit = number of inliers
```

```
        Fit model to all inliers
```

```
        bestmodel = model
```

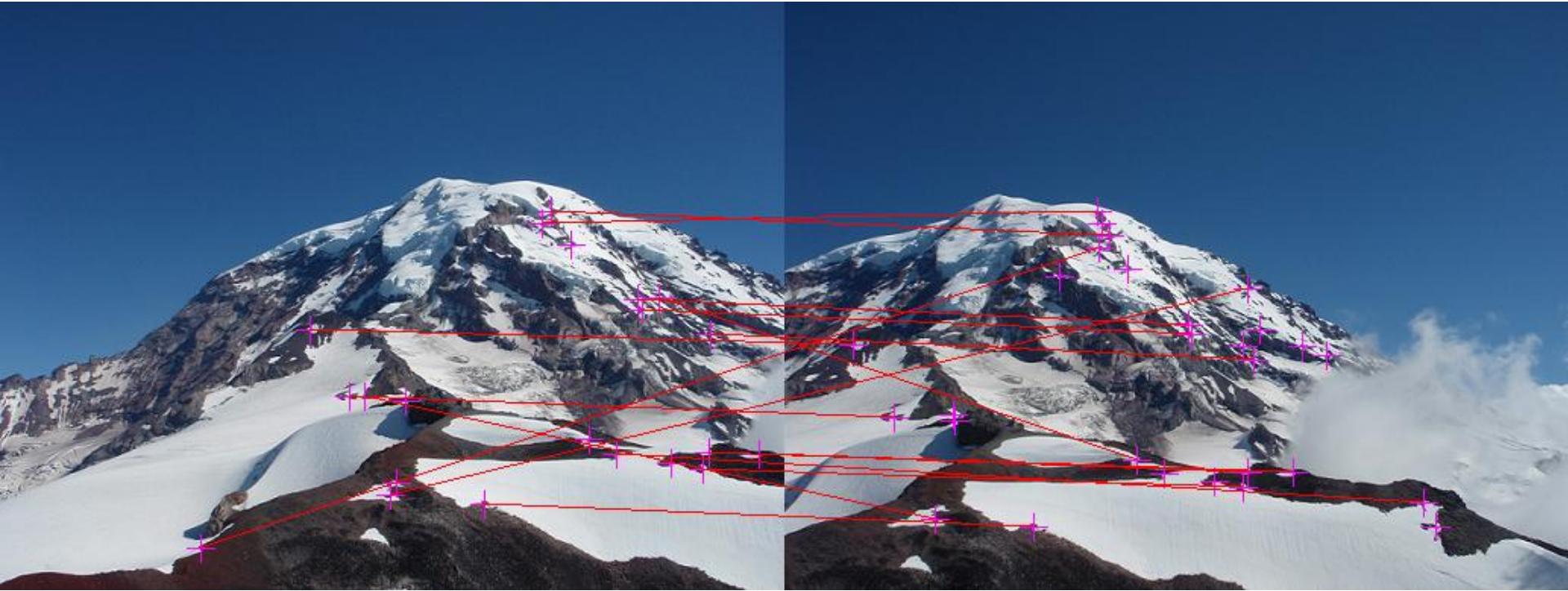
```
        if inliers > d:
```

```
            return bestmodel
```

```
return bestmodel
```

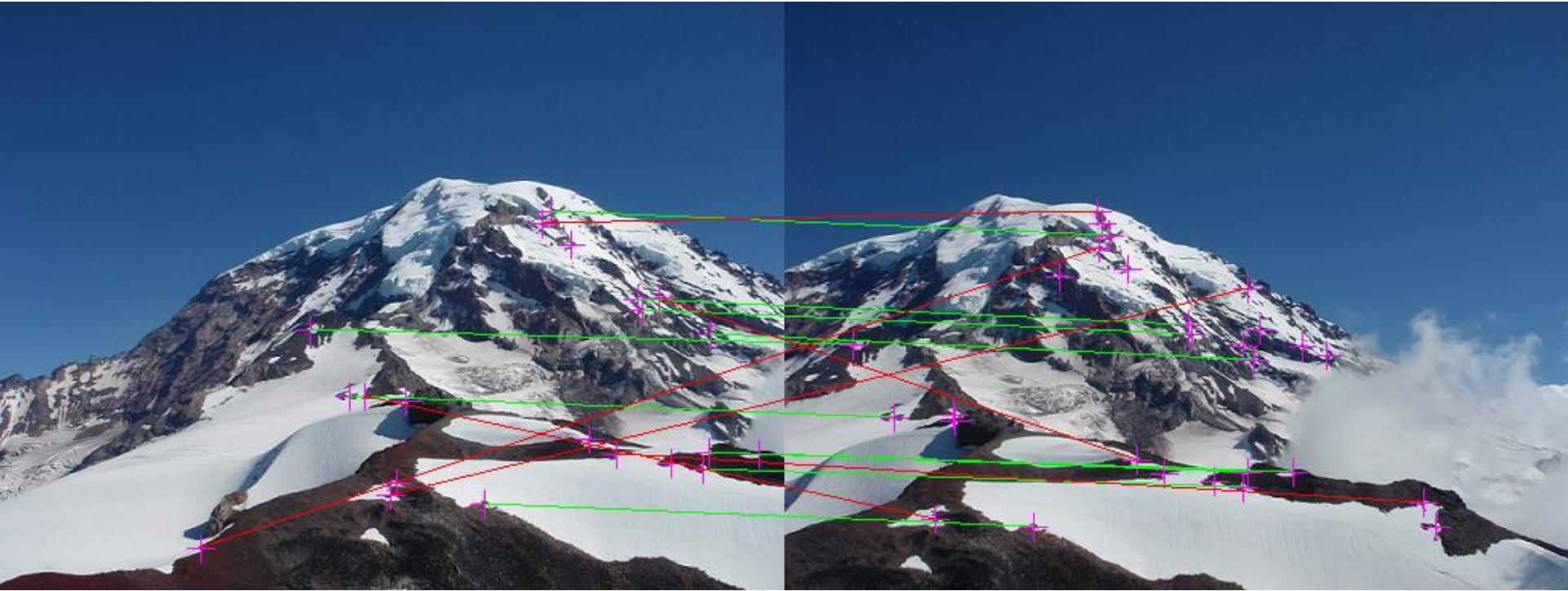
RANSAC: RANdom SAmple Consensus

- Хорошо работает с большим количеством шума



RANSAC: RANdom SAmple Consensus

- Хорошо работает с большим количеством шума



RANSAC: RANdom SAmple Consensus

- Параметры: данные, модель, n - количество точек для обучения, k - количество итераций, t - порог, d - порог “хорошой модели”
- Очень много гиперпараметров
- Хотим восстановить “правильную” модель
- t : довольно маленькая, ищем только “хорошие” инлаеры
- n : должно быть минимальным для обучения модели
- k : может быть очень большим
- d : должно быть сопоставимо с n

Аффинное преобразование

- Сколько уравнений задает одна пара точек? $m \mathbf{A} = \mathbf{n}$

- $n_x = a_{00} * m_x + a_{01} * m_y + a_{02} * 1$
- $n_y = a_{10} * m_x + a_{11} * m_y + a_{12} * 1$
- Решим уравнение $\mathbf{M} \mathbf{a} = \mathbf{b}$

- $\mathbf{M}^T \mathbf{M} \mathbf{a} = \mathbf{M}^T \mathbf{b}$
- $\mathbf{a} = (\mathbf{M}^T \mathbf{M})^{-1} \mathbf{M}^T \mathbf{b}$

- Работает даже если overdetermined
 - Почему???

$$\begin{bmatrix} \mathbf{M} & \mathbf{a} & \mathbf{b} \end{bmatrix} = \begin{bmatrix} m_{x1} & m_{y1} & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & m_{x1} & m_{y1} & 1 \\ m_{x2} & m_{y2} & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & m_{x2} & m_{y2} & 1 \\ m_{x3} & m_{y3} & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & m_{x3} & m_{y3} & 1 \\ \dots & & & & & \end{bmatrix} \begin{bmatrix} a_{00} \\ a_{01} \\ a_{02} \\ a_{10} \\ a_{11} \\ a_{12} \end{bmatrix} = \begin{bmatrix} n_{x1} \\ n_{y1} \\ n_{x2} \\ n_{y2} \\ n_{x3} \\ n_{y3} \end{bmatrix}$$

Проективное преобразование

- Как теперь выглядят уравнения?

- $n_x = (h_{00} * m_x + h_{01} * m_y + h_{02} * m_w) / (h_{20} * m_x + h_{21} * m_y + h_{22} * m_w)$
- $n_y = (h_{10} * m_x + h_{11} * m_y + h_{12} * m_w) / (h_{20} * m_x + h_{21} * m_y + h_{22} * m_w)$

$$\begin{bmatrix} \tilde{x}' \\ \tilde{y}' \\ \tilde{w}' \end{bmatrix} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{w} \end{bmatrix}$$

Проективное преобразование

- Как теперь выглядят уравнения?

- $n_x = (h_{00} * m_x + h_{01} * m_y + h_{02} * m_w) / (h_{20} * m_x + h_{21} * m_y + h_{22} * m_w)$
- $n_y = (h_{10} * m_x + h_{11} * m_y + h_{12} * m_w) / (h_{20} * m_x + h_{21} * m_y + h_{22} * m_w)$

- Предположим h_{22} и m_w равны 1, теперь 8 DOF

- $n_x = (h_{00} * m_x + h_{01} * m_y + h_{02}) / (h_{20} * m_x + h_{21} * m_y + 1)$
- $n_y = (h_{10} * m_x + h_{11} * m_y + h_{12}) / (h_{20} * m_x + h_{21} * m_y + 1)$

$$\begin{bmatrix} \tilde{x}' \\ \tilde{y}' \\ \tilde{w}' \end{bmatrix} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{w} \end{bmatrix}$$

Проективное преобразование

- Как теперь выглядят уравнения?
 - $n_x = (h_{00} * m_x + h_{01} * m_y + h_{02} * m_w) / (h_{20} * m_x + h_{21} * m_y + h_{22} * m_w)$
 - $n_y = (h_{10} * m_x + h_{11} * m_y + h_{12} * m_w) / (h_{20} * m_x + h_{21} * m_y + h_{22} * m_w)$
- Предположим h_{22} и m_w равны 1, теперь 8 DOF
 - $n_x = (h_{00} * m_x + h_{01} * m_y + h_{02}) / (h_{20} * m_x + h_{21} * m_y + 1)$
 - $n_y = (h_{10} * m_x + h_{11} * m_y + h_{12}) / (h_{20} * m_x + h_{21} * m_y + 1)$
- Покрутим еще n_x
 - $n_x * (h_{20} * m_x + h_{21} * m_y + 1) = (h_{00} * m_x + h_{01} * m_y + h_{02})$
 - $n_x * h_{20} * m_x + n_x * h_{21} * m_y + n_x = h_{00} * m_x + h_{01} * m_y + h_{02}$
 - $n_x = h_{00} * m_x + h_{01} * m_y + h_{02} - n_x * h_{20} * m_x - n_x * h_{21} * m_y$
- Аналогично n_y

Проективное преобразование

- Как теперь выглядят уравнения?

- $n_x = h_{00} * m_x + h_{01} * m_y + h_{02} - n_x * h_{20} * m_x - n_x * h_{21} * m_y$
- $n_y = h_{10} * m_x + h_{11} * m_y + h_{12} - n_x * h_{20} * m_x - n_x * h_{21} * m_y$

- В матричных терминах:

$$\mathbf{M} \begin{bmatrix} m_{x1} & m_{y1} & 1 & 0 & 0 & 0 & -m_{x1}n_{x1} & -m_{y1}n_{x1} \\ 0 & 0 & 0 & m_{x1} & m_{y1} & 1 & -m_{x1}n_{y1} & -m_{y1}n_{y1} \\ m_{x2} & m_{y2} & 1 & 0 & 0 & 0 & -m_{x2}n_{x2} & -m_{y2}n_{x2} \\ 0 & 0 & 0 & m_{x2} & m_{y2} & 1 & -m_{x2}n_{y2} & -m_{y2}n_{y2} \\ m_{x3} & m_{y3} & 1 & 0 & 0 & 0 & -m_{x3}n_{x3} & -m_{y3}n_{x3} \\ 0 & 0 & 0 & m_{x3} & m_{y3} & 1 & -m_{x3}n_{y3} & -m_{y3}n_{y3} \\ m_{x4} & m_{y4} & 1 & 0 & 0 & 0 & -m_{x4}n_{x4} & -m_{y4}n_{x4} \\ 0 & 0 & 0 & m_{x4} & m_{y4} & 1 & -m_{x4}n_{y4} & -m_{y4}n_{y4} \end{bmatrix} \begin{matrix} \mathbf{a} \\ \begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \end{bmatrix} \end{matrix} = \begin{matrix} \mathbf{b} \\ \begin{bmatrix} n_{x1} \\ n_{y1} \\ n_{x2} \\ n_{y2} \\ n_{x3} \\ n_{y3} \\ n_{x4} \\ n_{y4} \end{bmatrix} \end{matrix}$$

Проективное преобразование

- Новое матричное уравнение

$$\mathbf{M} \begin{bmatrix} m_{x1} & m_{y1} & 1 & 0 & 0 & 0 & -m_{x1}n_{x1} & -m_{y1}n_{x1} \\ 0 & 0 & 0 & m_{x1} & m_{y1} & 1 & -m_{x1}n_{y1} & -m_{y1}n_{y1} \\ m_{x2} & m_{y2} & 1 & 0 & 0 & 0 & -m_{x2}n_{x2} & -m_{y2}n_{x2} \\ 0 & 0 & 0 & m_{x2} & m_{y2} & 1 & -m_{x2}n_{y2} & -m_{y2}n_{y2} \\ m_{x3} & m_{y3} & 1 & 0 & 0 & 0 & -m_{x3}n_{x3} & -m_{y3}n_{x3} \\ 0 & 0 & 0 & m_{x3} & m_{y3} & 1 & -m_{x3}n_{y3} & -m_{y3}n_{y3} \\ m_{x4} & m_{y4} & 1 & 0 & 0 & 0 & -m_{x4}n_{x4} & -m_{y4}n_{x4} \\ 0 & 0 & 0 & m_{x4} & m_{y4} & 1 & -m_{x4}n_{y4} & -m_{y4}n_{y4} \end{bmatrix} \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix} = \begin{bmatrix} n_{x1} \\ n_{y1} \\ n_{x2} \\ n_{y2} \\ n_{x3} \\ n_{y3} \\ n_{x4} \\ n_{y4} \end{bmatrix}$$

- Тот же самый алгоритм: решаем $\mathbf{M} \mathbf{a} = \mathbf{b}$
 - Точное решение если количество строк в $\mathbf{M} = 8$
 - МНК если количество строк в $\mathbf{M} > 8$

Проблема...

$$\mathbf{M} \begin{bmatrix} m_{x1} & m_{y1} & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & m_{x1} & m_{y1} & 1 \\ m_{x2} & m_{y2} & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & m_{x2} & m_{y2} & 1 \\ m_{x3} & m_{y3} & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & m_{x3} & m_{y3} & 1 \\ m_{x4} & m_{y4} & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & m_{x4} & m_{y4} & 1 \end{bmatrix} \begin{bmatrix} -m_{x1}n_{x1} & -m_{y1}n_{x1} \\ -m_{x1}n_{y1} & -m_{y1}n_{y1} \\ -m_{x2}n_{x2} & -m_{y2}n_{x2} \\ -m_{x2}n_{y2} & -m_{y2}n_{y2} \\ -m_{x3}n_{x3} & -m_{y3}n_{x3} \\ -m_{x3}n_{y3} & -m_{y3}n_{y3} \\ -m_{x4}n_{x4} & -m_{y4}n_{x4} \\ -m_{x4}n_{y4} & -m_{y4}n_{y4} \end{bmatrix} = \begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \end{bmatrix} = \begin{bmatrix} n_{x1} \\ n_{y1} \\ n_{x2} \\ n_{y2} \\ n_{x3} \\ n_{y3} \\ n_{x4} \\ n_{y4} \end{bmatrix}$$

Проблема...

- h_{22} может равняться нулю

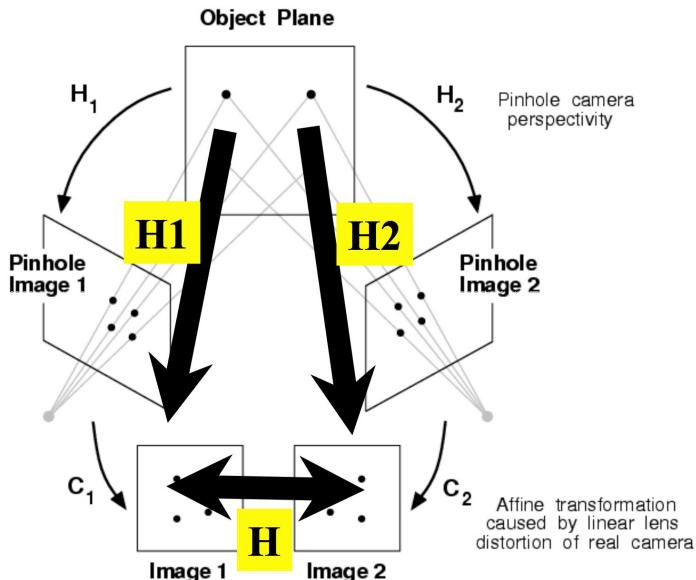
$$\mathbf{M} \begin{bmatrix} m_{x1} & m_{y1} & 1 & 0 & 0 & 0 & -m_{x1}n_{x1} & -m_{y1}n_{x1} \\ 0 & 0 & 0 & m_{x1} & m_{y1} & 1 & -m_{x1}n_{y1} & -m_{y1}n_{y1} \\ m_{x2} & m_{y2} & 1 & 0 & 0 & 0 & -m_{x2}n_{x2} & -m_{y2}n_{x2} \\ 0 & 0 & 0 & m_{x2} & m_{y2} & 1 & -m_{x2}n_{y2} & -m_{y2}n_{y2} \\ m_{x3} & m_{y3} & 1 & 0 & 0 & 0 & -m_{x3}n_{x3} & -m_{y3}n_{x3} \\ 0 & 0 & 0 & m_{x3} & m_{y3} & 1 & -m_{x3}n_{y3} & -m_{y3}n_{y3} \\ m_{x4} & m_{y4} & 1 & 0 & 0 & 0 & -m_{x4}n_{x4} & -m_{y4}n_{x4} \\ 0 & 0 & 0 & m_{x4} & m_{y4} & 1 & -m_{x4}n_{y4} & -m_{y4}n_{y4} \end{bmatrix} \begin{matrix} \mathbf{a} \\ \mathbf{h}_{00} \\ \mathbf{h}_{01} \\ \mathbf{h}_{02} \\ \mathbf{h}_{10} \\ \mathbf{h}_{11} \\ \mathbf{h}_{12} \\ \mathbf{h}_{20} \\ \mathbf{h}_{21} \end{matrix} = \begin{matrix} \mathbf{b} \\ n_{x1} \\ n_{y1} \\ n_{x2} \\ n_{y2} \\ n_{x3} \\ n_{y3} \\ n_{x4} \\ n_{y4} \end{matrix}$$

Алгоритм построения панорамы:

- Найти углы на обоих изображениях
- Посчитать дескрипторы
- Сматчить дескрипторы
- С помощью RANSAC найти параметры преобразования
- Склейте изображения с помощью этого преобразования

Склейивание панорам

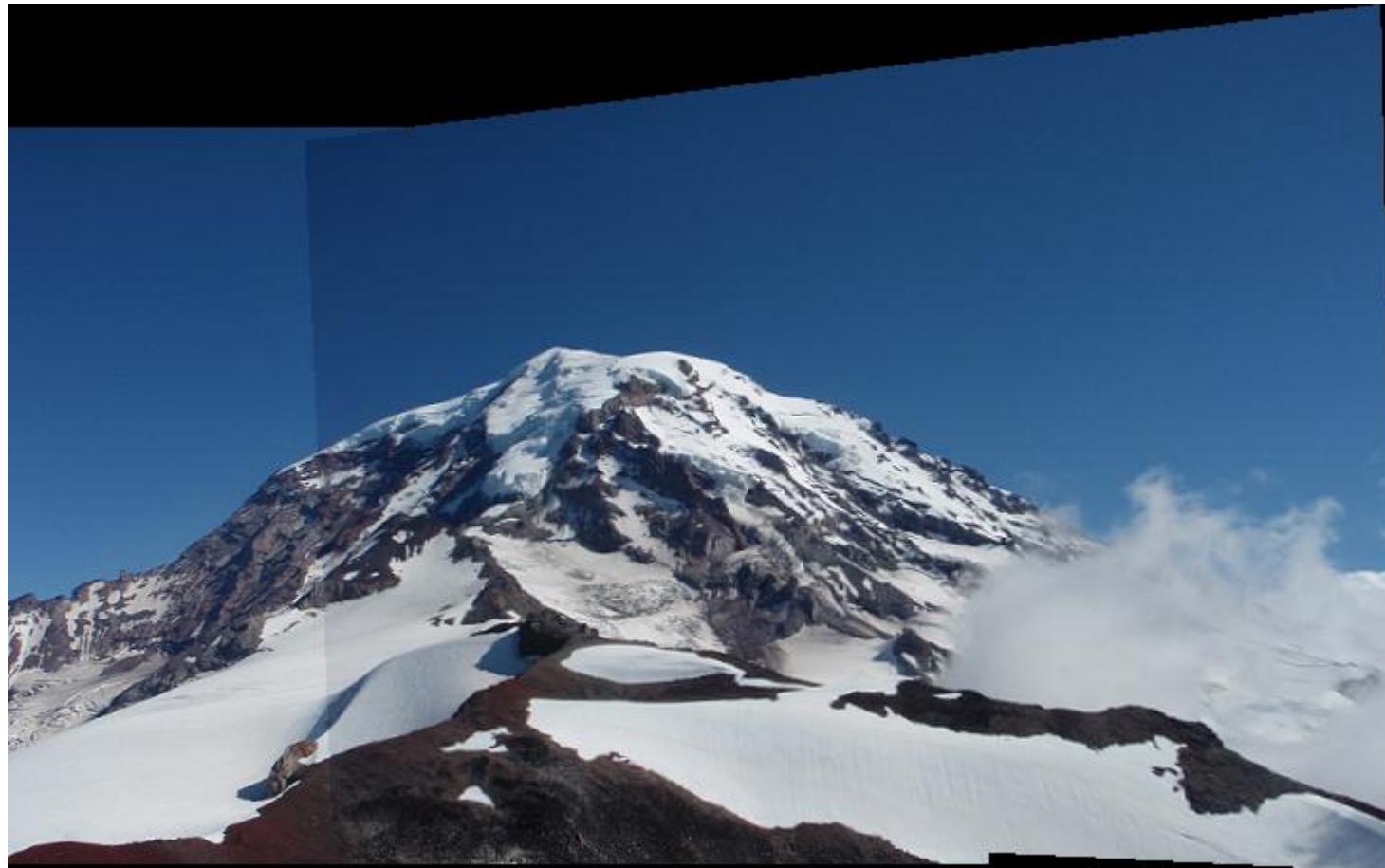
- Проективное преобразование - хороший выбор при соблюдении некоторых условий
- Предполагаем, что мы делаем фотографии некоторого плоского объекта



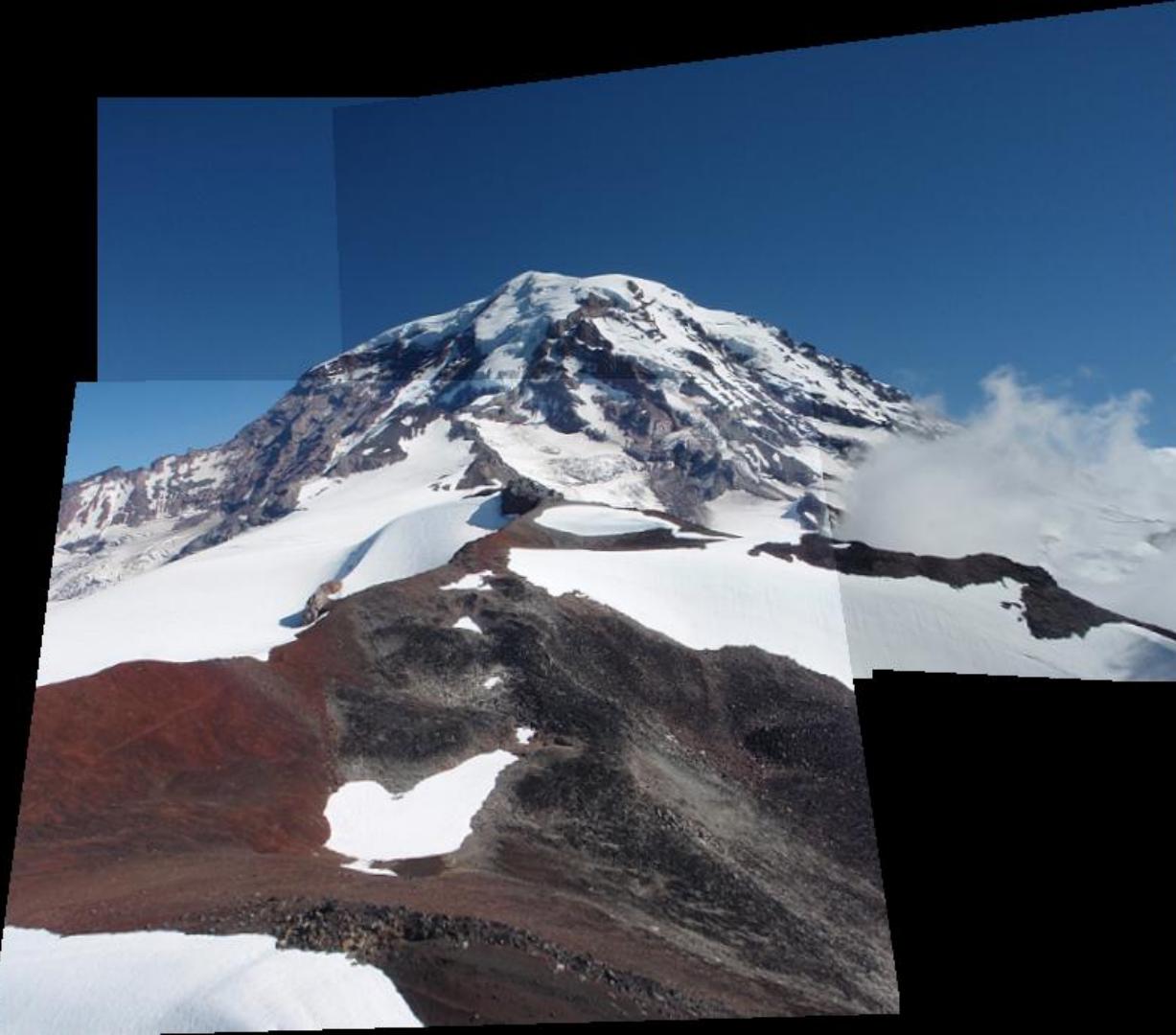
На практике



На практике

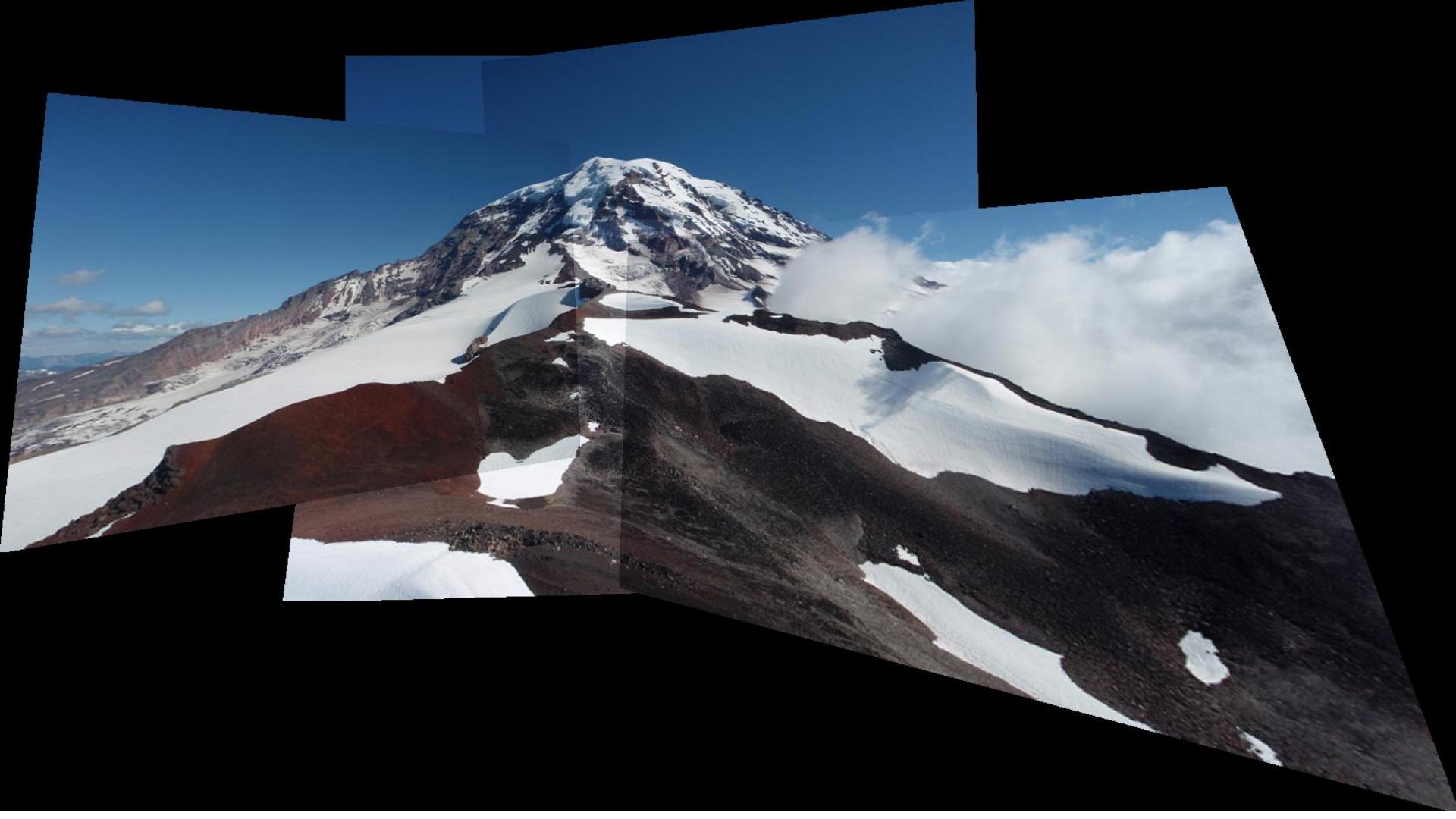


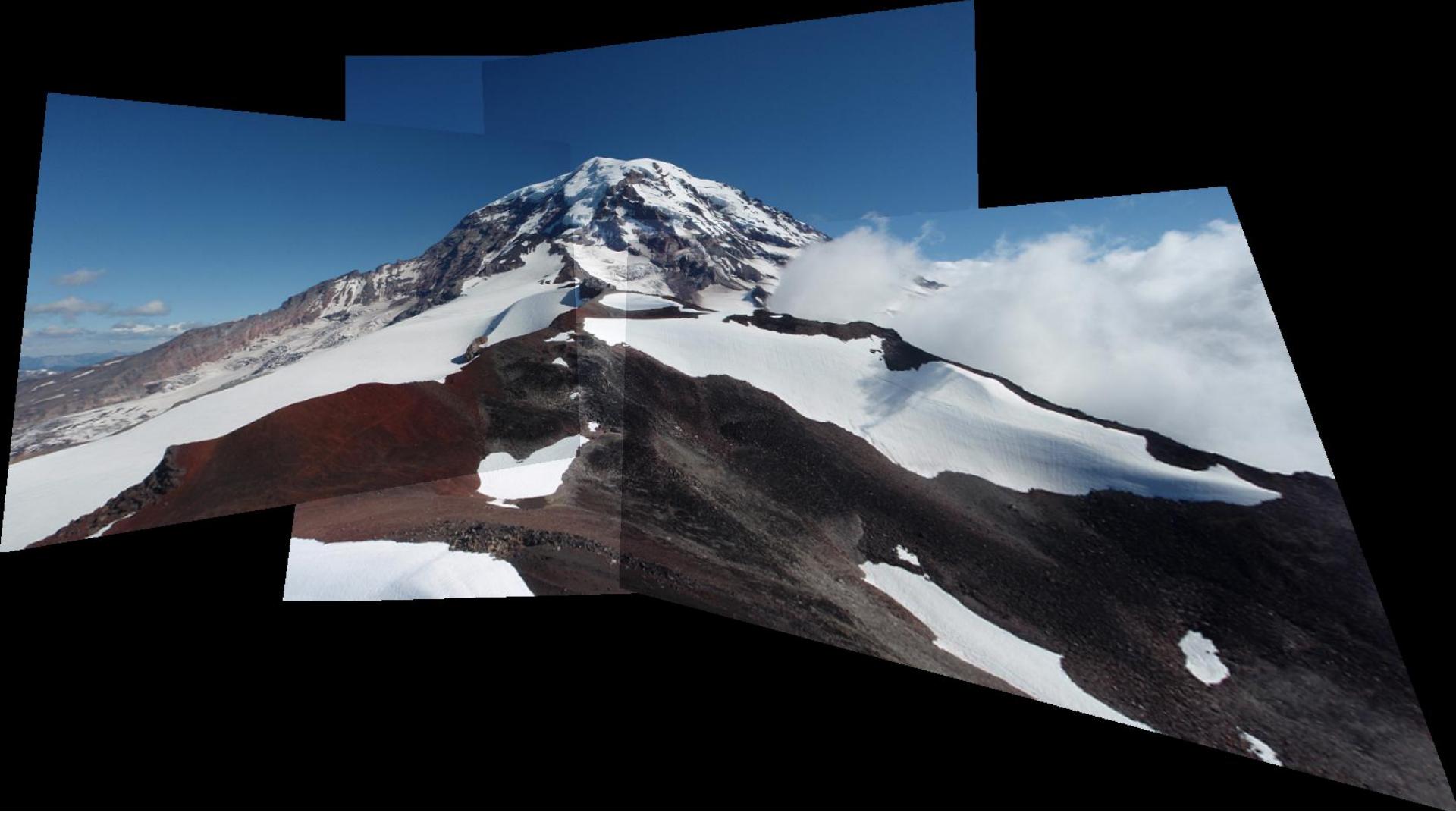
На практике



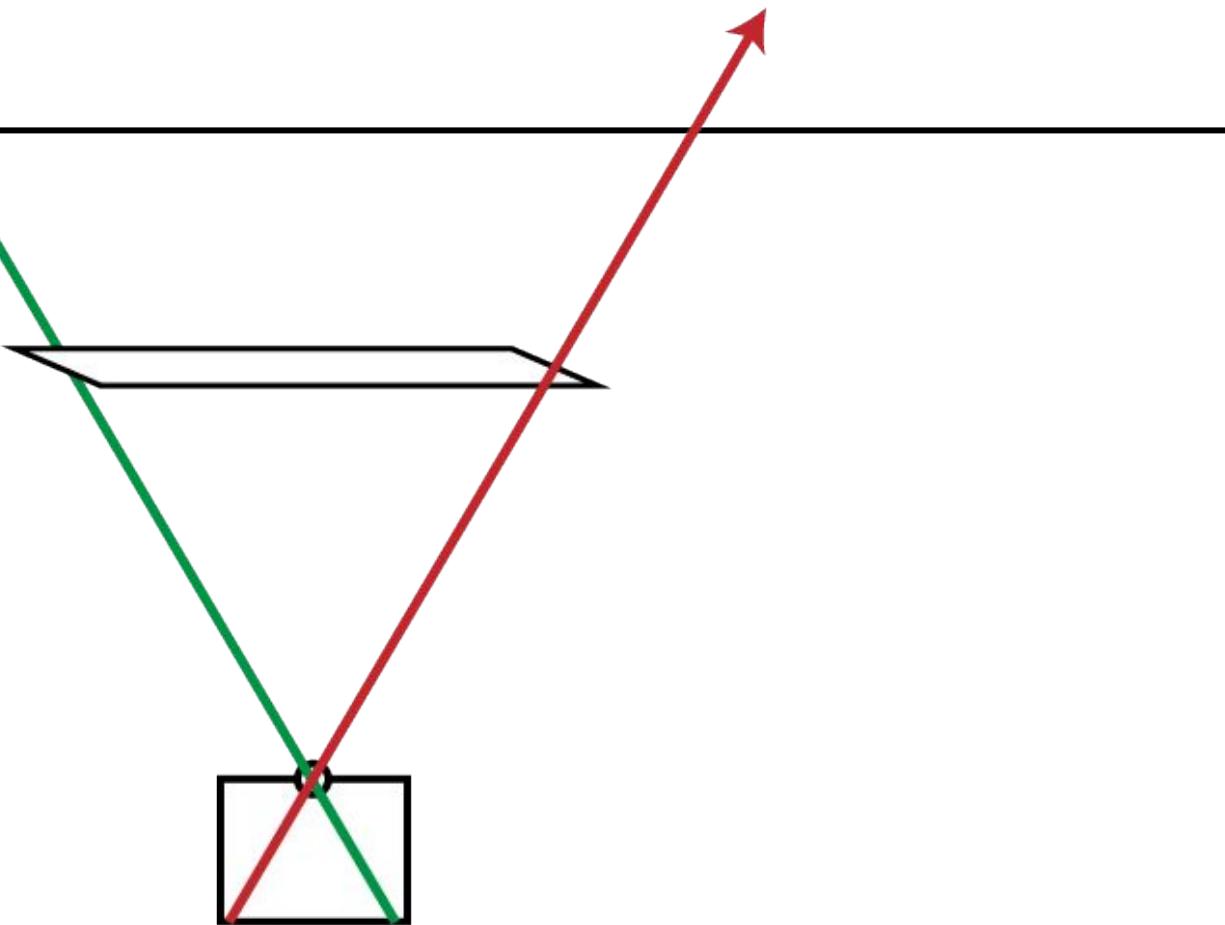
Ha



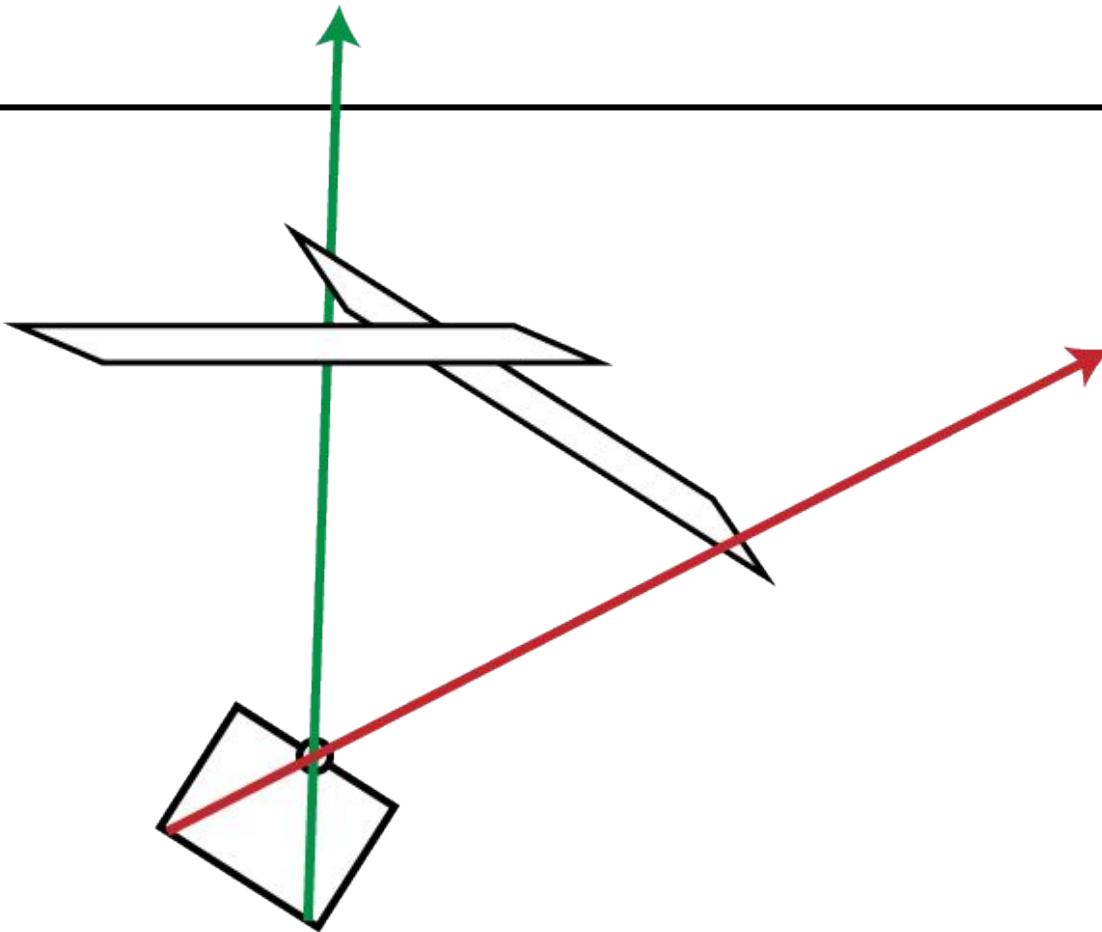




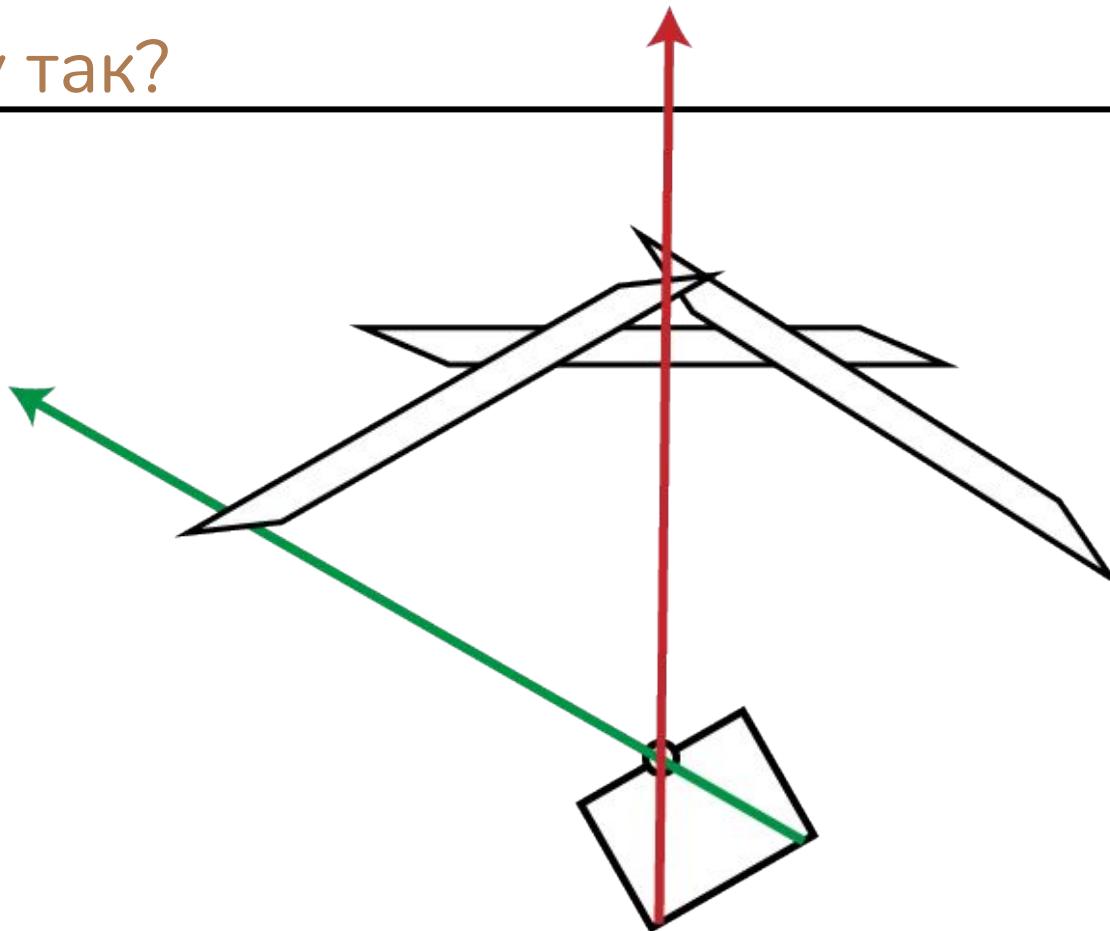
Почему так?



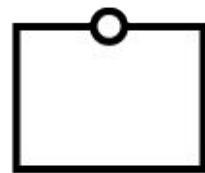
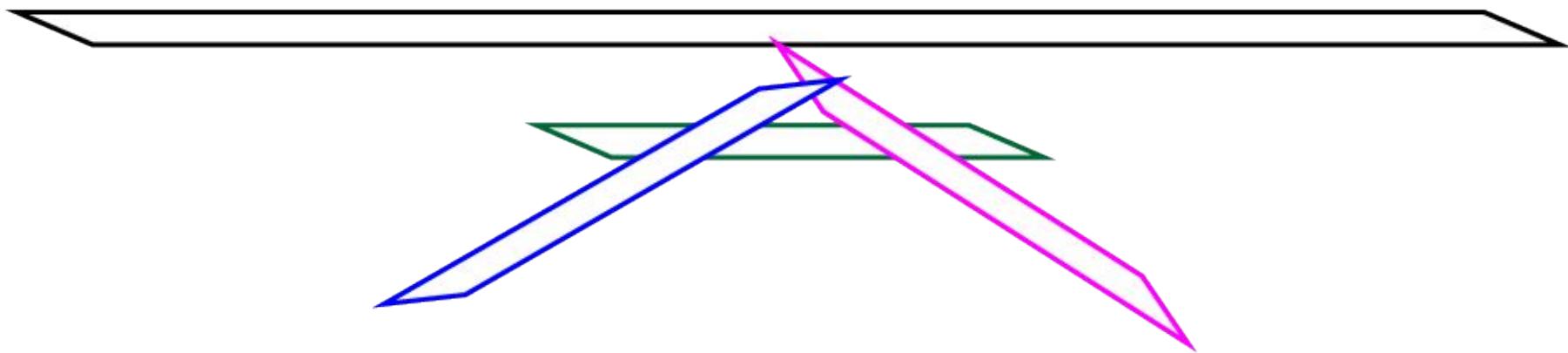
Почему так?



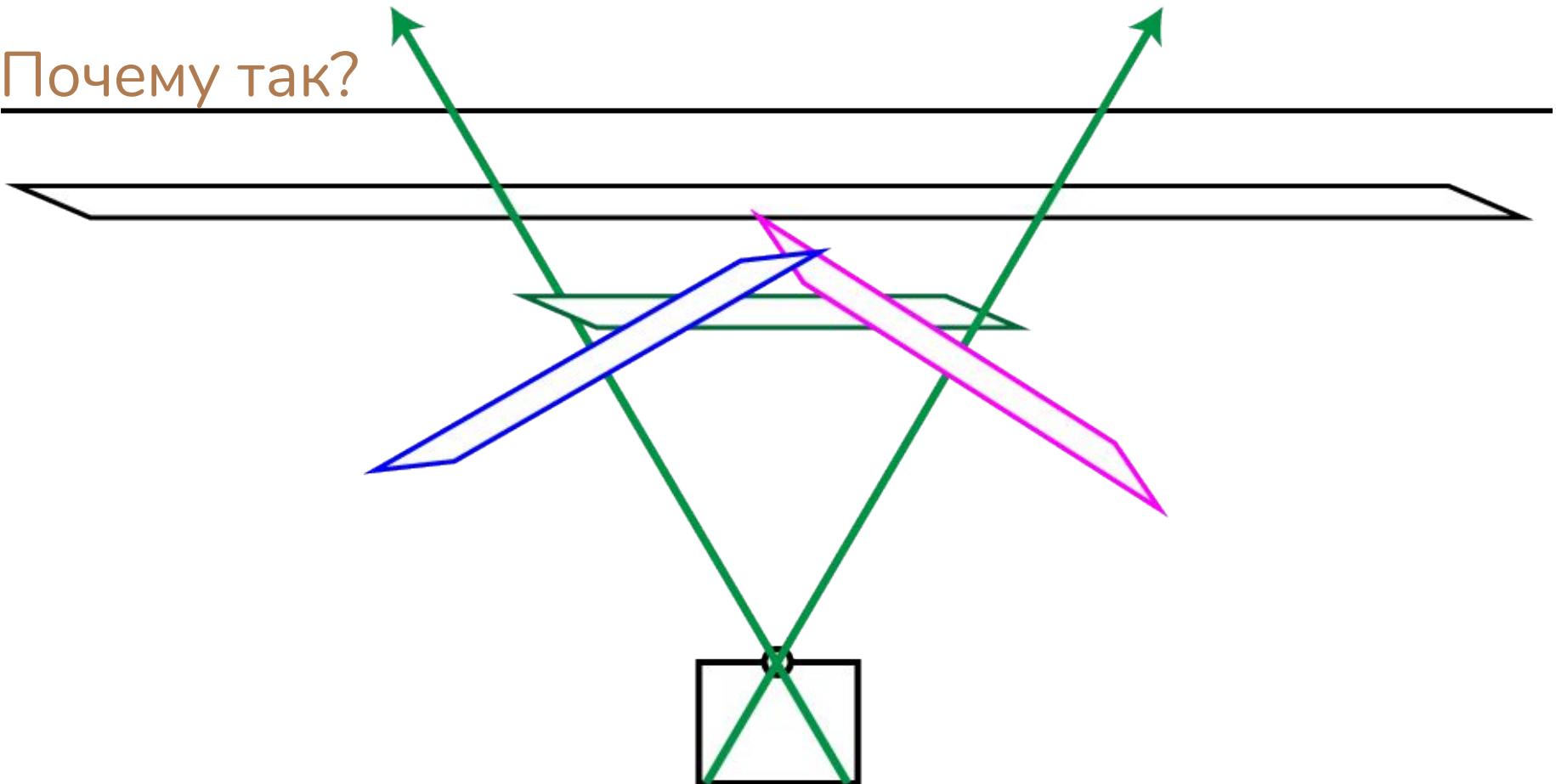
Почему так?



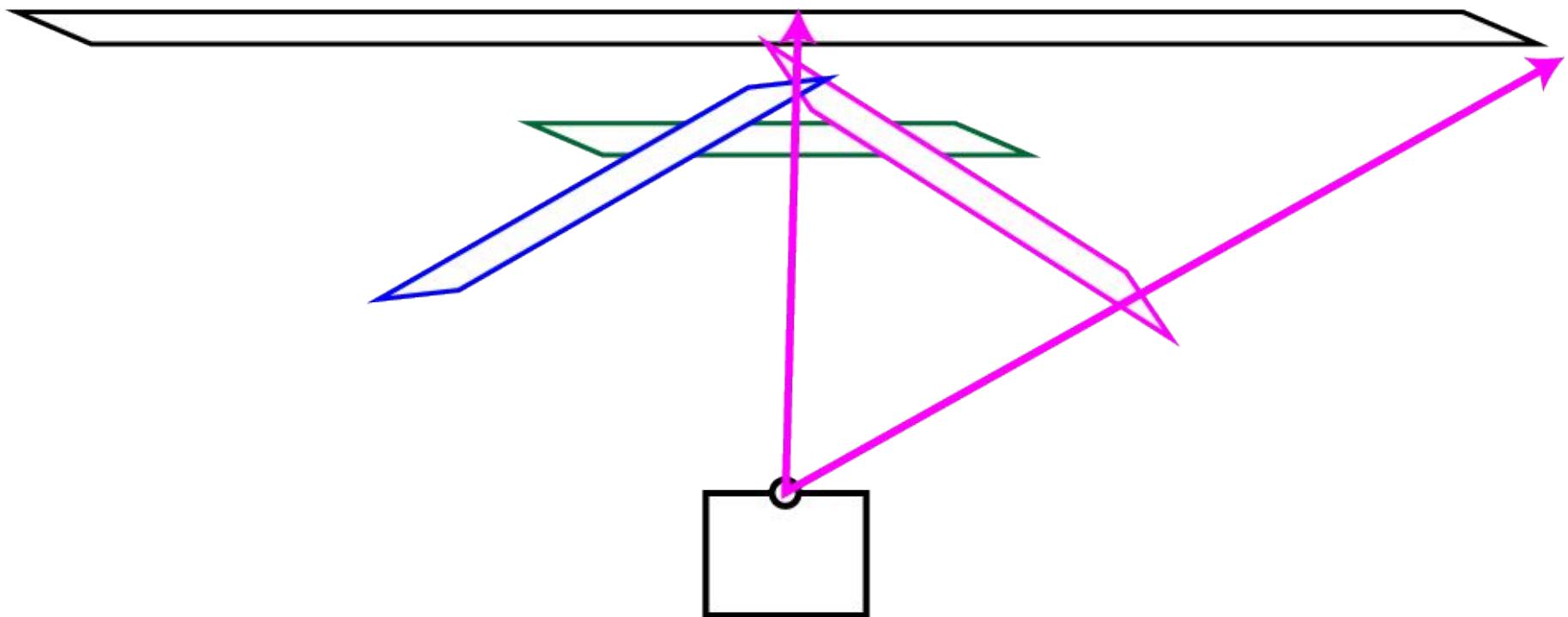
Почему так?



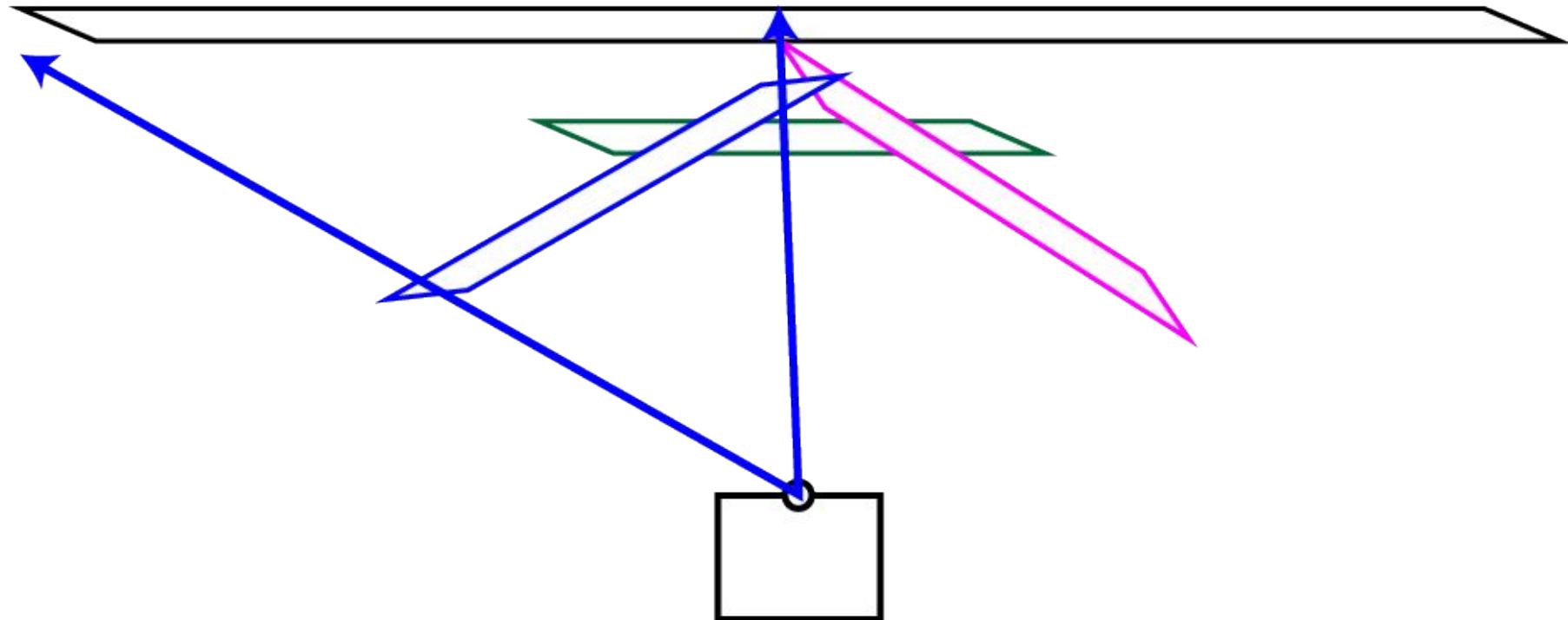
Почему так?



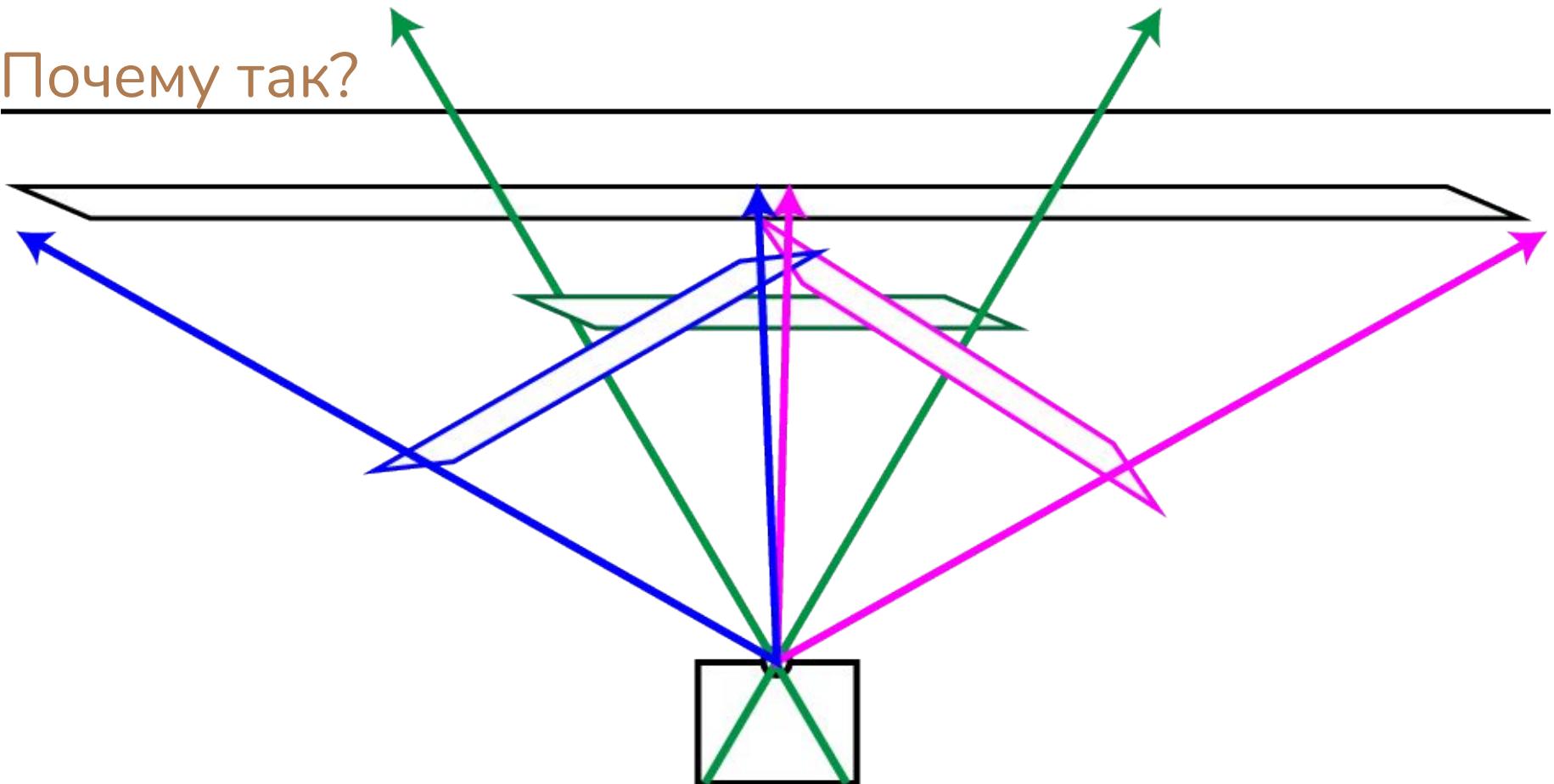
Почему так?



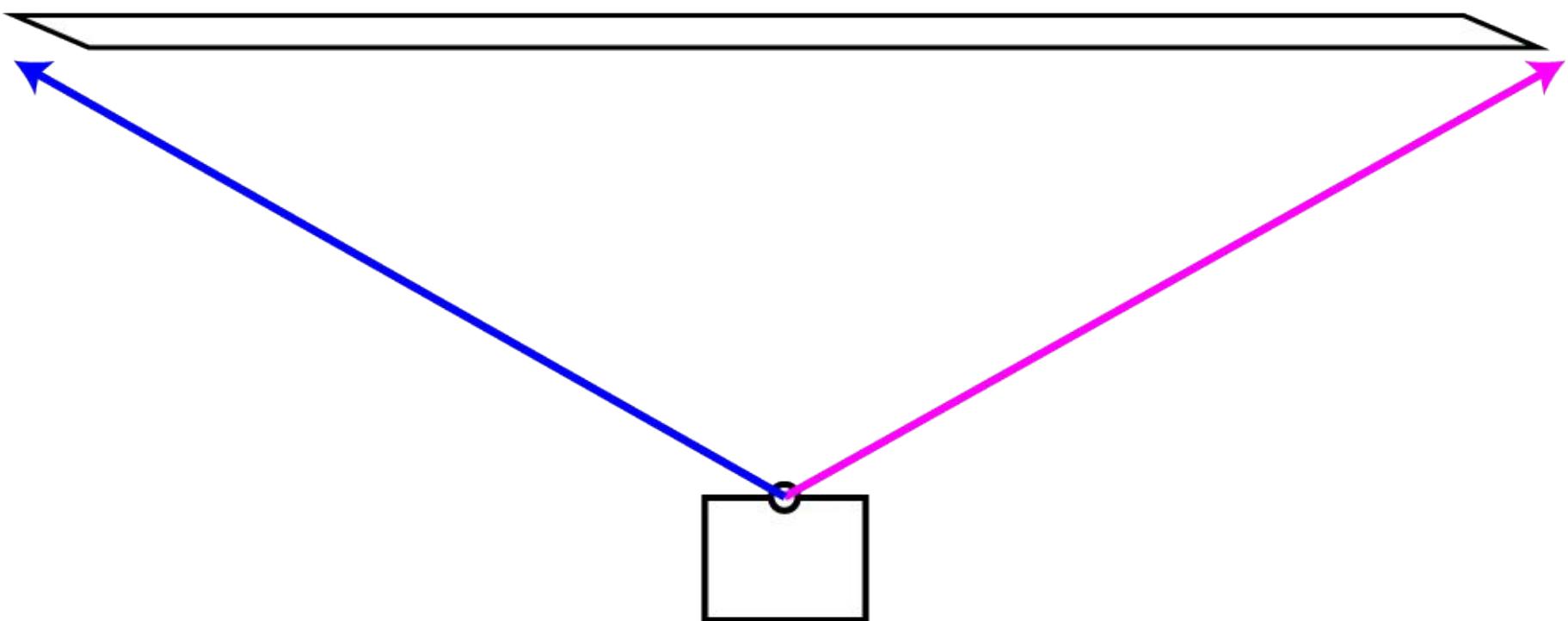
Почему так?



Почему так?



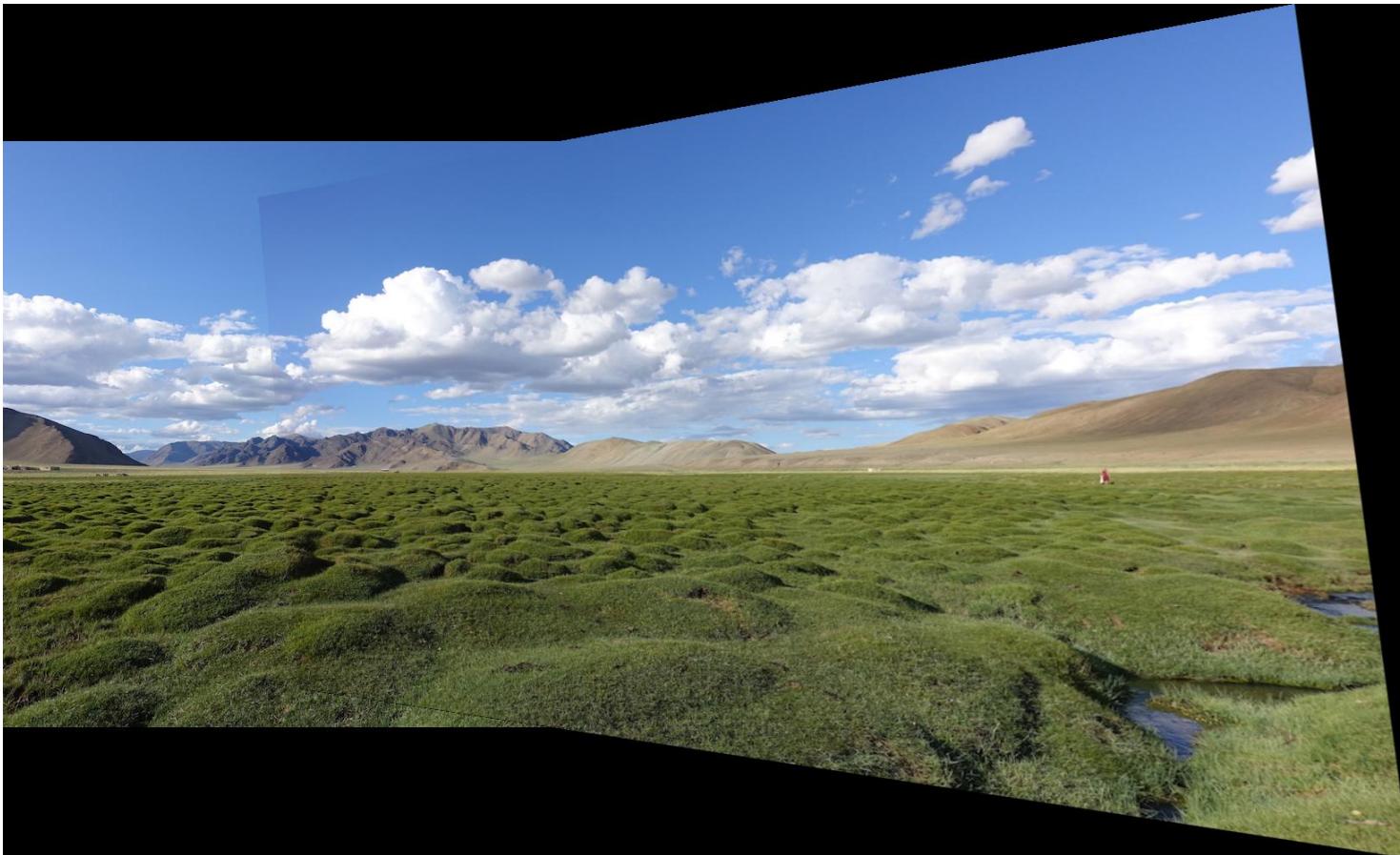
Почему так?



Очень плохо для больших панорам



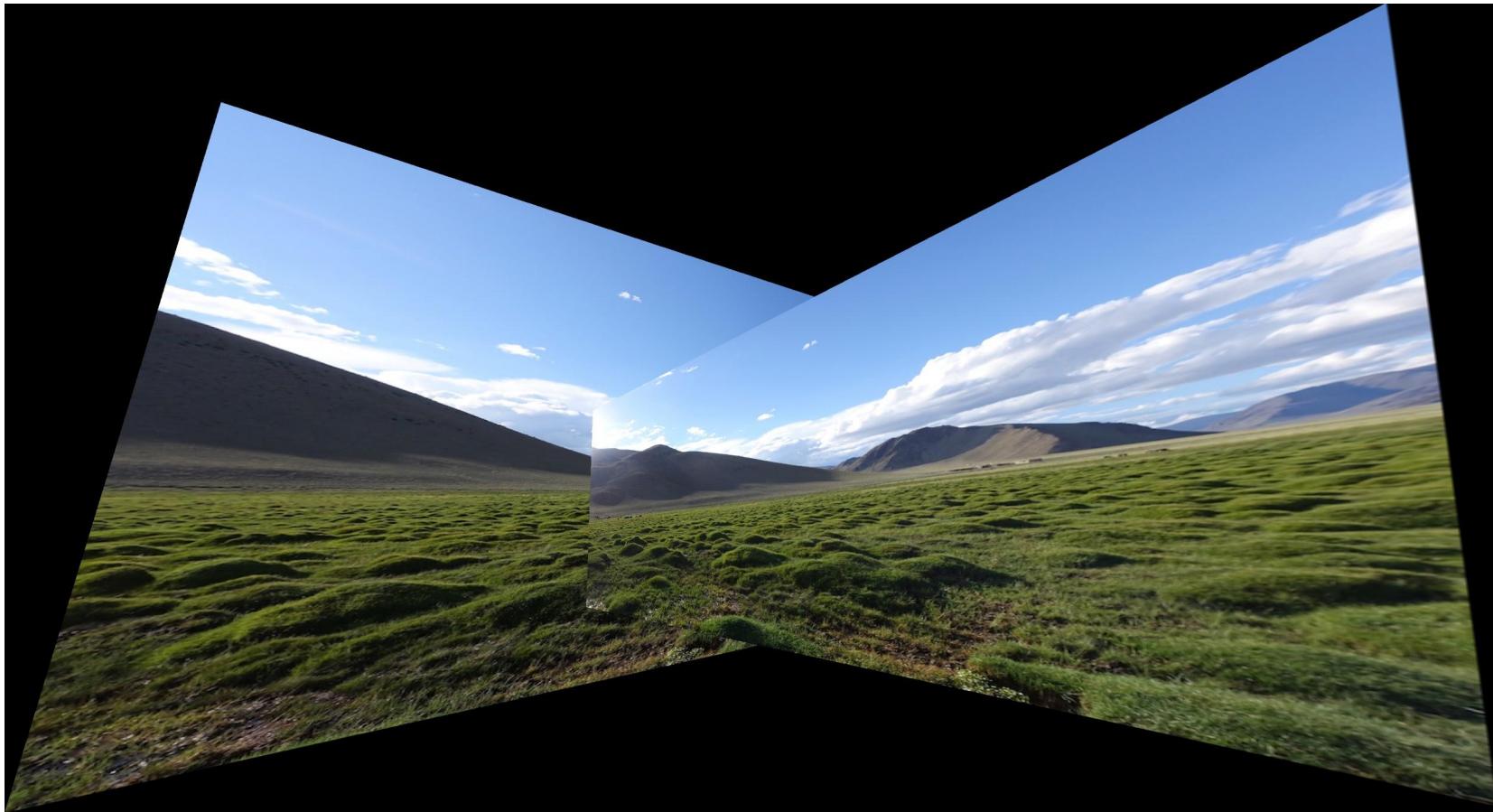
Очень плохо для больших панорам



Очень плохо для больших панорам



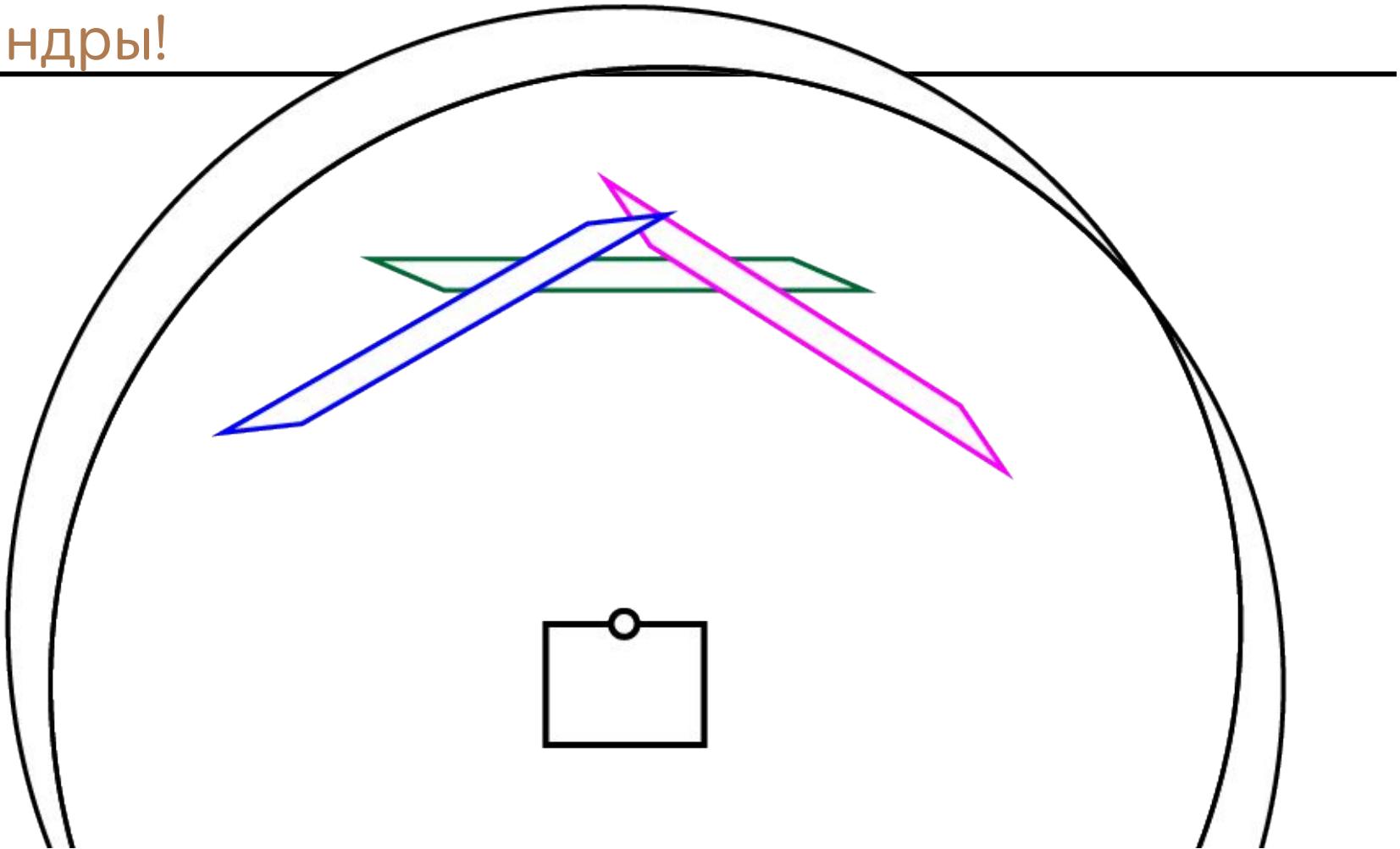
:(|



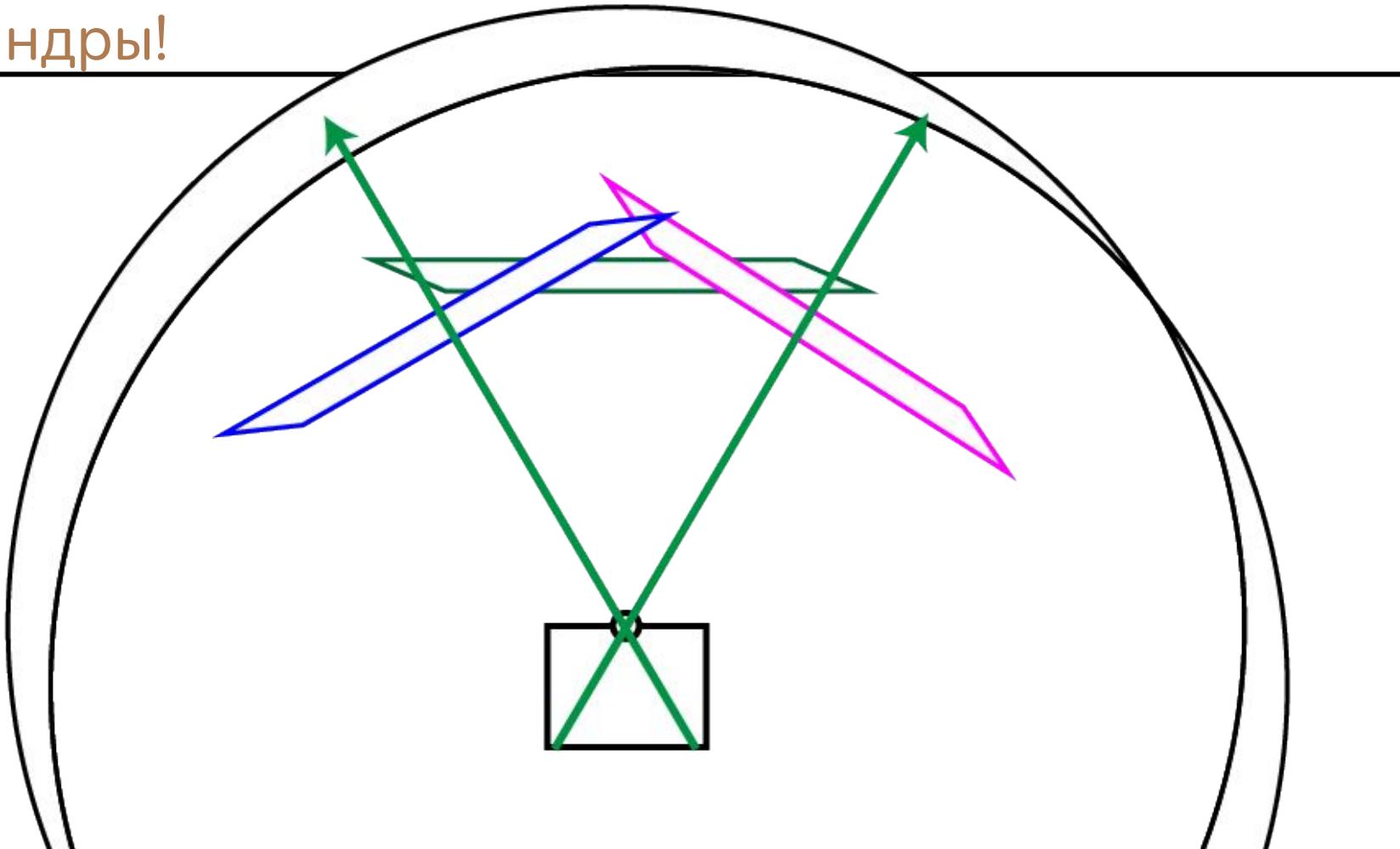
Как это исправить?

Цилиндры!

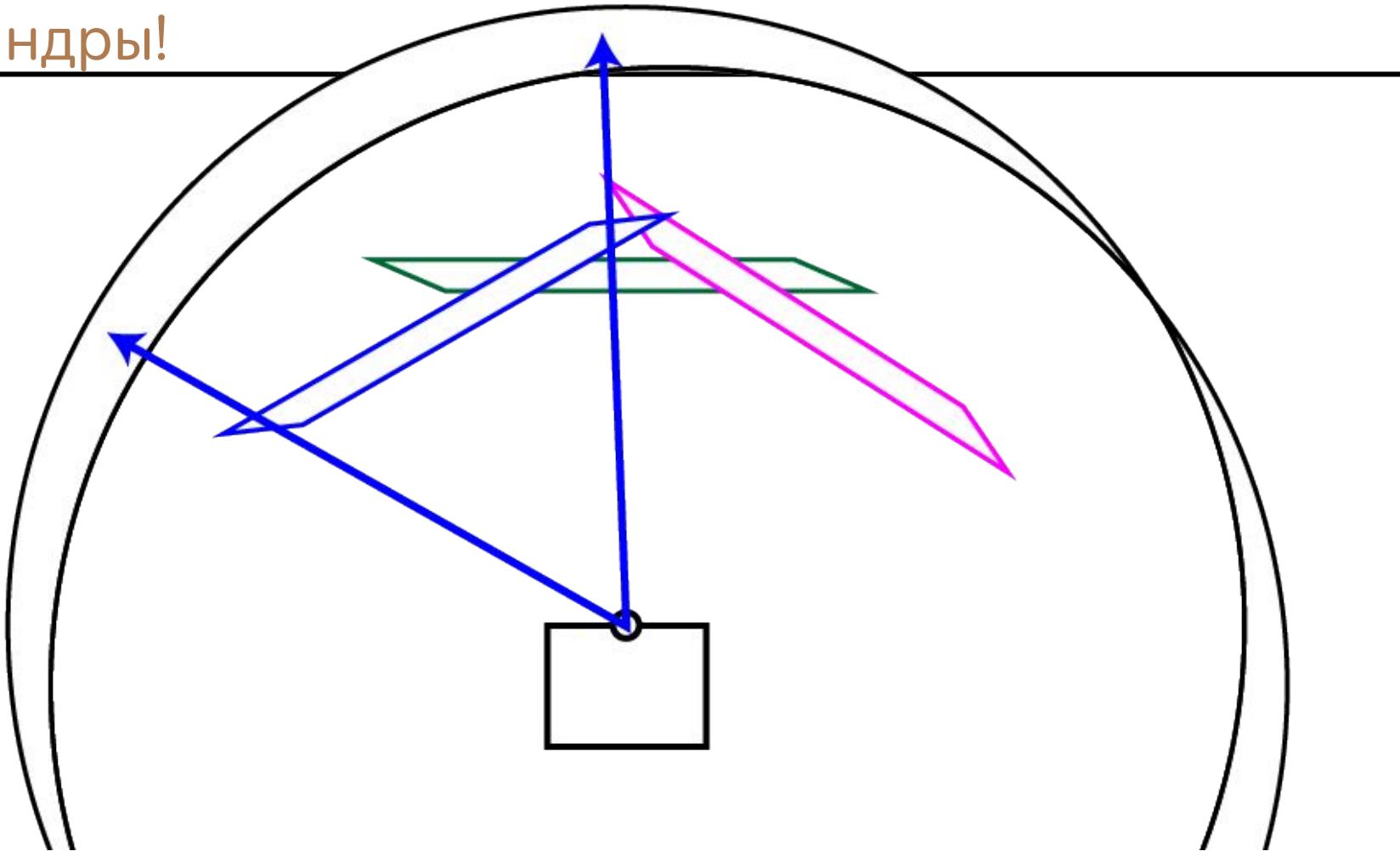
Цилиндры!



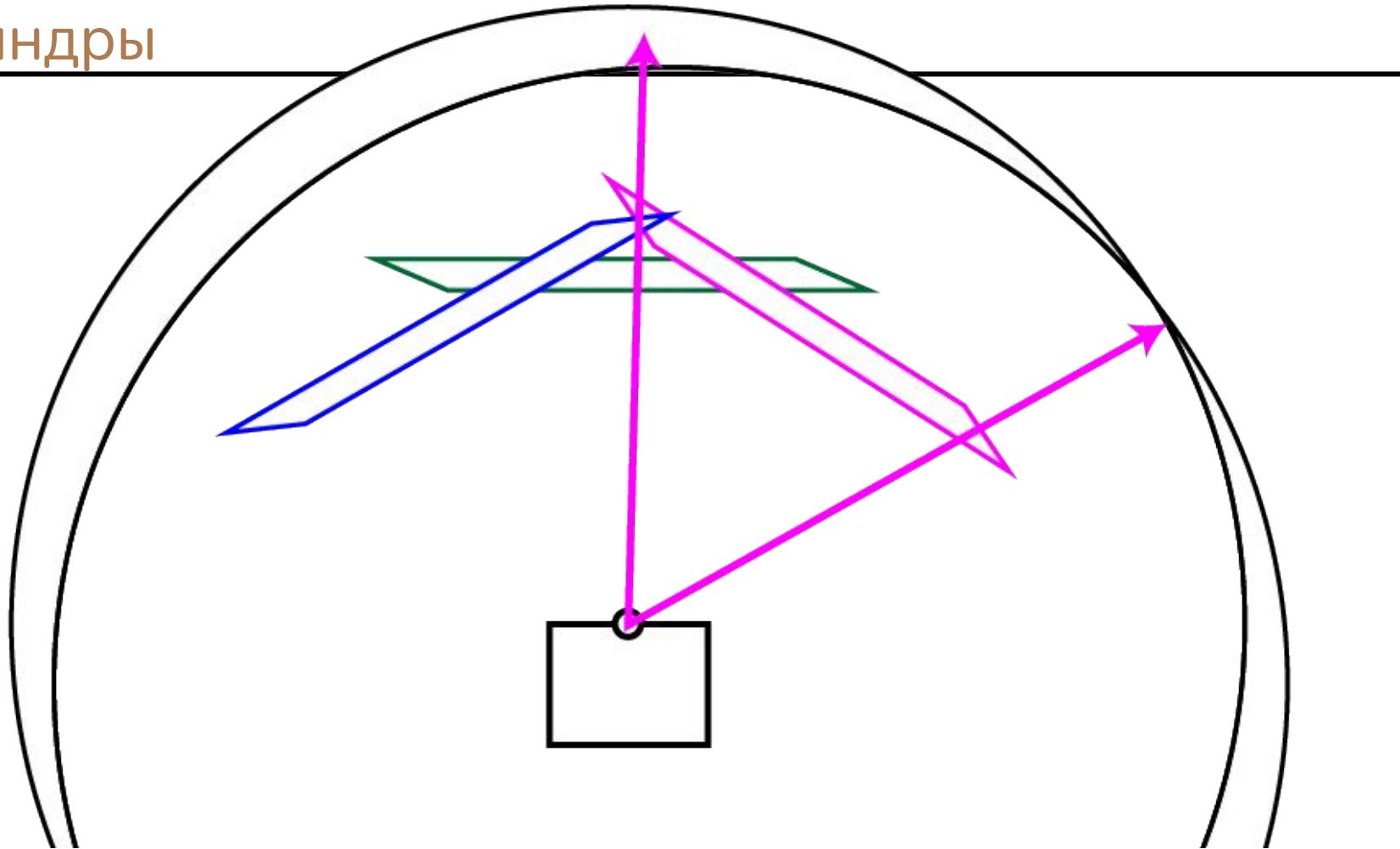
Цилиндры!



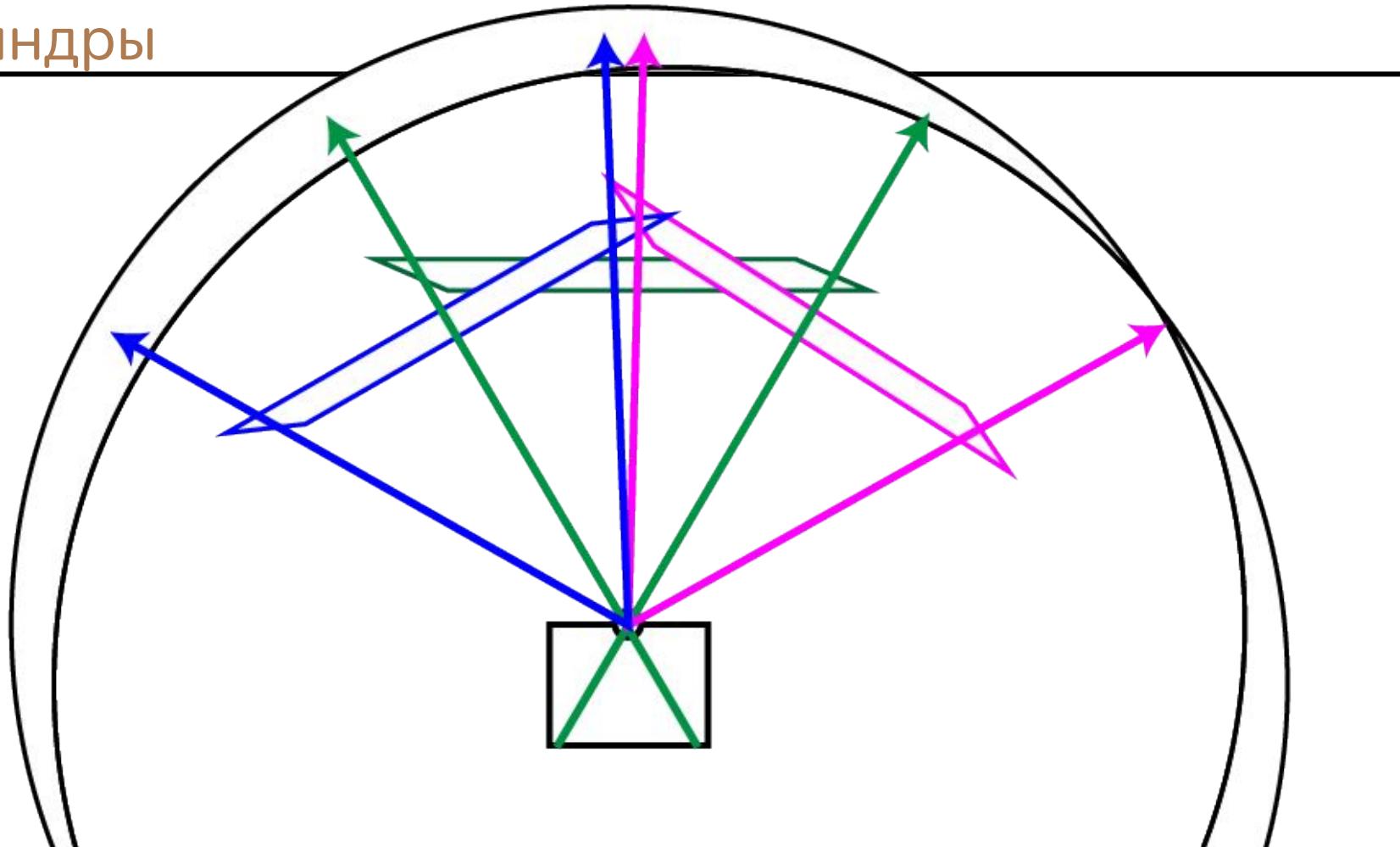
Цилинды!



Цилиндры



Цилиндры



Цилиндры

Calculate angle and height:

$$\theta = (x - x_c) / f$$

$$h = (y - y_c) / f$$

Find unit cylindrical coords:

$$X' = \sin(\theta)$$

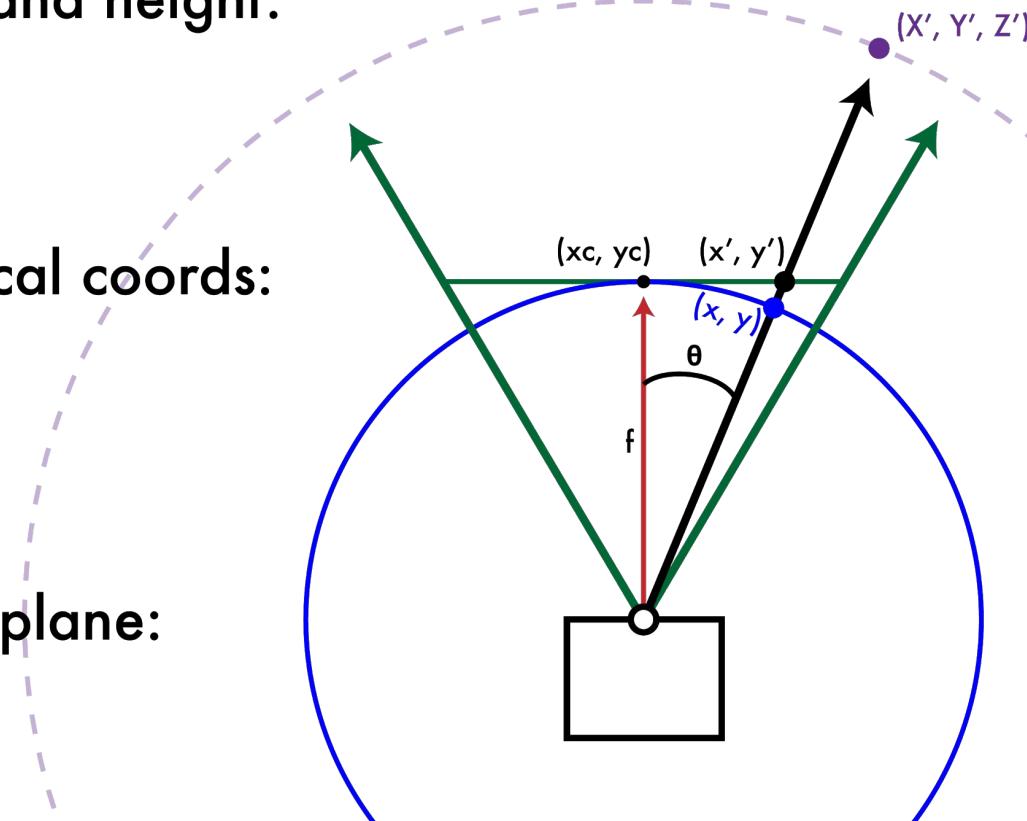
$$Y' = h$$

$$Z' = \cos(\theta)$$

Project to image plane:

$$x' = f X' / Z' + x_c$$

$$y' = f Y' / Z' + y_c$$



Фокусное расстояние



$f = 300$



$f = 500$



$f = 1000$



$f = 1400$



$f = 10,000$



Пример



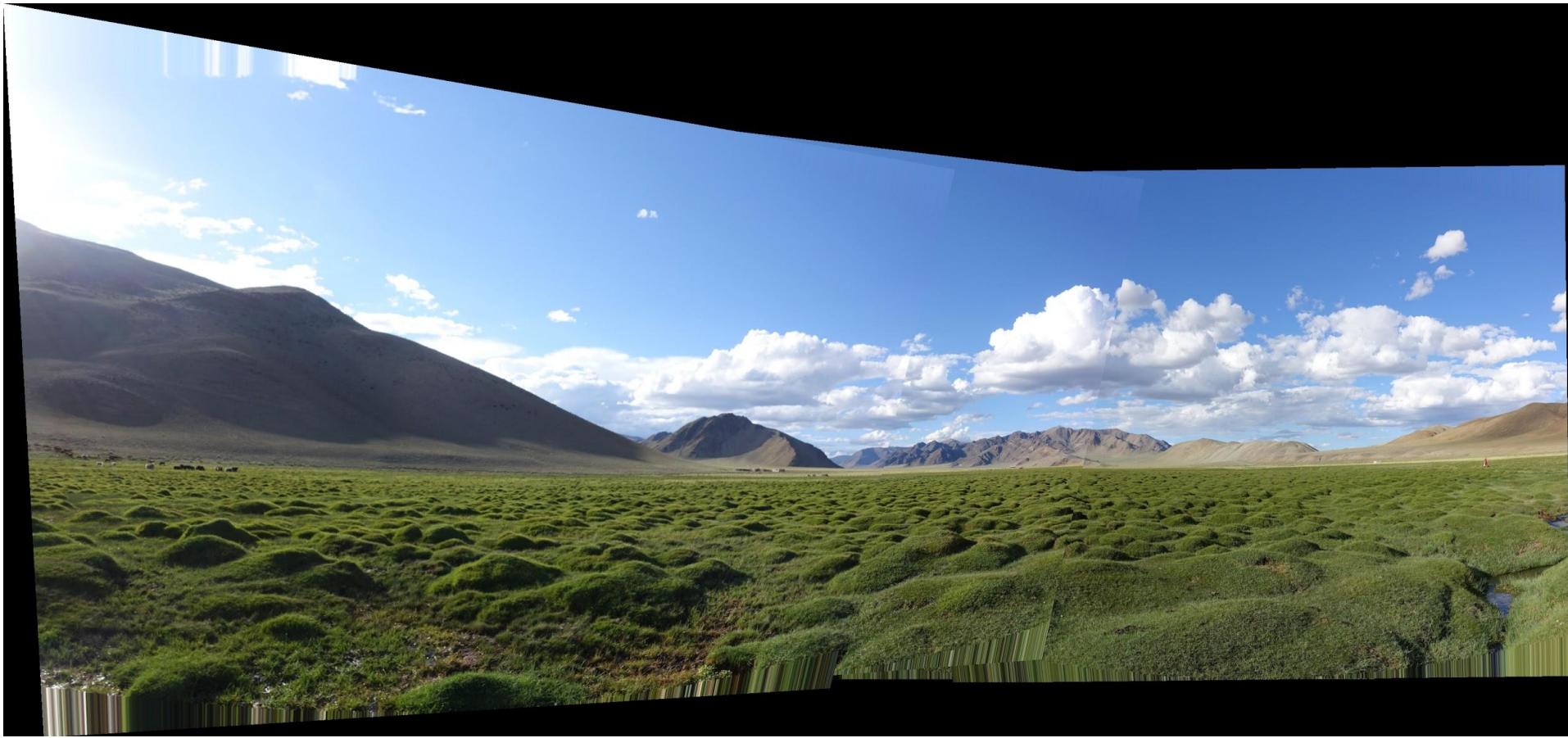
Пример



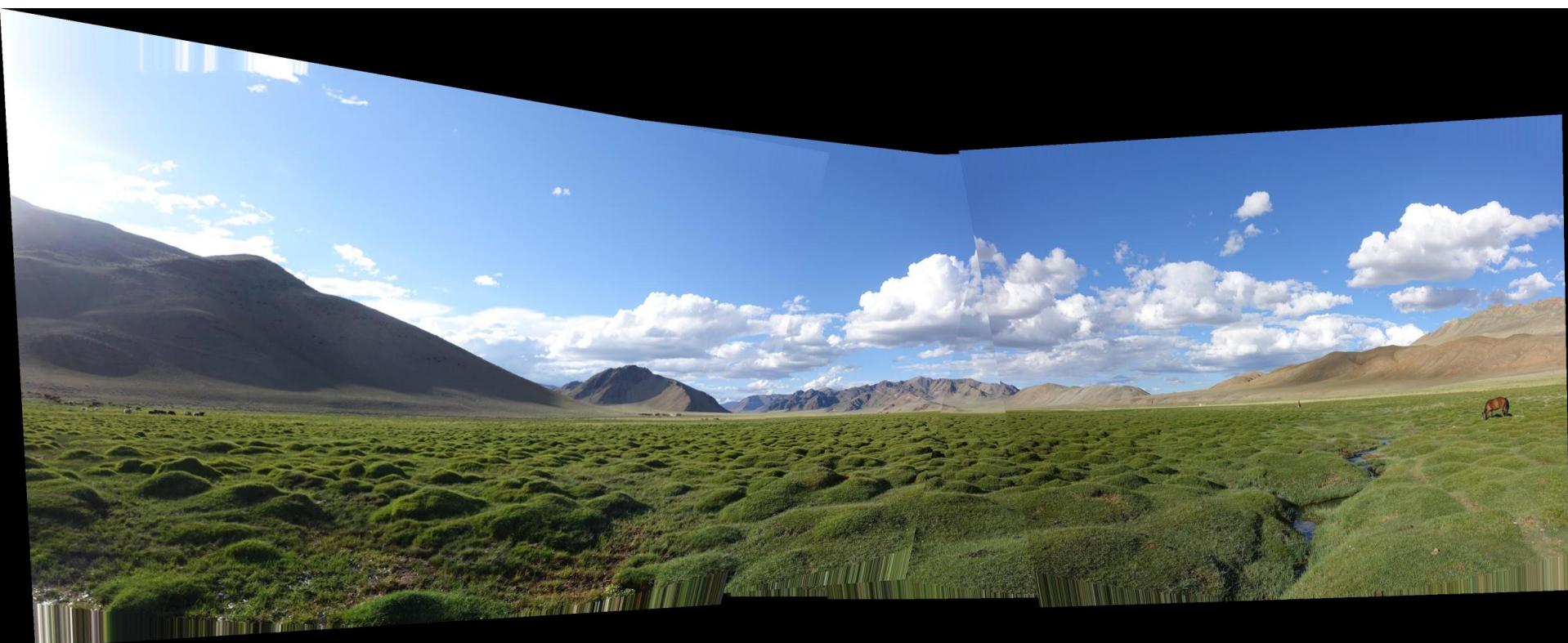
Пример



Пример



Пример

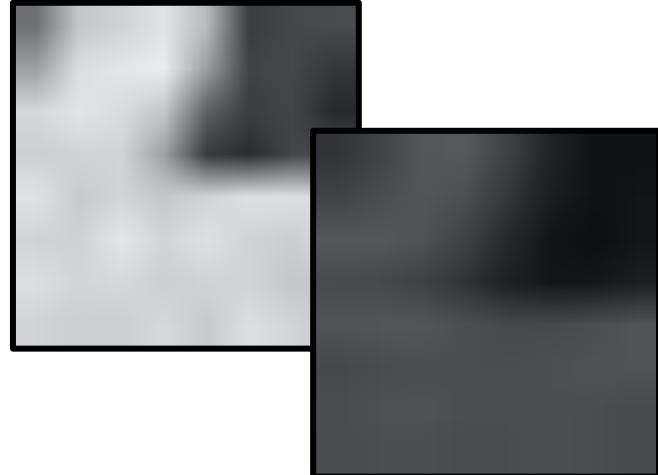
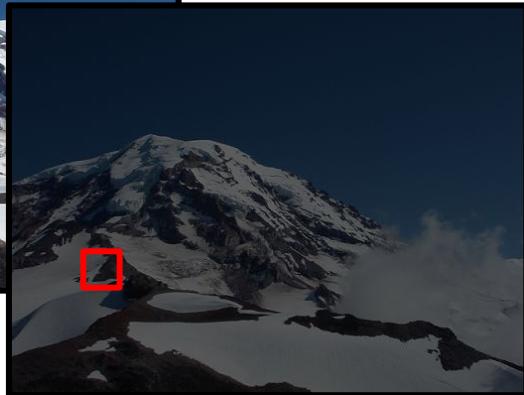


Пример



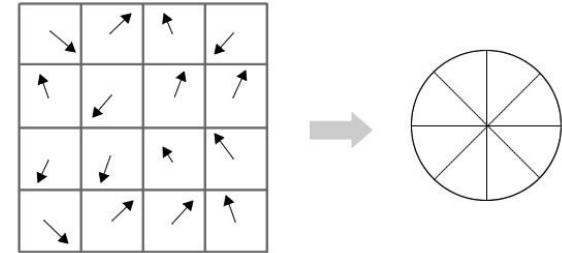
Вернемся к дескрипторам

- Тривиальный дескриптор (“сырые” значения пикселей) - плохо
 - Не инвариантны к изменению освещенности!
 - К поворотам
- Хотим инвариантные к преобразованиям дескрипторы

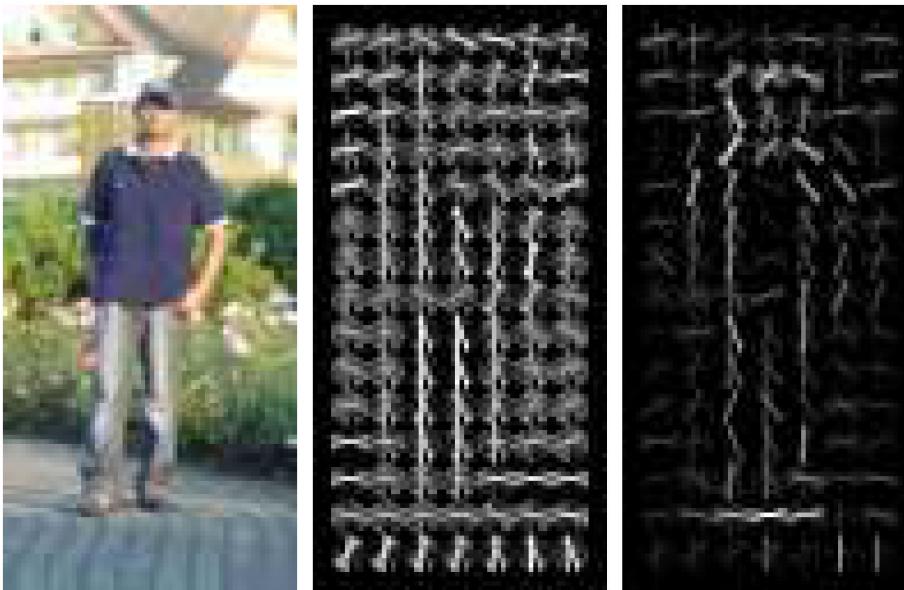


Histogram of Oriented Gradients (HOG)

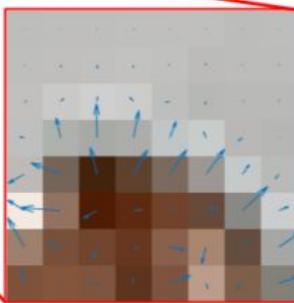
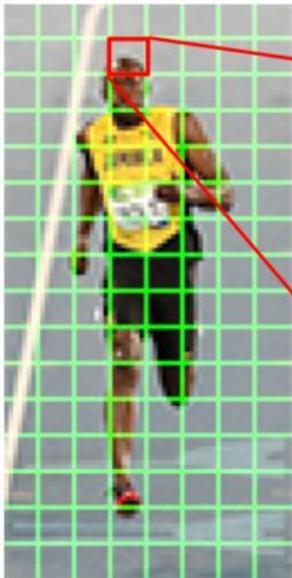
- Dalal and Triggs, 2005
(но придумали еще раньше)
- Неплохой дескриптор
 - Вычислить градиенты
 - Разбить на блоки (4x4, 16x16 cells)
 - Построить гистограммы
 - Нормализовать гистограммы
- Абсолютные значения магнитуды не важны
 - Позволяет бороться с изменением освещенности
- Обучаем SVM на распознавание людей



Histogram of Oriented Gradients (HOG)



Histogram of Oriented Gradients (HOG)



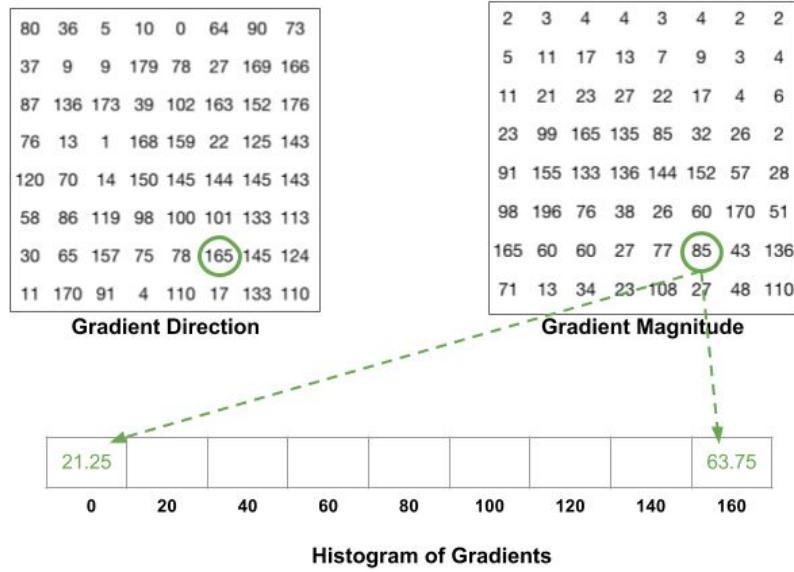
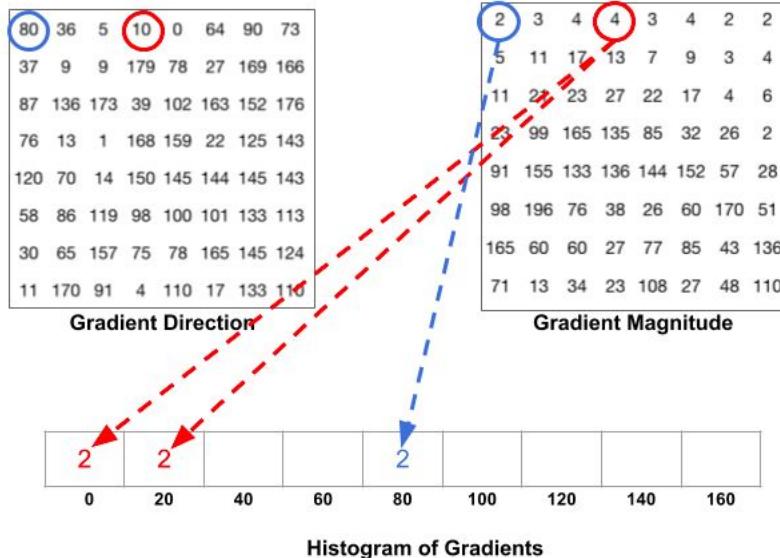
2	3	4	4	3	4	2	2
5	11	17	13	7	9	3	4
11	21	23	27	22	17	4	6
23	99	165	135	85	32	26	2
91	155	133	136	144	152	57	28
98	196	76	38	26	60	170	51
165	60	60	27	77	85	43	136
71	13	34	23	108	27	48	110

Gradient Magnitude

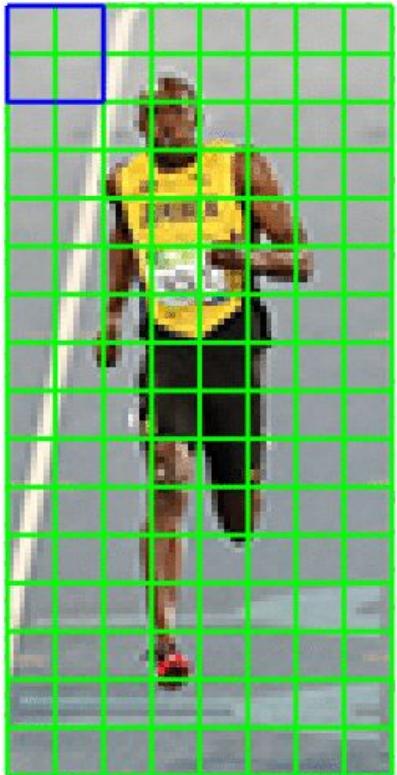
80	36	5	10	0	64	90	73
37	9	9	179	78	27	169	166
87	136	173	39	102	163	152	176
76	13	1	168	159	22	125	143
120	70	14	150	145	144	145	143
58	86	119	98	100	101	133	113
30	65	157	75	78	165	145	124
11	170	91	4	110	17	133	110

Gradient Direction

Histogram of Oriented Gradients (HOG)



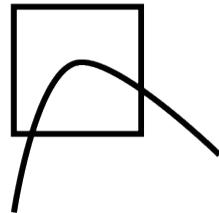
Histogram of Oriented Gradients (HOG)



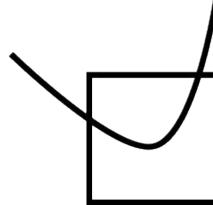
Еще проблемы

- Детектор Харриса:
 - Инвариантен к поворотам
 - Не инвариантен к масштабированиям
- Дескрипторы:
 - Пиксельное описание - не инвариантно к поворотам
 - HOG - также не инвариантно к поворотам

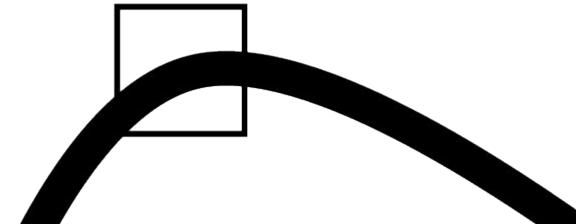
Corner



Still Corner



Not Corner

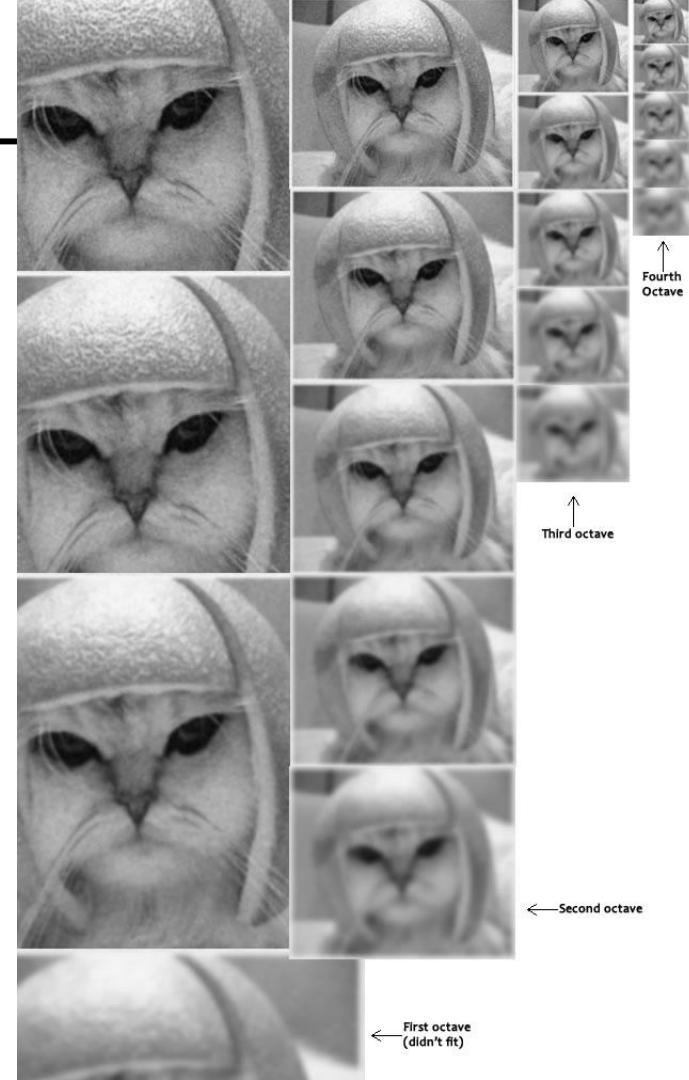


SIFT

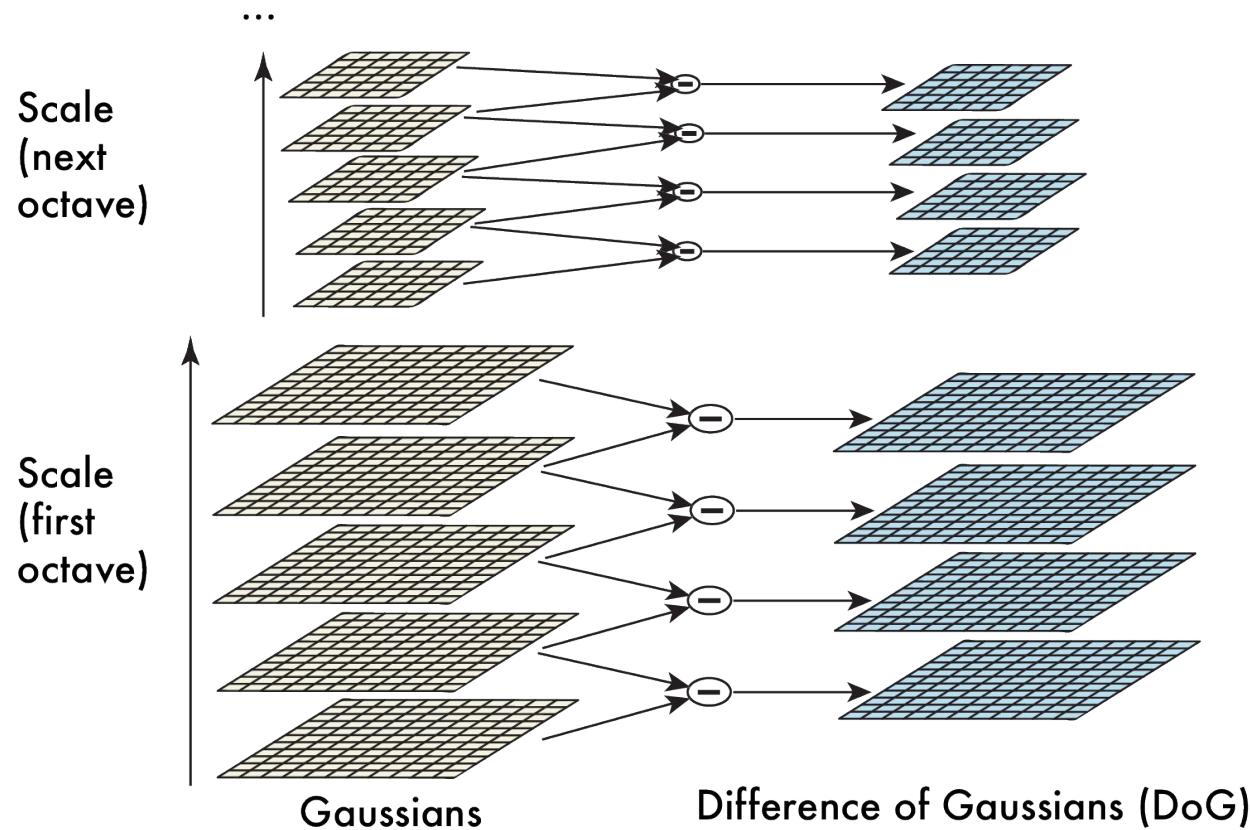
- Scale Invariant Feature Transform (SIFT)
 - Lowe et al. 2004
 - Самая цитируемая статья в CV
 - Патент закончился этой весной
- Построить scale space
- Найти ключевые точки
- Построить устойчивый к повороту к дескриптор
 - Нормализовать по направлению
 - Нормализовать по свету

Интерактивный [тutorиал](#)

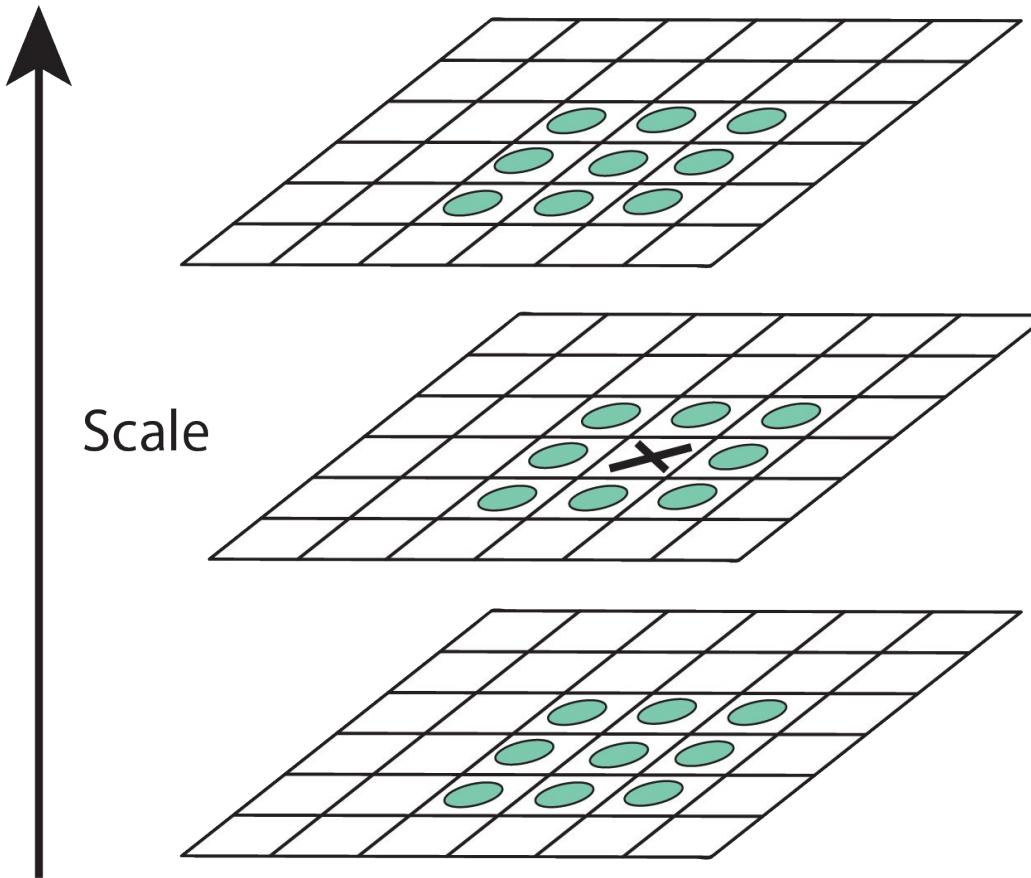
Scale space



DoG



Нахождение локальных максимумов



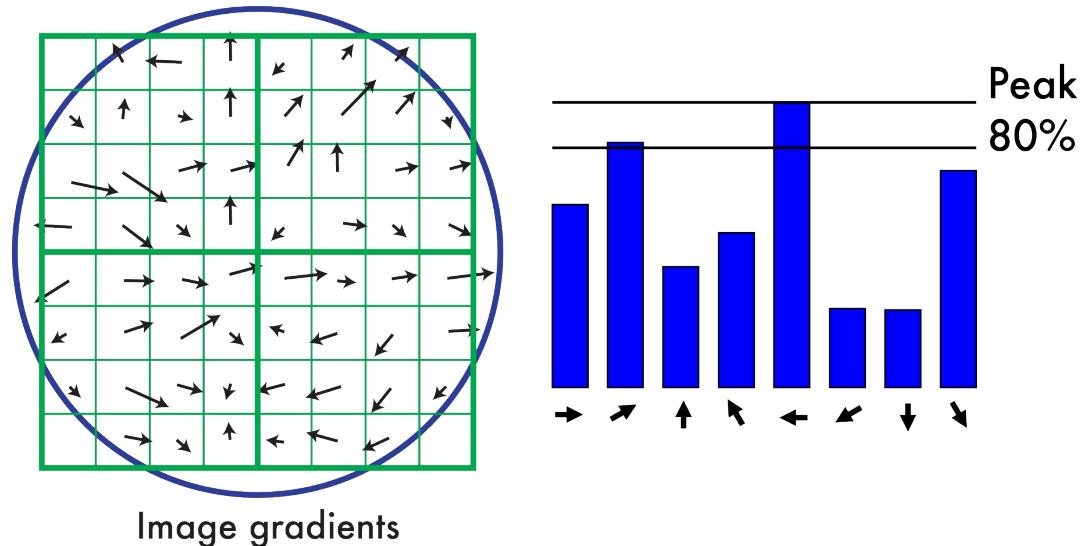
Убрать лишние точки

- Вычислить градиенты
 - Не по всему изображению, а по избранным точкам. Быстрее!
 - Убрать точки со слабым откликом
- ФУНКЦИЯ ОТКЛИКА УГЛА
 - То же самое, structure matrix, детерминант и след матрицы
 - r - масштаб, на котором рассматривается точка

$$\frac{\text{Tr}(\mathbf{H})^2}{\text{Det}(\mathbf{H})} < \frac{(r+1)^2}{r}$$

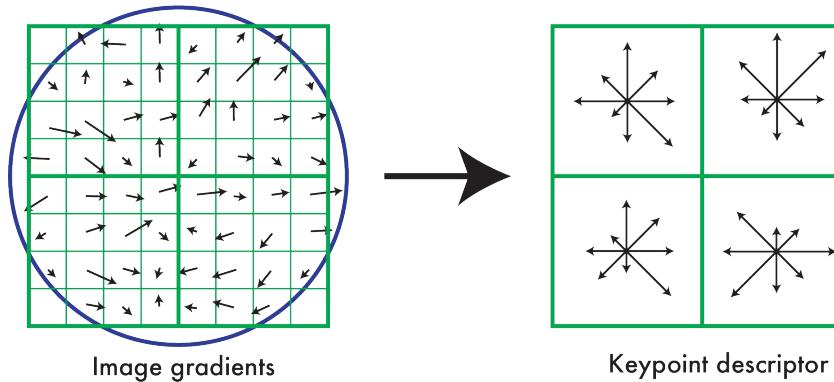
Нахождение ориентации региона

- Взвешенная гистограмма градиентов соседей
 - Любой градиент больший или равный 80 % значения пика, получает свой дескриптор
 - Один пиксель -> несколько ключевых точек
 - Дескрипторы смещаются, чтобы главное направление стояло первым



Дескриптор (аля HOG)

- Разделить на подобласти (2×2 , 4×4)
- В каждой подобласти построить гистограмму
 - Нормализовать
 - Столбцы большие чем 0.2, обрезать до 0.2
 - Снова нормализовать
 - Позволяет бороться с изменением освещения

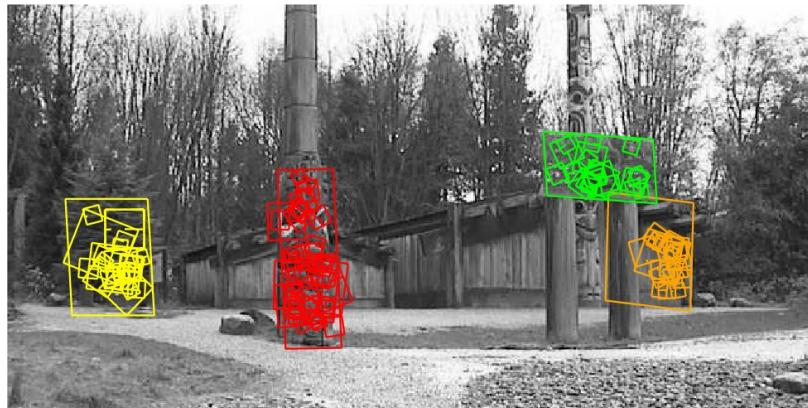


SIFT

- Находит хорошие точки, описывает их
- Нахождение объектов, распознавание, панорамы, ...



SIFT is great!





It's coding time!