

Введение в фотограмметрию

Сопоставление ключевых точек



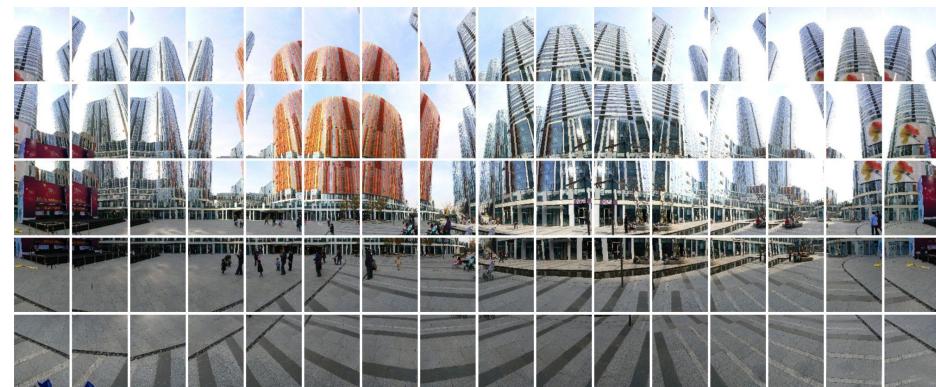
Фотограмметрия. Лекция 3

- Brute-force & ANN feature matching
- Фильтрация сопоставлений
- Guided matching

Симиутин Борис
simiyutin.boris@yandex.ru

Мотивация

- 1) Научились находить хорошие ключевые точки с инвариантным дескриптором
- 2) Хотим объединить снимки, например, склеить в панораму



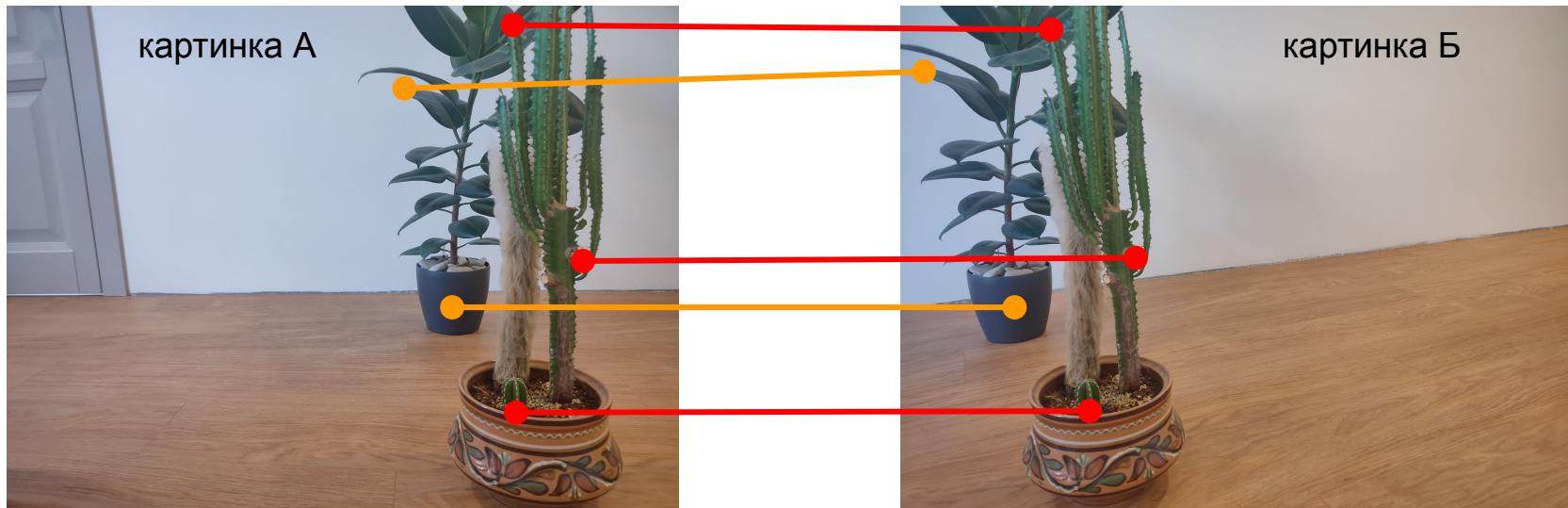
[Источник картинки](#)

Панорама полок магазинов (для мерчендайзеров)



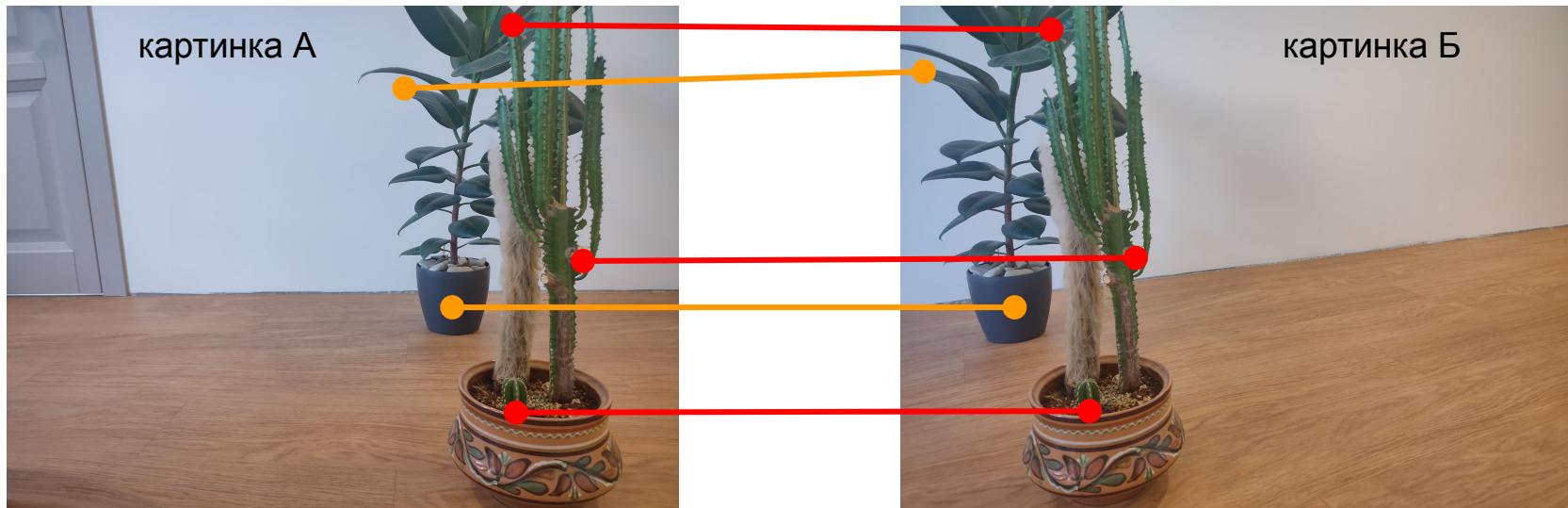
Feature matching

- 1) Нужны сопоставления между найденными ключевыми точками:
 - a) Brute-force matching



Feature matching

- 1) Нужны сопоставления между найденными ключевыми точками:
 - a) Brute-force matching
 - b) ANN matching



Brute-force matching

- 1) Полный перебор: для каждой ключевой точки ищем ближайшую на соседнем изображении
- 2) Сложность $O(N^2 * M^2)$
 N - число ключевых точек на кадр, M - число кадров
- 3) Расстояние:
 - **SIFT Descriptor:** 128 Float = 512 bytes
 - Для пары SIFT дескрипторов A, B: $dist(A, B) = \sqrt{\sum_{i=1}^{128} (A_i - B_i)^2}$
 - Для пары бинарных дескрипторов A, B: $dist(A, B) = popcnt(A \ xor \ B)$

Brute-force matching

Сложность $O(N^2 * M^2)$

N - число ключевых точек на кадр, M - число кадров

Brute-force matching

Сложность $O(N^2 * M^2)$

N - число ключевых точек на кадр, **M** - число кадров

Как можно ускорить?

Ускорение вклада N^2 :

- Использовать бинарный дескриптор вместо floating-point, тогда доступна быстрая инструкция **popcnt()**
- Обычно **N** - универсальная константа => можно дожать железом: использовать **видеокарты!**

$N = 2000$, тогда время на CPU ~ 1 sec, на GPU ~ 0.01 sec

- Использовать не Brute-force matching :)

ANN matching

- 1) Если есть только CPU и/или только floating-point дескриптор, то приходится жертвовать качеством ради скорости
- 2) Как обычно ускоряют поиск ближайших соседей?

ANN matching

- 1) Если есть только CPU и/или только floating-point дескриптор, то приходится жертвовать качеством ради скорости
- 2) Как обычно ускоряют поиск ближайших соседей? Например, **KD-tree**

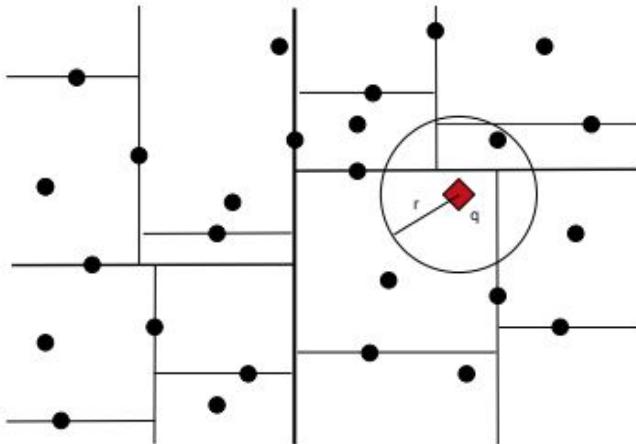


Figure 2: Illustration of space partition generated by the kd-tree algorithm in \mathbb{R}^2 . The width of the separating lines indicate the depth in the tree of the corresponding split.

ANN matching

- 1) Если есть только CPU и/или только floating-point дескриптор, то приходится жертвовать качеством ради скорости
- 2) Как обычно ускоряют поиск ближайших соседей? Например, **KD-tree**
- 3) В чем в нашем случае проблема? (SIFT descriptor, 128 floats)

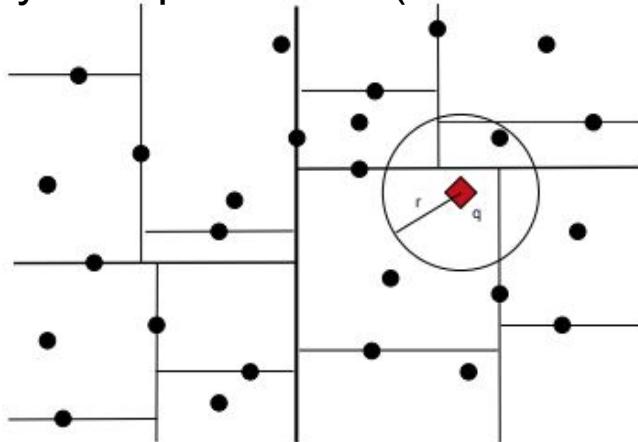
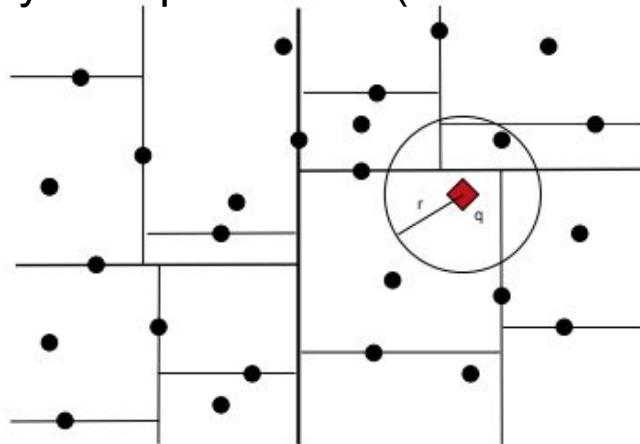


Figure 2: Illustration of space partition generated by the kd-tree algorithm in \mathbb{R}^2 . The width of the separating lines indicate the depth in the tree of the corresponding split.

ANN matching

- 1) Если есть только CPU и/или только floating-point дескриптор, то приходится жертвовать качеством ради скорости
- 2) Как обычно ускоряют поиск ближайших соседей? Например, **KD-tree**
- 3) В чем в нашем случае проблема? (SIFT descriptor, 128 floats)



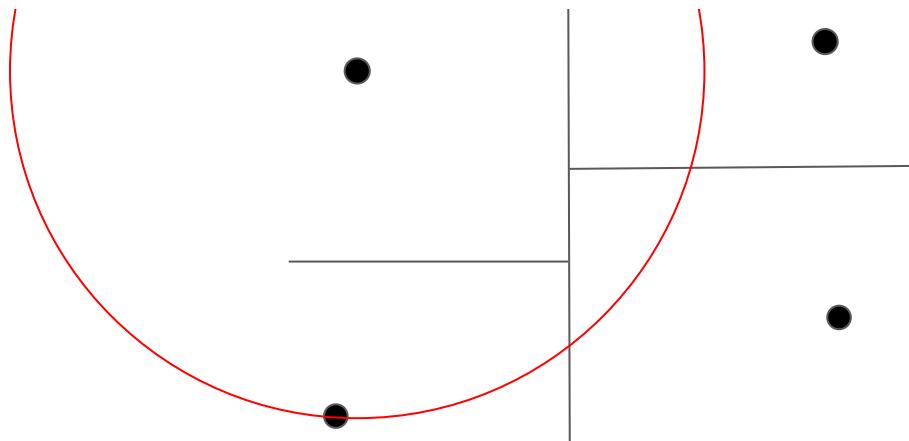
Dimensionality curse

[Nearest-neighbor search and curse of dimensionality](#)

Figure 2: Illustration of space partition generated by the kd-tree algorithm in \mathbb{R}^2 . The width of the separating lines indicate the depth in the tree of the corresponding split.

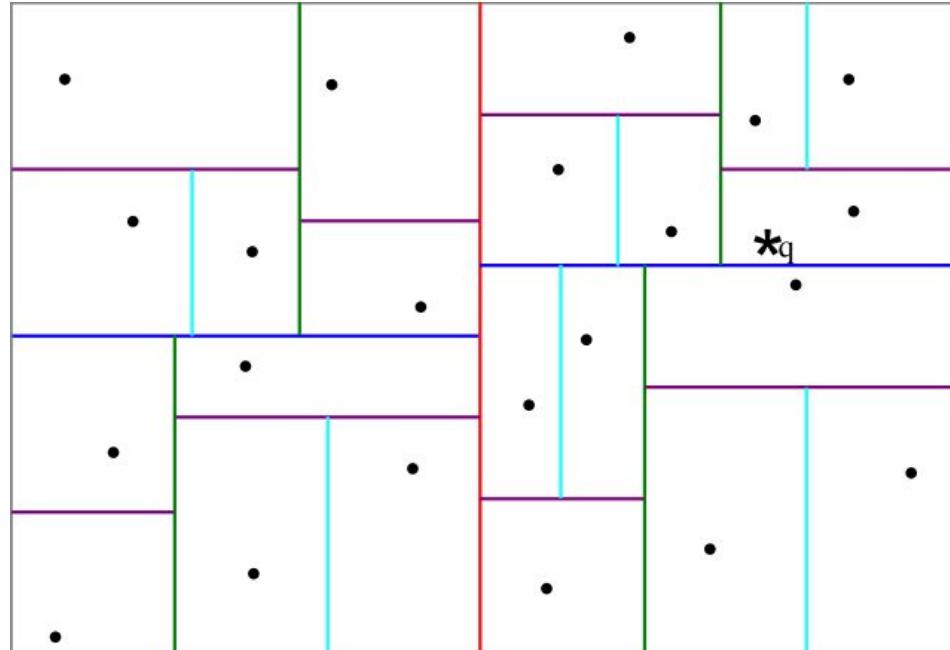
Dimensionality curse

- 1) Когда у нас большое количество измерений, то почти все соседи находятся далеко, и для поиска ближайшего приходится просматривать целиком большие клеточки иерархии -> поиск приближается к полному перебору



FLANN

- 1) Для того чтобы сохранить ускорение, приходится жертвовать точностью
- 2) Посещаем не все листья а фиксированное количество
- 3) Чтобы уменьшить рычаг ошибки на верхних уровнях, строим несколько рандомизированных деревьев

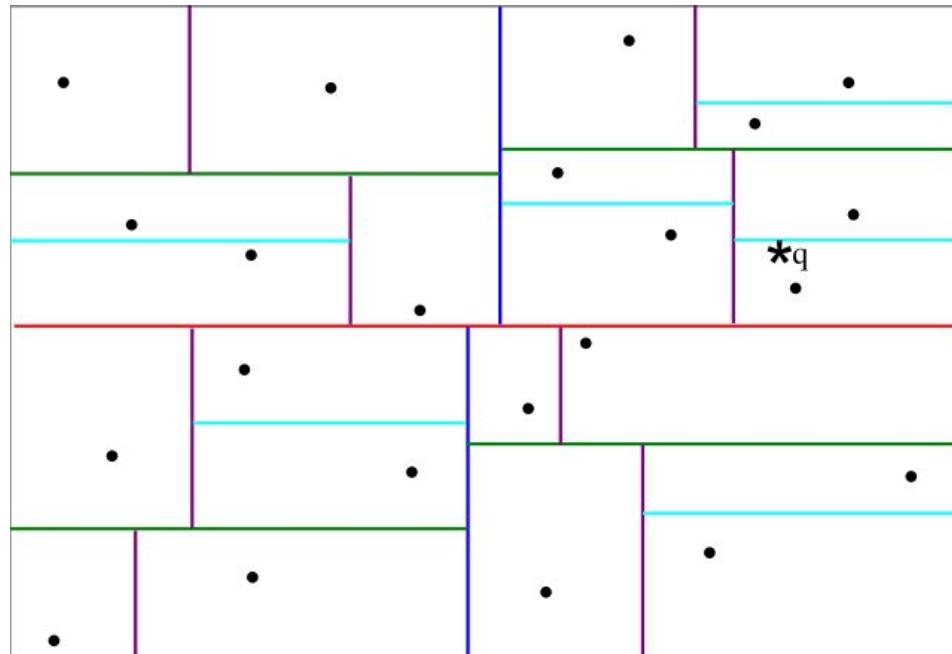


Разминулись с ответом на втором разбиении

[Scalable Nearest Neighbor Algorithms for High Dimensional Data](#)

FLANN

- 1) Для того чтобы сохранить ускорение, приходится жертвовать точностью
- 2) Посещаем не все листья а фиксированное количество
- 3) Чтобы уменьшить рычаг ошибки на верхних уровнях, строим несколько рандомизированных деревьев



Ответ попал в тот же лист дерева, ура!

[Scalable Nearest Neighbor Algorithms for High Dimensional Data](#)

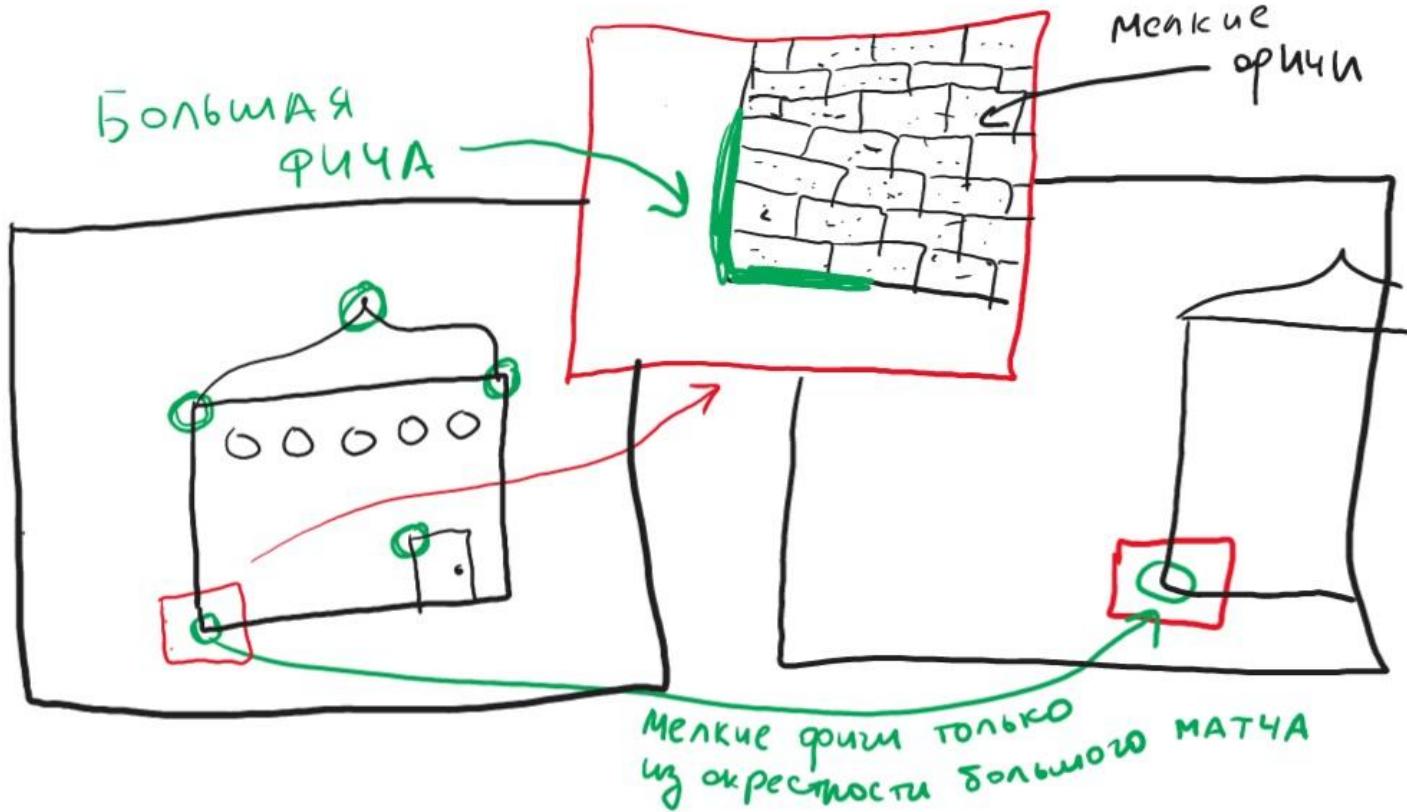
Guided matching

- 1) Пусть у нас есть очень большие снимки (~400Mpx) и мы используем Bruteforce matching и бинарные дескрипторы
- 2) С одной стороны, **хотим брать маленькие фичи**, чтобы получить максимум точности от большого разрешения камеры
- 3) С другой стороны, маленьких фичей на таком снимке очень много (миллионы). Если взять все, то **взорвется асимптотика $O(N^2)$**
- 4) ???

Guided matching

- 1) Пусть у нас есть очень большие снимки (~400Mpx) и мы используем Bruteforce matching и бинарные дескрипторы
- 2) С одной стороны, **хотим брать маленькие фичи**, чтобы получить максимум точности от большого разрешения камеры
- 3) С другой стороны, маленьких фичей на таком снимке очень много (миллионы). Если взять все, то **взорвется асимптотика $O(N^2)$**
- 4) ???
- 5) Большие фичи хорошо воспроизводятся (например, углов дома на картинке немного, в отличие от деталей отдельных кирпичиков). Давайте использовать **немного (2000) больших фичей** для подсказок, а сопоставления между маленькими фичами искать **только в небольшой окрестности**.

Guided matching



Сокращение количества пар изображений

Сложность brute-force матчинга $O(N^2 * M^2)$

N - число ключевых точек на кадр, M - число кадров

Как можно ускорить?

Ускорение вклада M^2 :

- Асимптотическое: использовать порядок фотографий / GPS чтобы искать соответствия только между близкими в пространстве изображениями. Тогда $O(M^2) \rightarrow O(M) / O(M \log(M))$
- Неасимптотическое: с каждого изображения взять немного самых больших (= самых повторяемых) ключевых точек, полностью поматчить, выбрать только самые обнадеживающие пары кадров для матчинга с полным $N = 2000$

Преселекция по ближайшим кадрам



Преселекция по ближайшим кадрам

- Если известно, что снимки делались последовательно (или известно GPS позиционирование), то можно сопоставлять только соседние кадры



Преселекция по ближайшим кадрам

- Если известно, что снимки делались последовательно (или известно GPS позиционирование), то можно сопоставлять только соседние кадры



$$O(M^2N^2) \rightarrow O(MN^2)$$

img3



img0



img1



img2



img4



img5



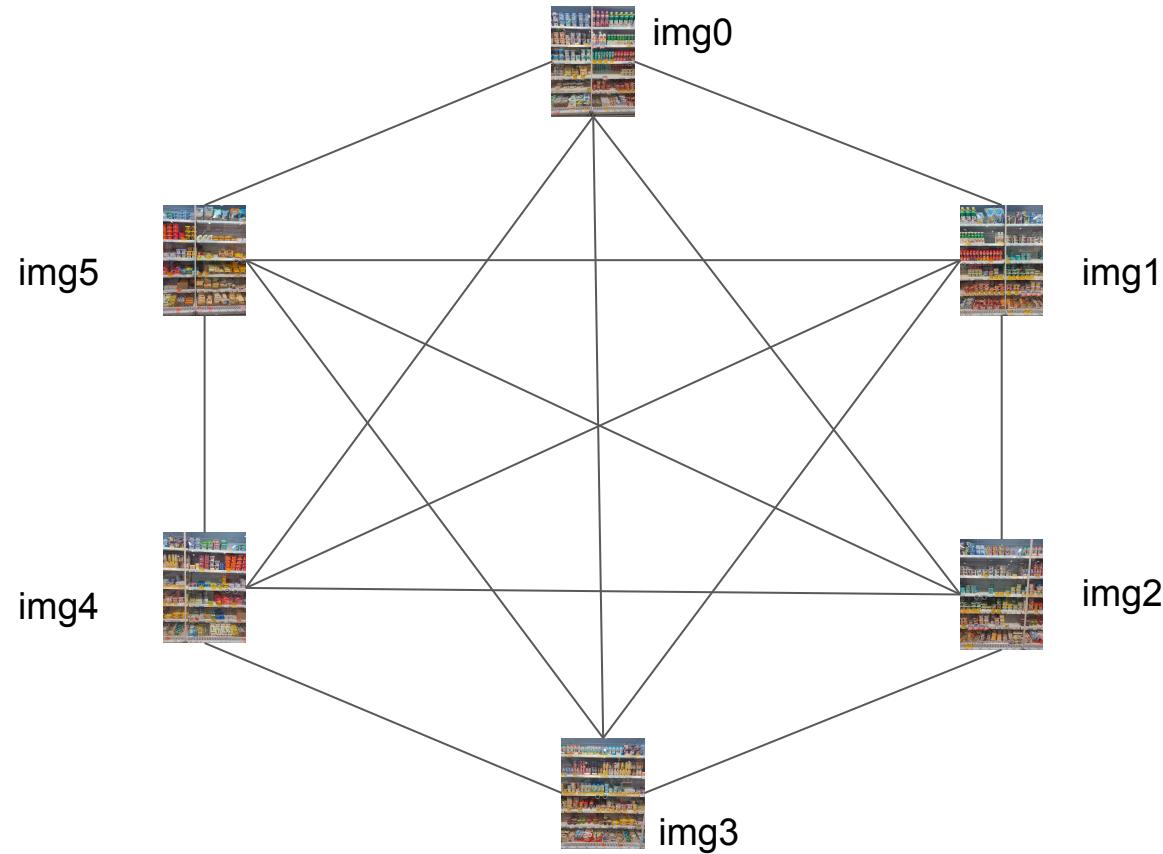
img6



img7

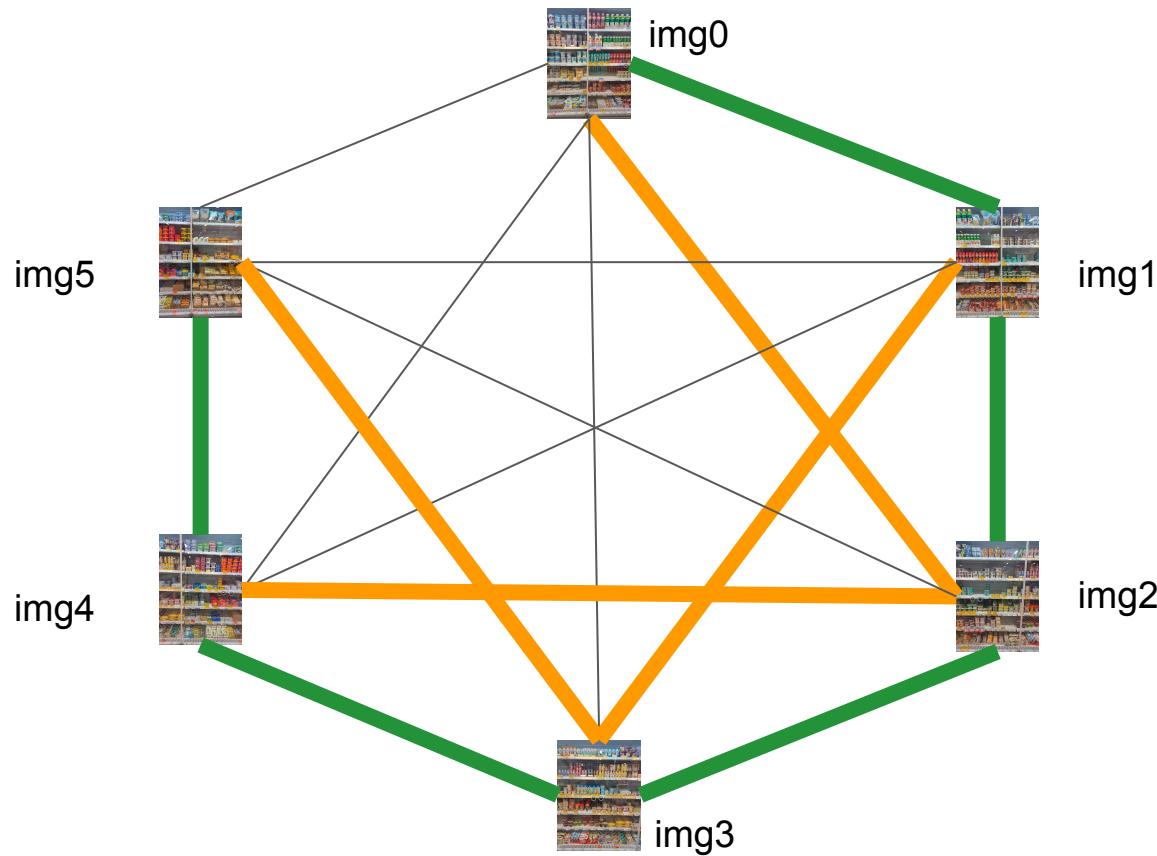
Преселекция по грубым матчам

- $O(M^2)$ пар матчинга с маленьким количеством крупных фичей ($N_{\text{strong}} = 100$)



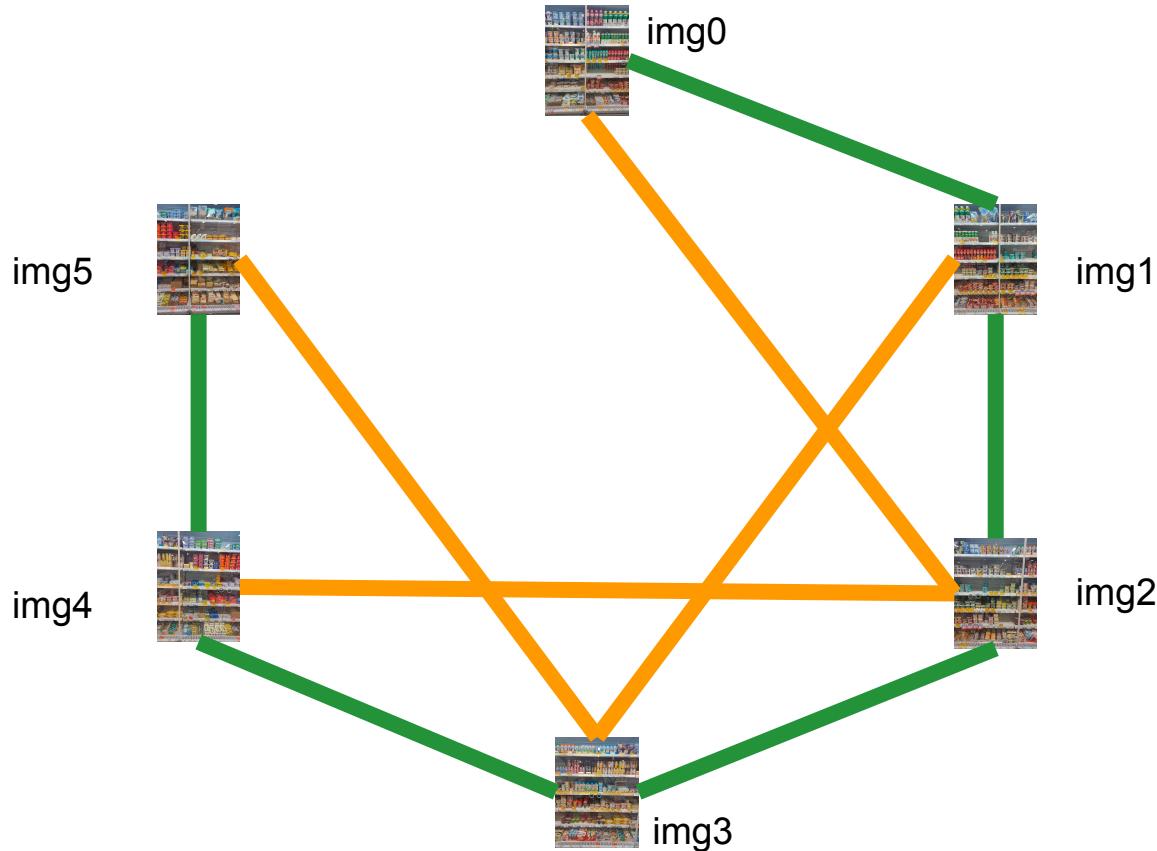
Преселекция по грубым матчам

- $O(M^2)$ пар матчинга с маленьким количеством крупных фичей ($N_{\text{strong}} = 100$)
- Найдем два остовных леса максимальных по количеству матчей



Преселекция по грубым матчам

- $O(M^2)$ пар матчинга с маленьким количеством крупных фичей ($N_{\text{strong}} = 100$)
- Найдем два остовных леса максимальных по количеству матчей
- $O(M)$ пар матчинга с полным количеством фичей ($N=2000$)
- Суммарная сложность (только матчинг):
 $O(M^2N_{\text{strong}}^2) + O(MN^2)$



Фильтрация сопоставлений

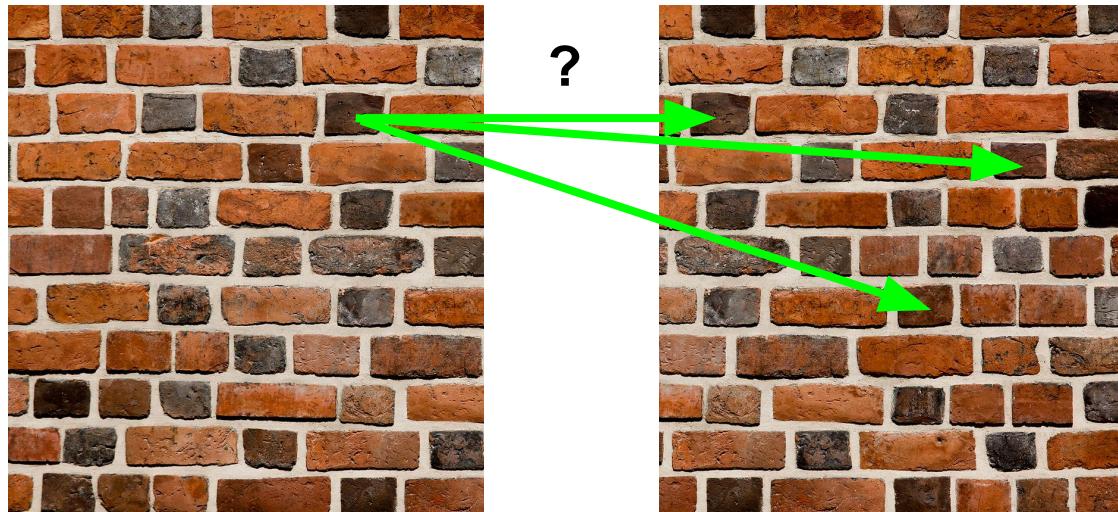
- 1) Научились быстро сопоставлять похожие ключевые точки
- 2) Ложные сопоставления?

Фильтрация сопоставлений

- 1) Научились быстро сопоставлять похожие ключевые точки
- 2) Ложные сопоставления? Какой может быть пример когда их много?

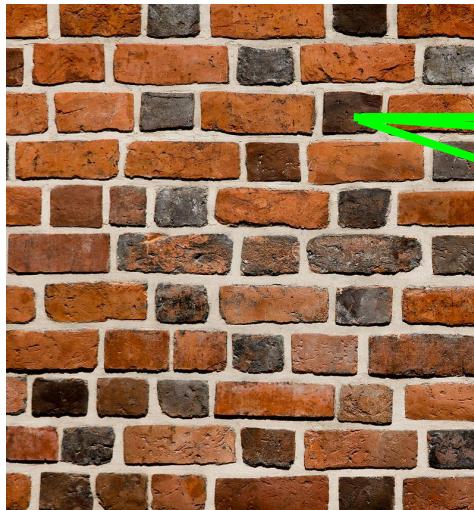
Фильтрация сопоставлений

- 1) Научились быстро сопоставлять похожие ключевые точки
- 2) Ложные сопоставления? Какой может быть пример когда их много?

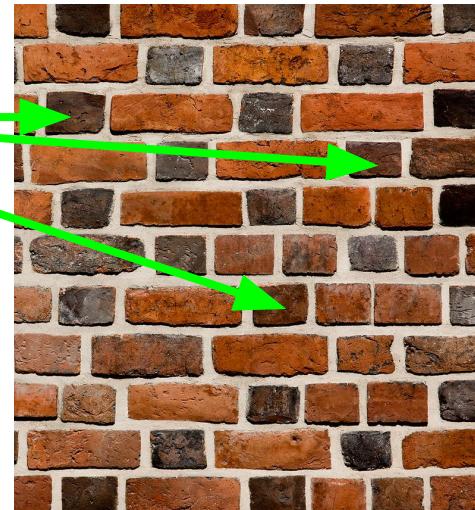


Фильтрация сопоставлений

- 1) Научились быстро сопоставлять похожие ключевые точки
- 2) Ложные сопоставления? Какой может быть пример когда их много?
- 3) Как отличаются первый и второй дескрипторы в хорошем и плохом случае?

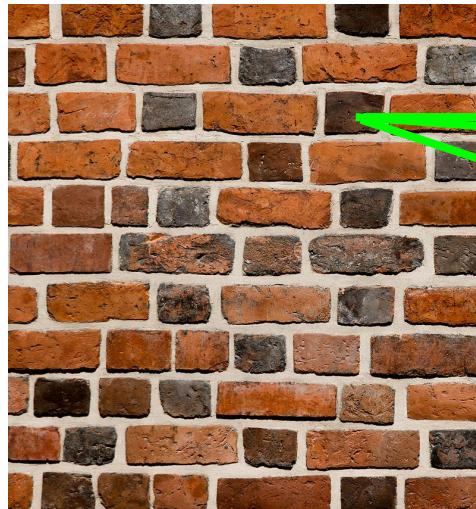


?

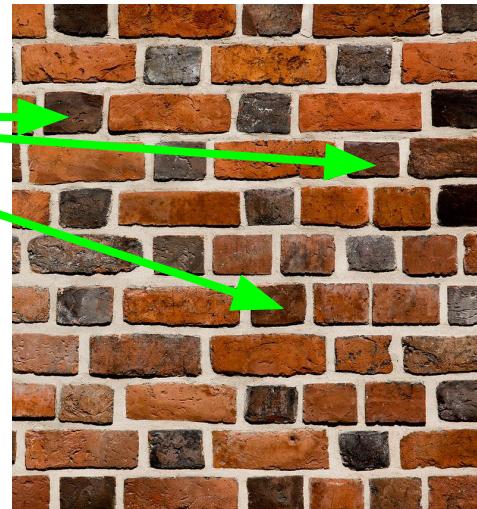


Фильтрация сопоставлений. Ratio-test

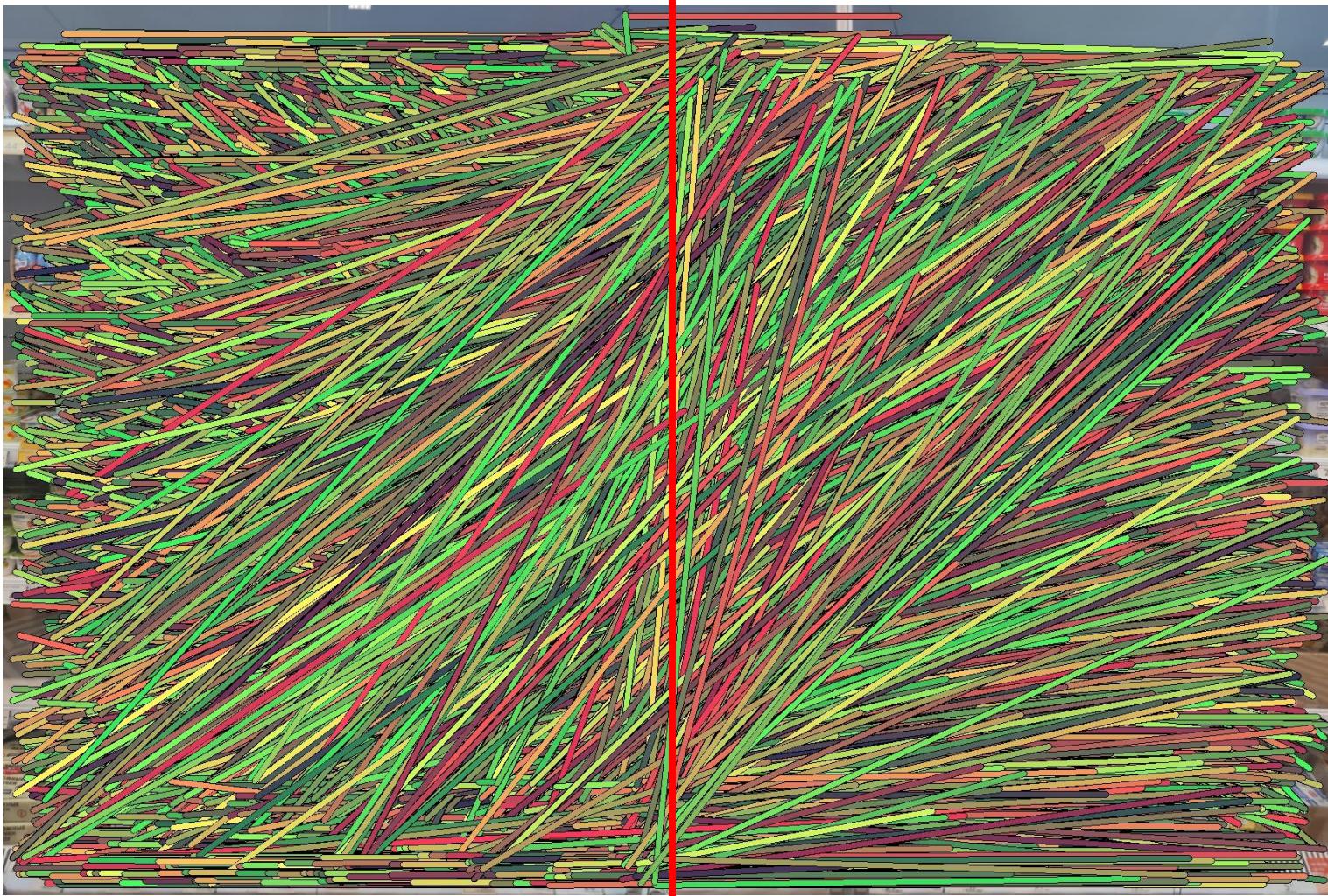
- 1) Ищем не только ближайшего соседа ($dist_0$) но и второго ($dist_1$) по близости
- 2) Если значение метрики отличается слабо ($dist_0 / dist_1 > 0.8$), то отбрасываем такое сопоставление



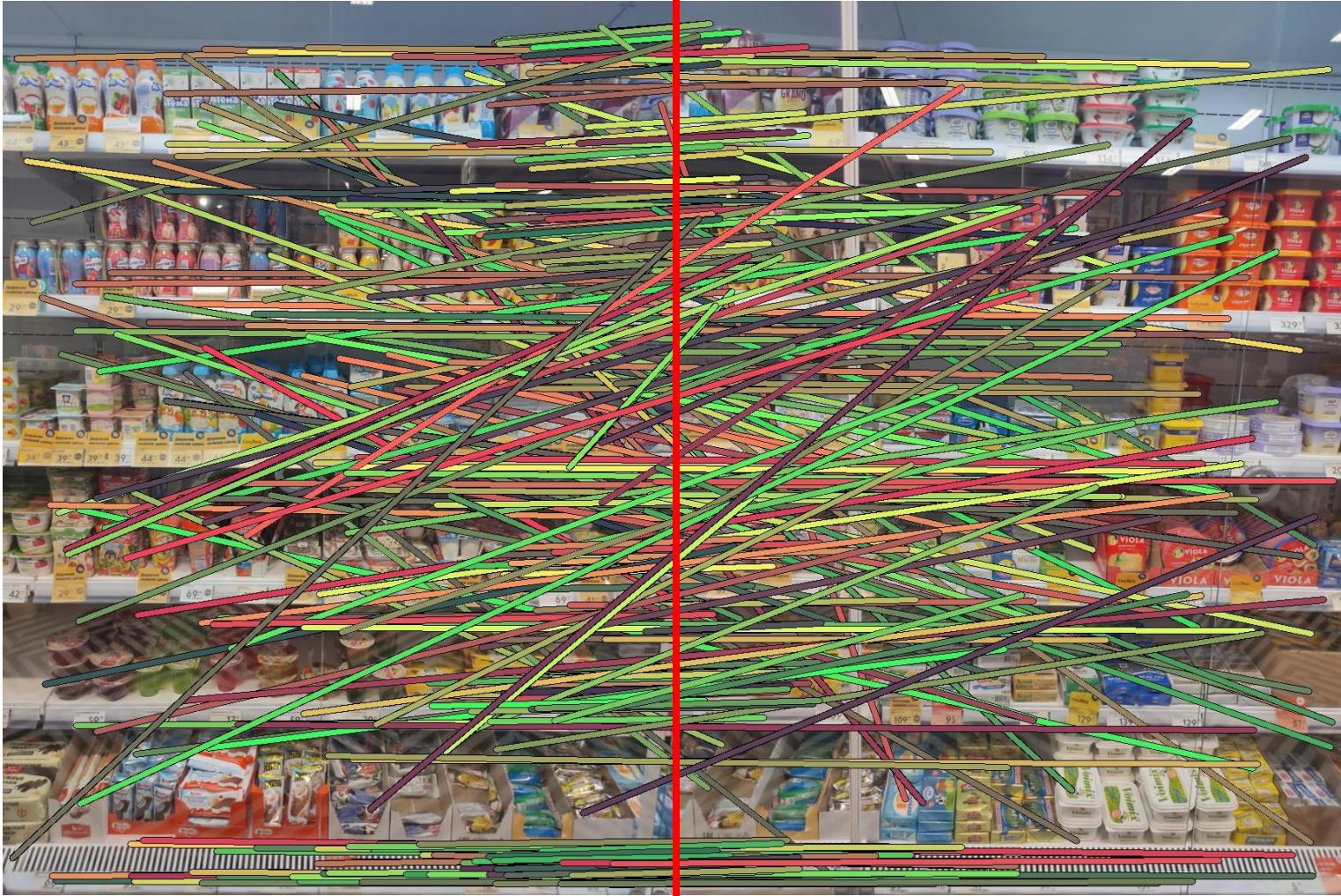
?



1. Nearest Neighbors



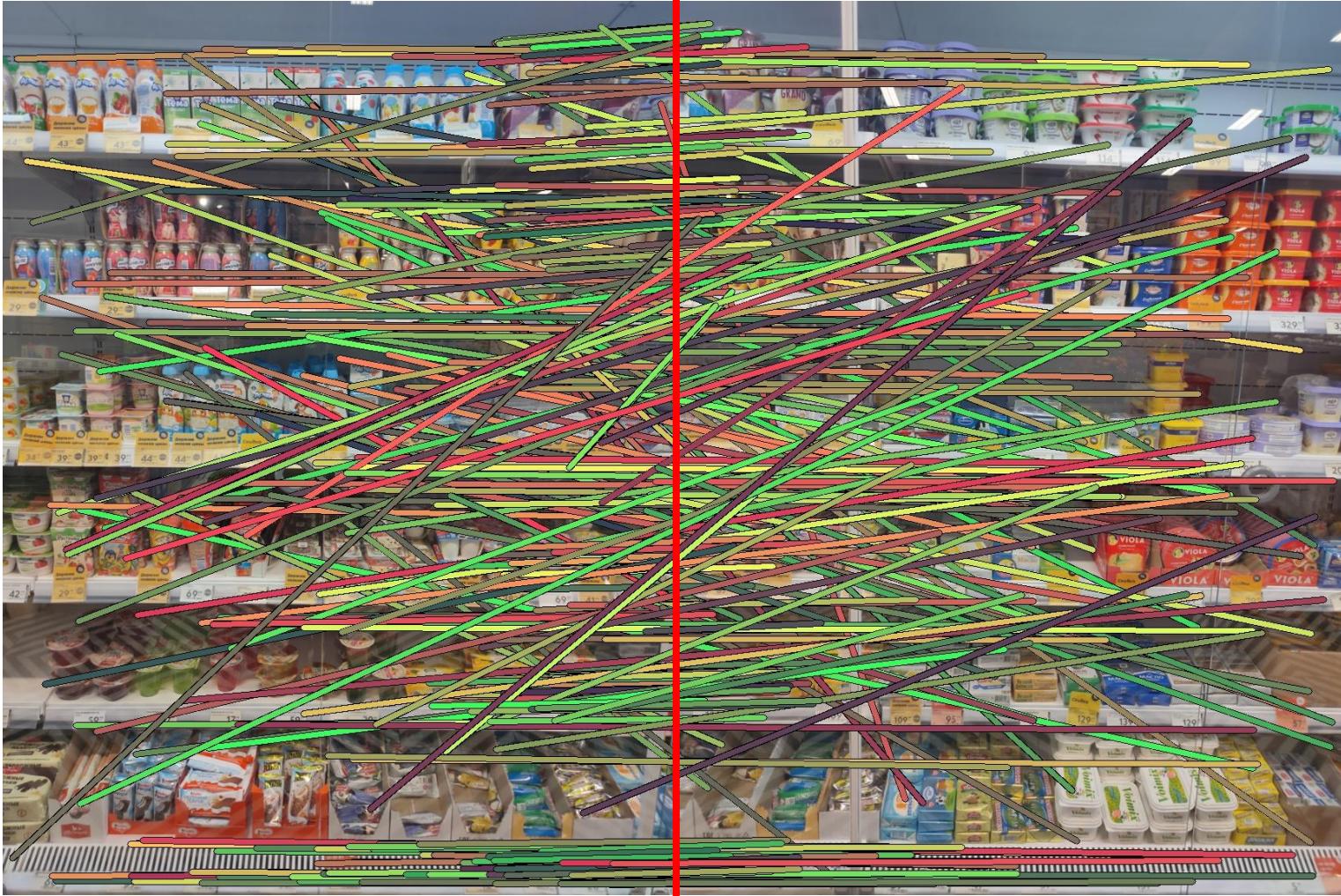
2. K-ratio test



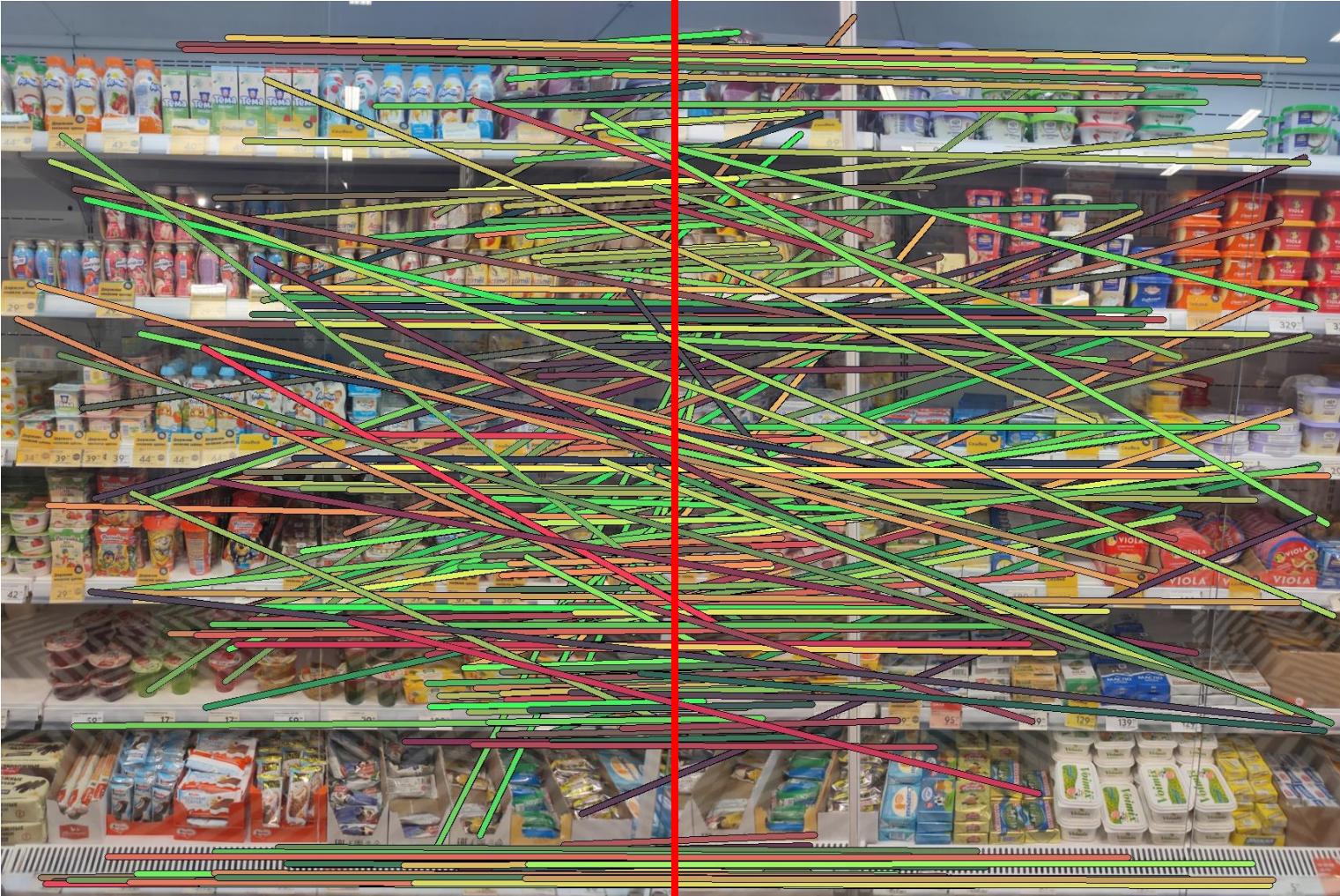
Фильтрация сопоставлений

- 1) Кроме дубликатов есть и просто шумы, которые достаточно уникальны, но не являются хорошими матчами

2. K-ratio test



2. K-ratio test (right-to-left)

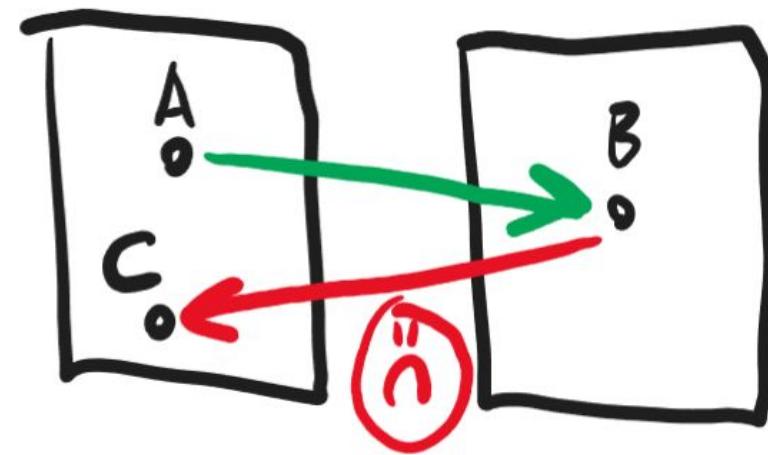
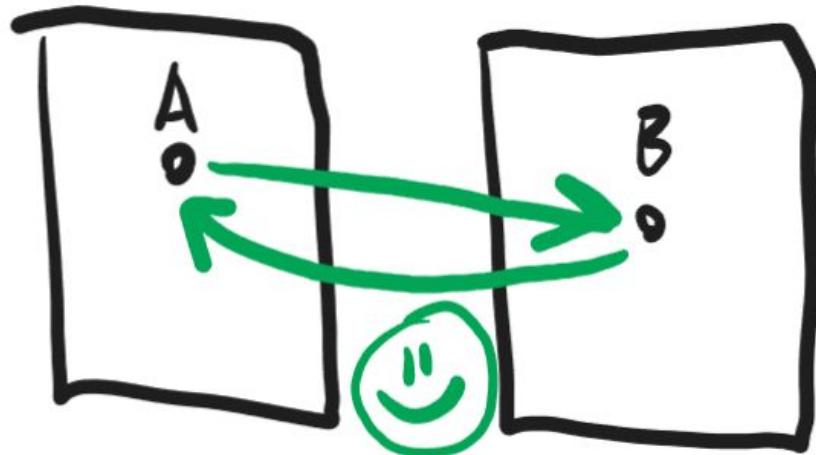


Фильтрация сопоставлений

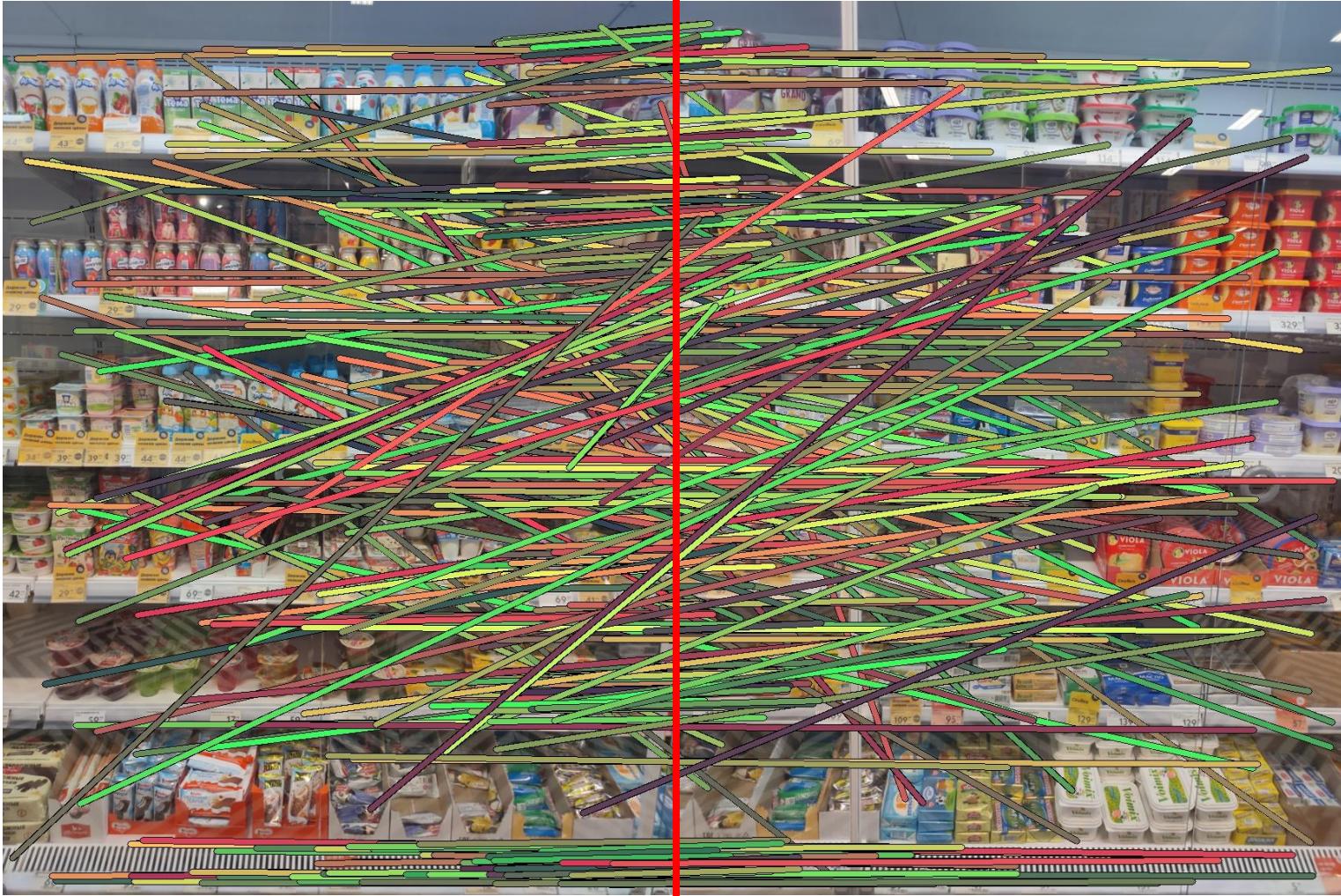
- 1) Кроме дубликатов есть и просто шумы, которые достаточно уникальны, но не являются хорошими матчами
- 2) Можно ли как-то использовать информацию о том, какие сопоставления нашлись при поиске ближайших матчей со второй картинки на первую?

Фильтрация сопоставлений. Left-right check

- 1) Кроме дубликатов есть и просто шумы, которые достаточно уникальны, но не являются хорошими матчами
- 2) Можно ли как-то использовать информацию о том, какие сопоставления нашлись при поиске ближайших матчей со второй картинки на первую?
- 3) Матчим справа-налево и слева-направо, оставляем только согласующиеся



2. K-ratio test

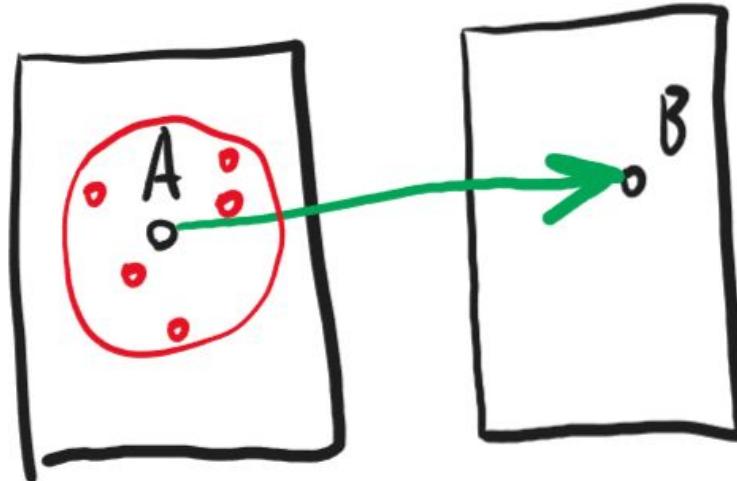


3. Left-right check



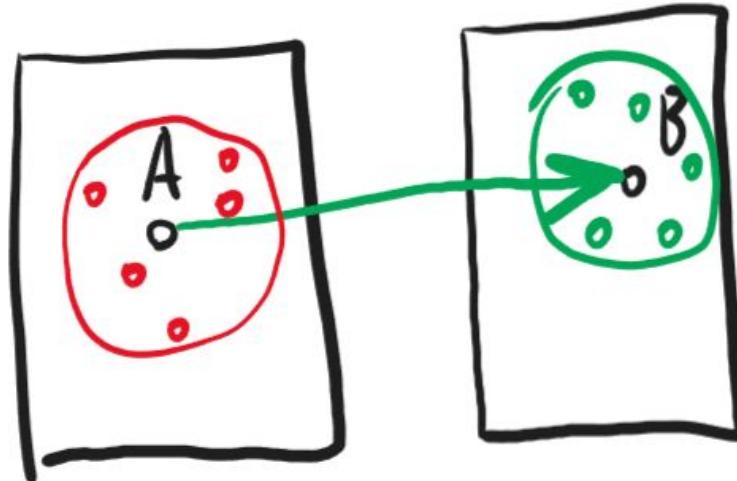
Фильтрация сопоставлений

- 1) Кроме дубликатов есть и просто шумы, которые достаточно уникальны, но не являются хорошими матчами
- 2) Как должна перейти кучка ключевых точек с первой картинки на вторую?



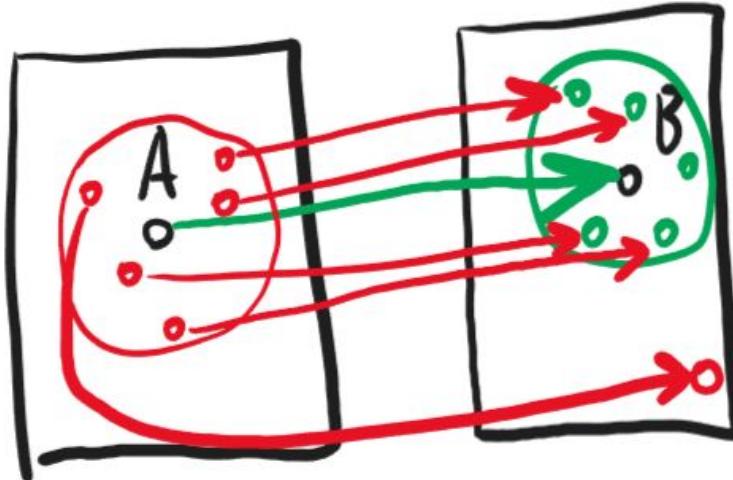
Фильтрация сопоставлений

- 1) Кроме дубликатов есть и просто шумы, которые достаточно уникальны, но не являются хорошими матчами
- 2) Как должна перейти кучка ключевых точек с первой картинки на вторую?



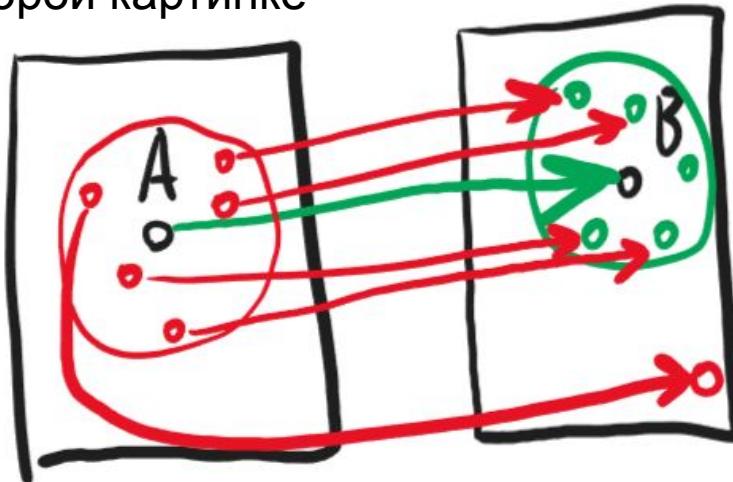
Фильтрация сопоставлений

- 1) Кроме дубликатов есть и просто шумы, которые достаточно уникальны, но не являются хорошими матчами
- 2) Как должна перейти кучка ключевых точек с первой картинки на вторую?



Фильтрация сопоставлений. Cluster filtering

- 1) Кроме дубликатов есть и просто шумы, которые достаточно уникальны, но не являются хорошими матчами
- 2) Как должна перейти кучка ключевых точек с первой картинки на вторую?
- 3) Из нашей **k**-окрестности **больше половины** точек должны перейти в **k**-окрестность на второй картинке



3. Left-right check



4. Cluster filtering

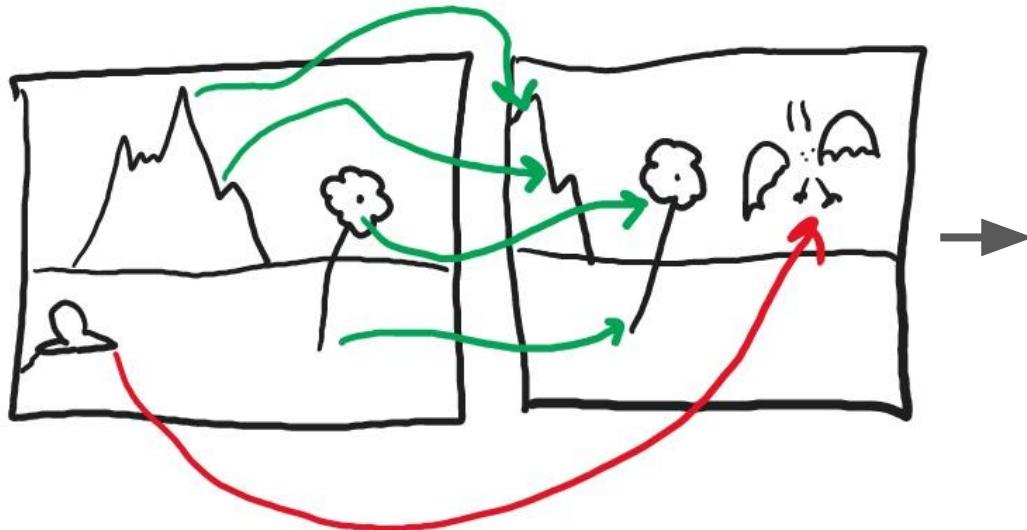


Фильтрация сопоставлений

- 1) Попробуем как-нибудь использовать модель задачи которую решаем?

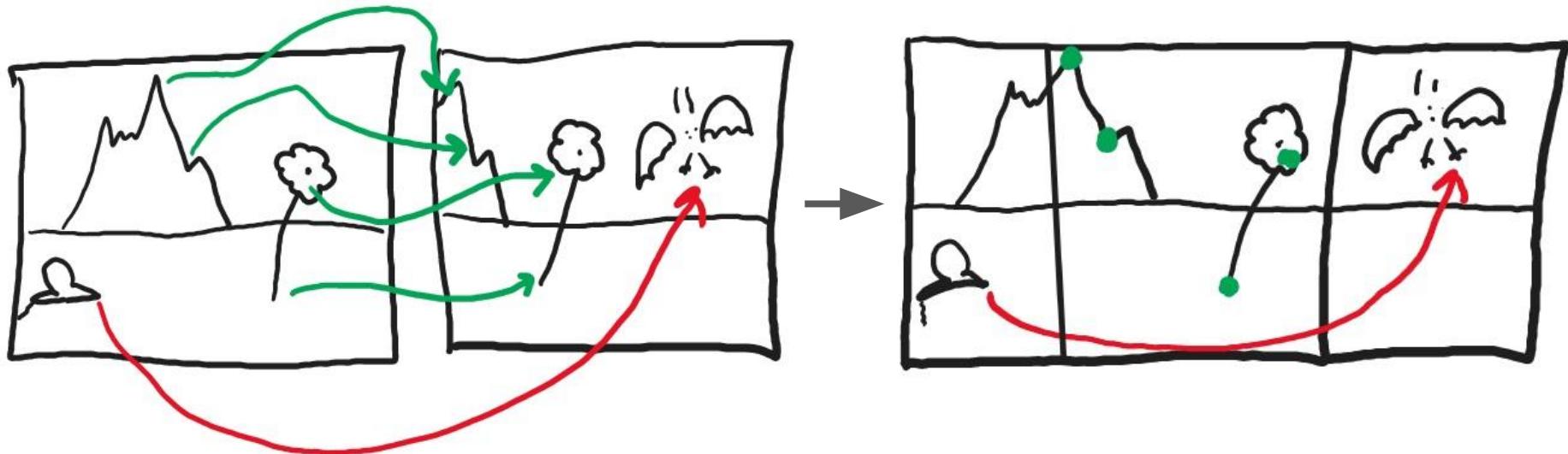
Фильтрация сопоставлений

- 1) Попробуем как-нибудь использовать модель задачи которую решаем?
- 2) Пусть мы знаем, как одна картинка накладывается на другую. Что произойдет с координатами плохих и хороших сматченных точек со второй картинки после сдвига?



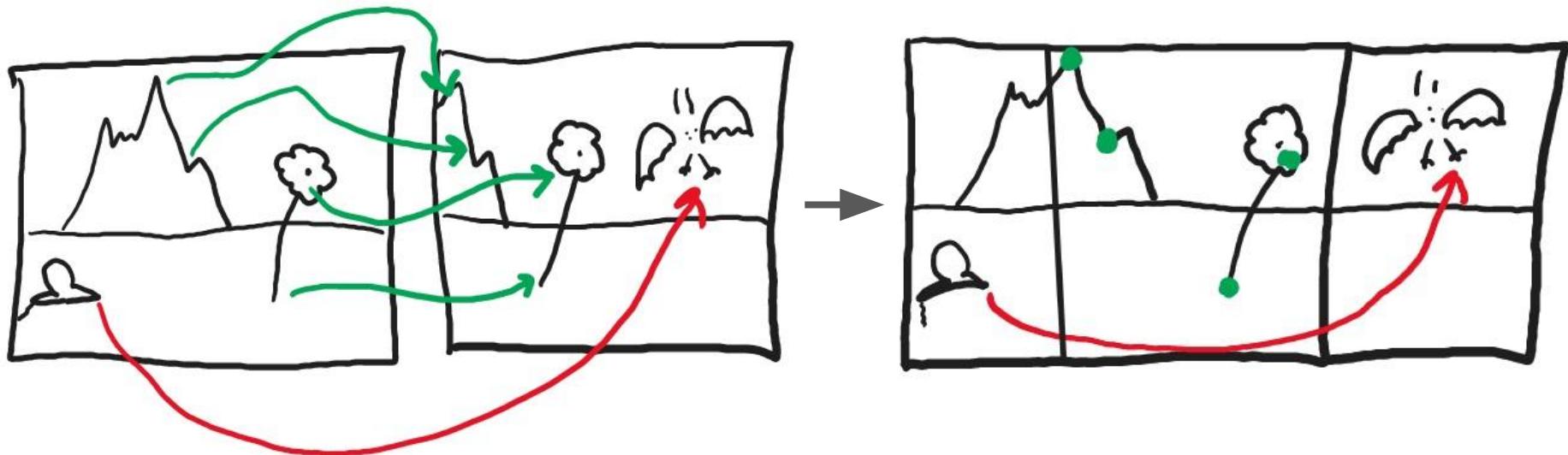
Фильтрация сопоставлений

- 1) Попробуем как-нибудь использовать модель задачи которую решаем?
- 2) Пусть мы знаем, как одна картинка накладывается на другую. Что произойдет с координатами плохих и хороших сматченных точек со второй картинки после сдвига?



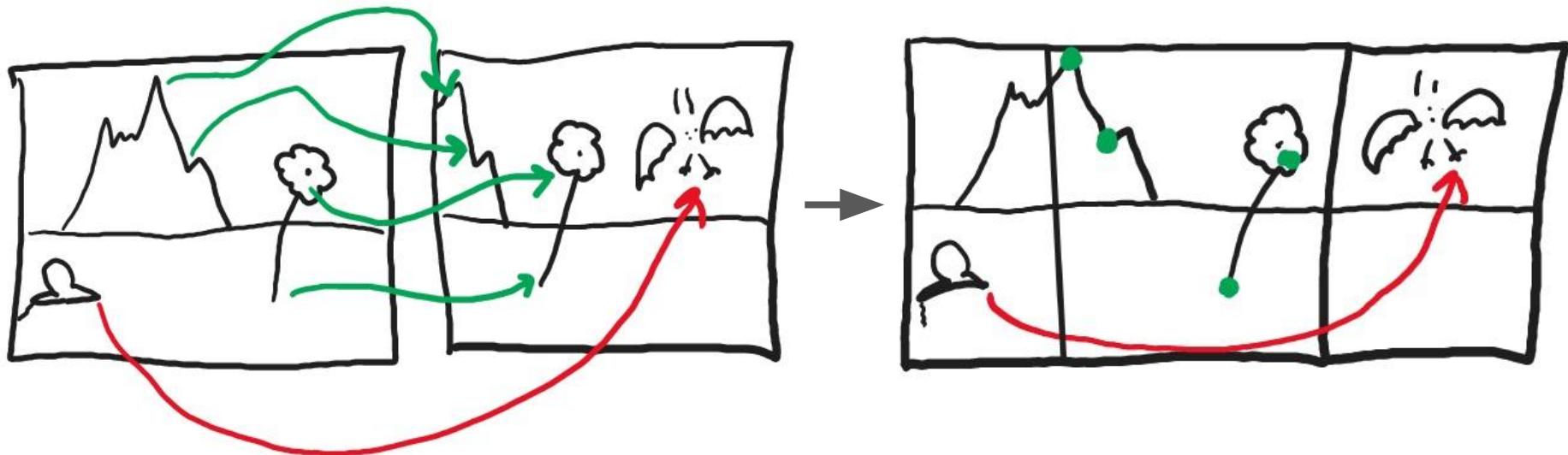
Фильтрация сопоставлений

- 1) Попробуем как-нибудь использовать модель задачи которую решаем?
- 2) Пусть мы знаем, как одна картинка накладывается на другую. Тогда применим к точкам со второй картинки преобразование, и выбросим те матчи, точки с которых не сойдутся



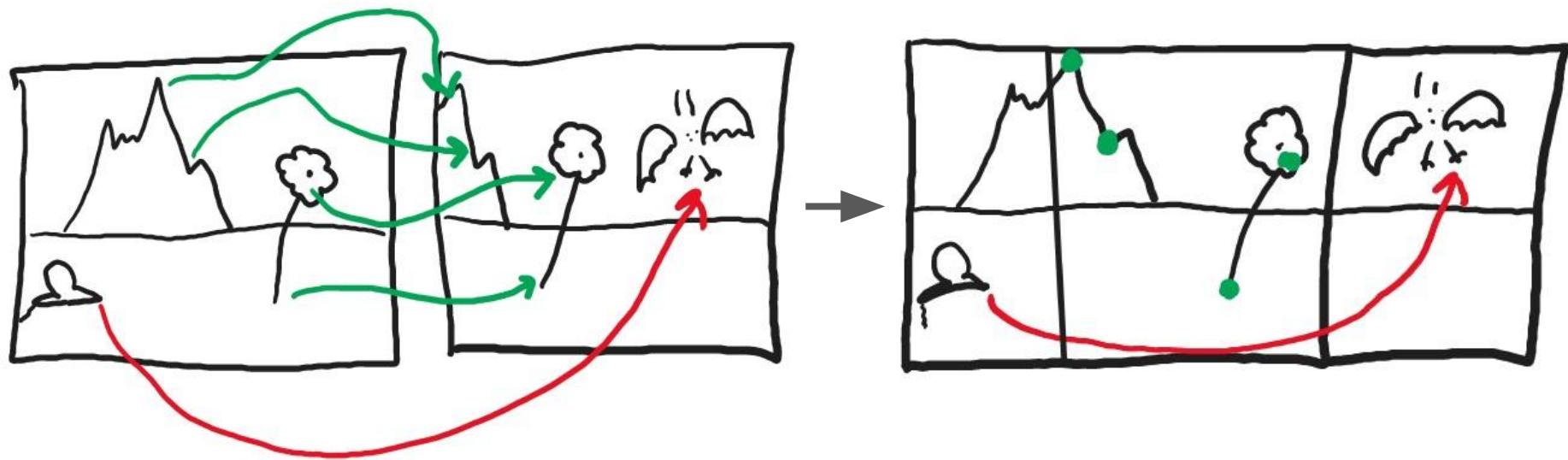
Фильтрация сопоставлений

- 1) Попробуем как-нибудь использовать модель задачи которую решаем?
- 2) **Пусть мы знаем**, как одна картинка накладывается на другую. Тогда применим к точкам со второй картинки преобразование, и выбросим те матчи, точки с которых не сойдутся



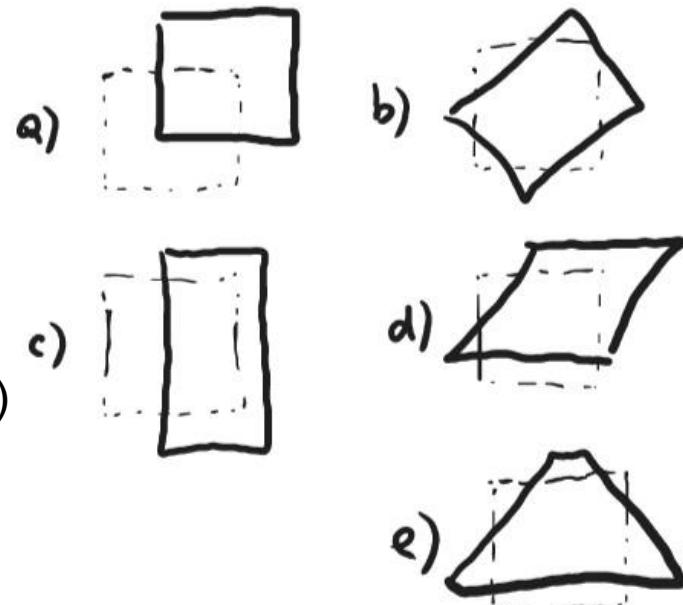
Фильтрация сопоставлений

Проблема курицы и яйца, откуда узнаем как передвинуть вторую картинку?



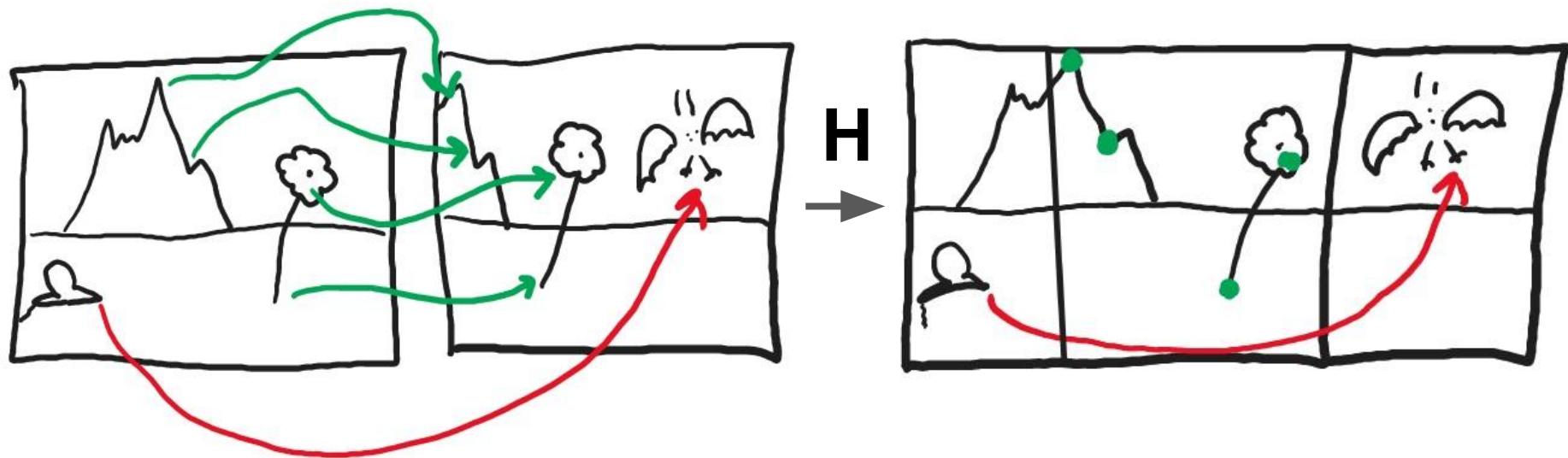
Справка: Гомография

- 1) Преобразование пикселей со второй картинки в первую можно описать с помощью особой матрицы \mathbf{H} , называемой гомографией
- 2) Гомография описывает:
 - a) Сдвиг
 - b) Поворот
 - c) Растяжение по осям X и Y
 - d) Shear
 - e) Перспективное преобразование
- 3) Переводит прямые линии в прямые
- 4) Можно рассчитать по 4 матчам (след. лекция)
- 5) Зная парные гомографии можно по цепочке склеить все картинки в одну панораму



Фильтрация сопоставлений

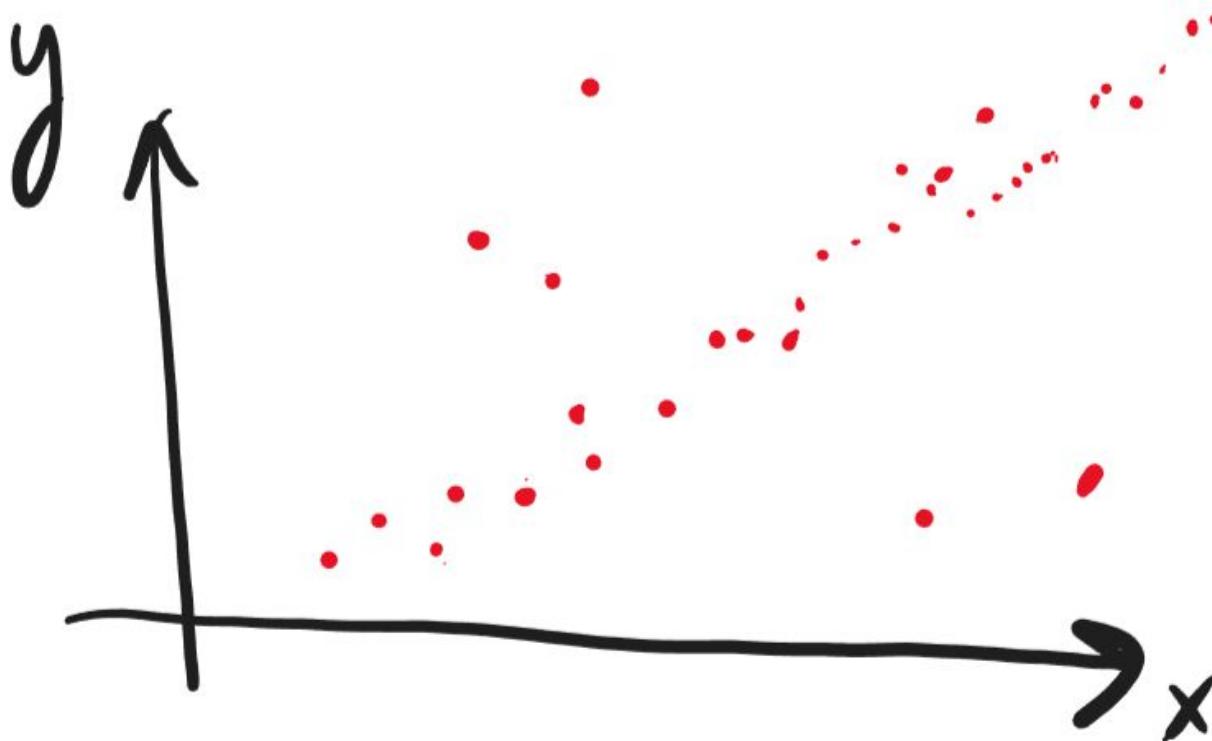
Проблема курицы и яйца, откуда узнаем гомографию?



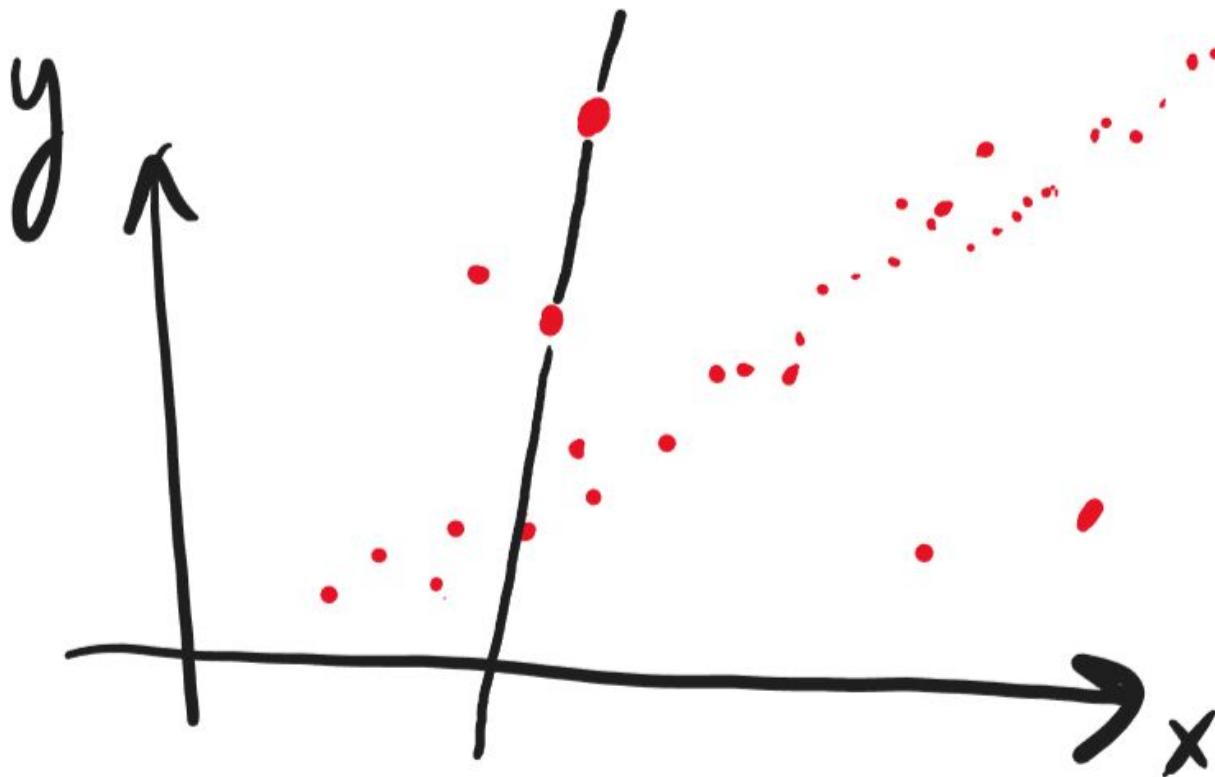
RANSAC. Реклама

- 1) Устойчивый к шуму и выбросам способ оценить какую-то (какую угодно) модель по набору измерений, про которые не знаем, какие верные а какие - нет
- 2) Можно применять для фиттинга прямых, парабол, сложных нелинейных кривых, плоскостей, параболоидов, фотограмметрических матриц (гомографий, фундаментальных/вещественных матриц), поиска фигур в облаках точек и тд.....
- 3) Есть много вариаций, в том числе с автоматической оценкой порогов. см. MSAC, AC-RANSAC, PROSAC, MLSAC etc.
- 4) Можно накладывать ограничения, например ищем только плоскости без дырок
- 5) Хорошо параллелится, просто пишется
- 6) Сложно перехвалить...

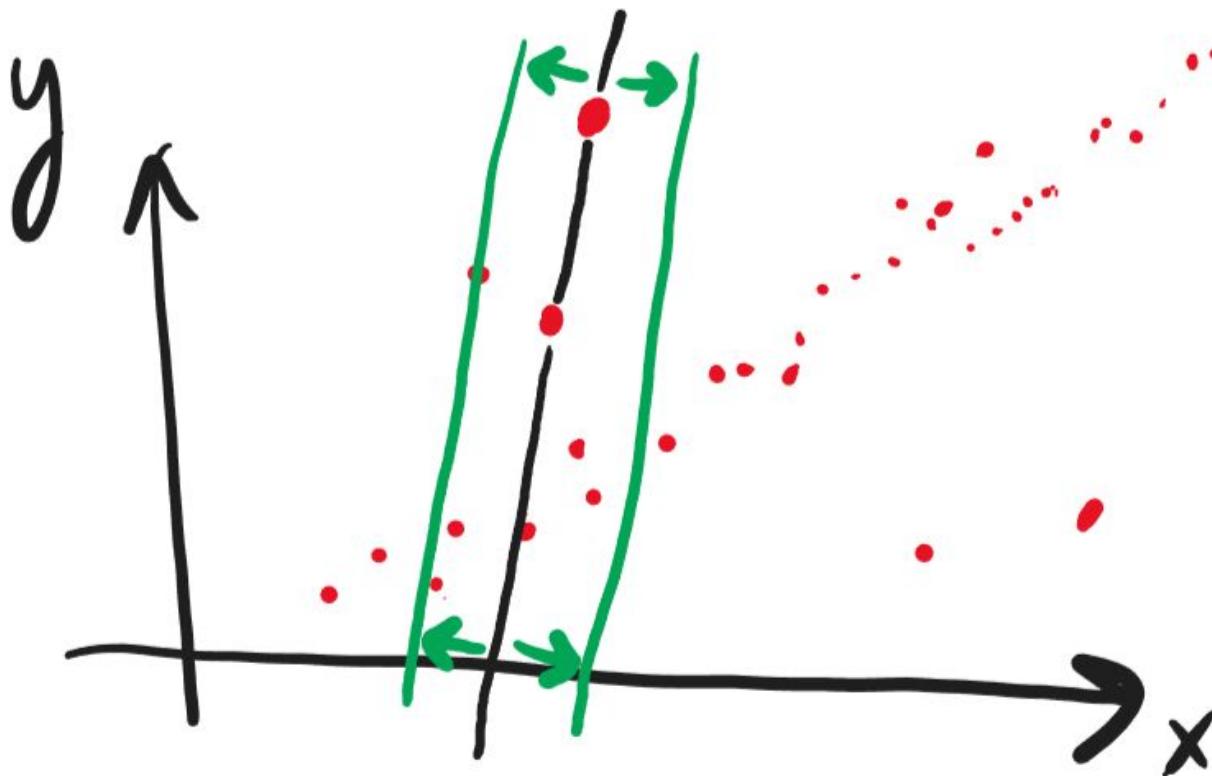
RANSAC для поиска прямой среди точек



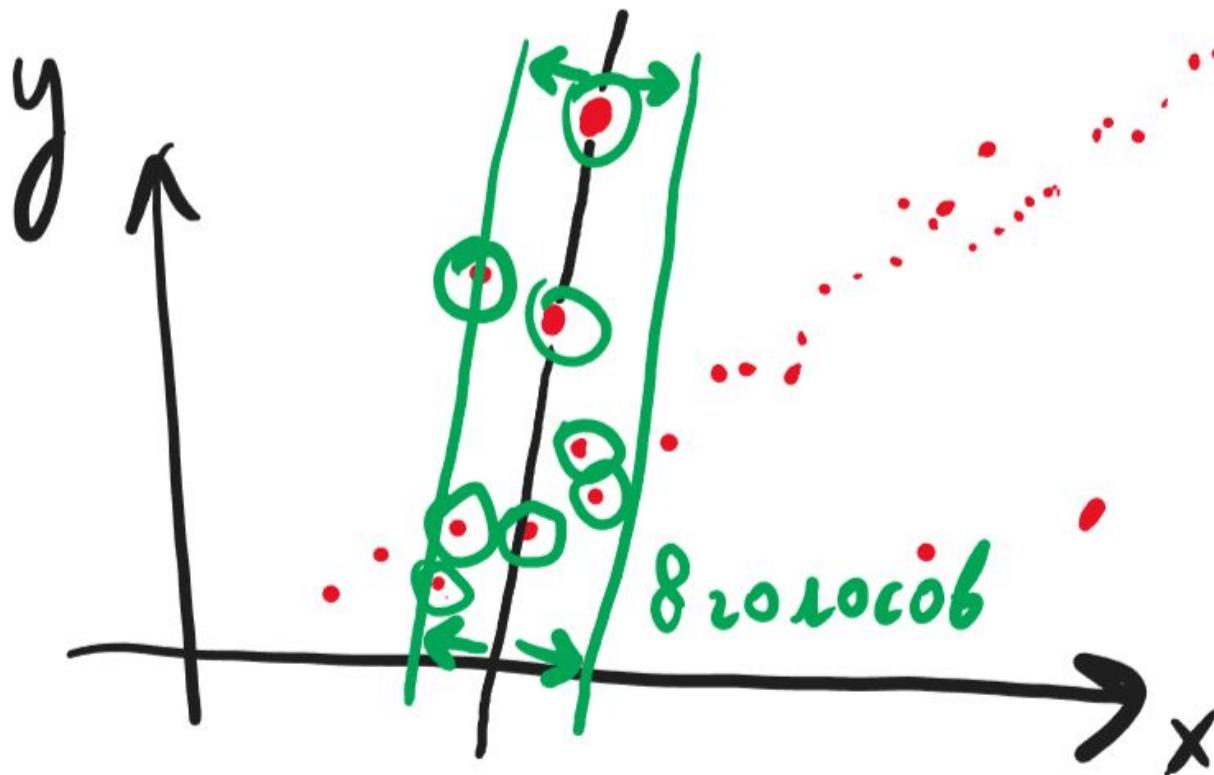
RANSAC для поиска прямой среди точек



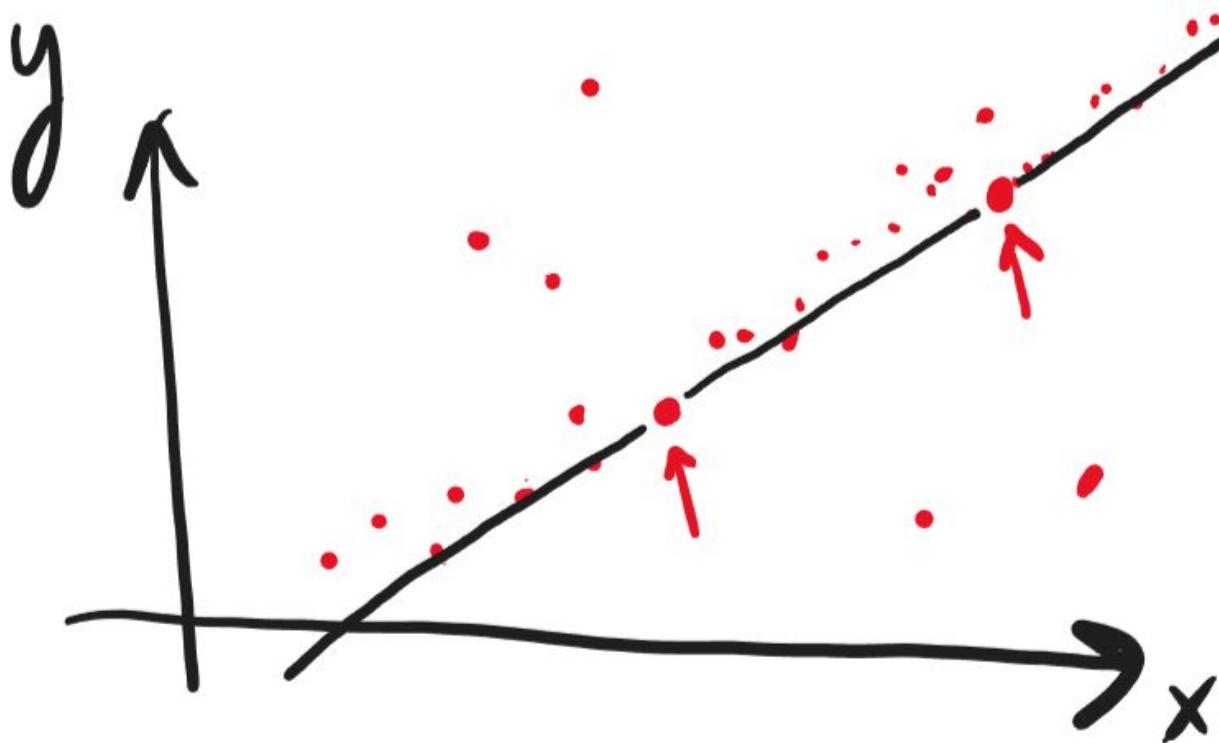
RANSAC для поиска прямой среди точек



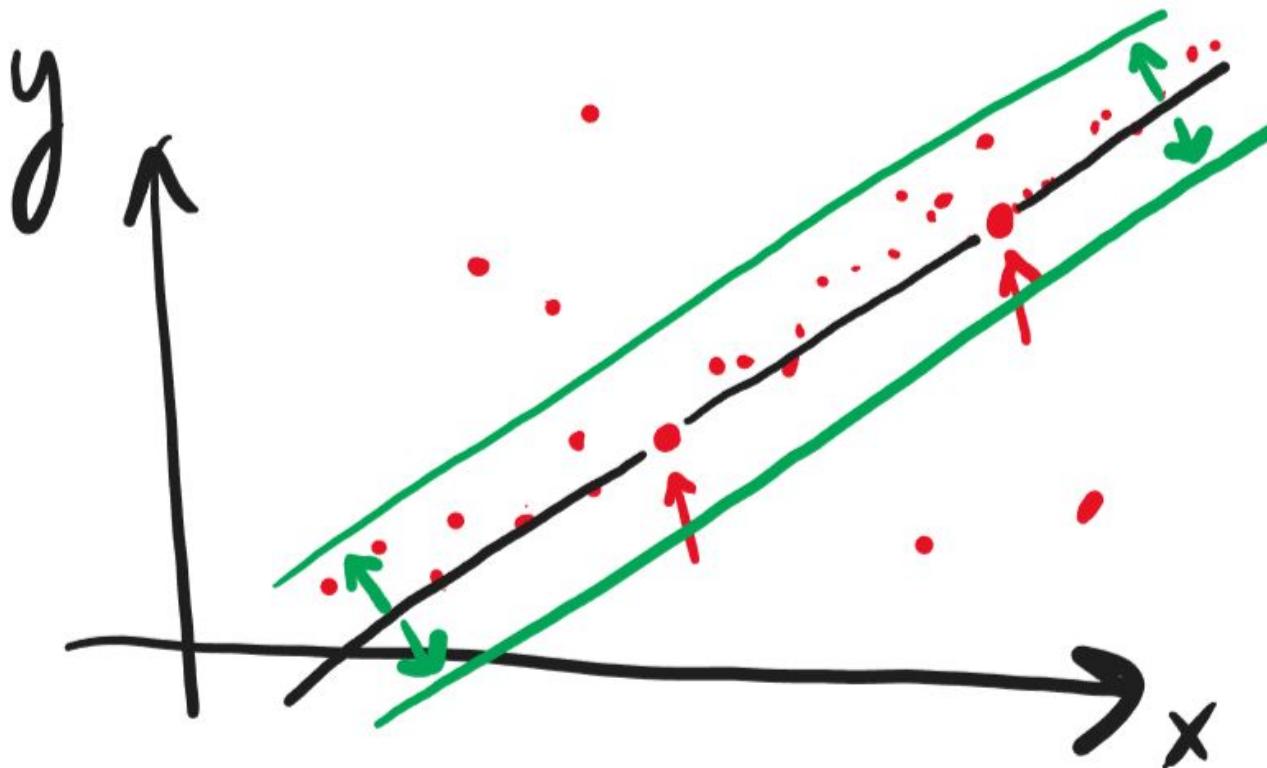
RANSAC для поиска прямой среди точек



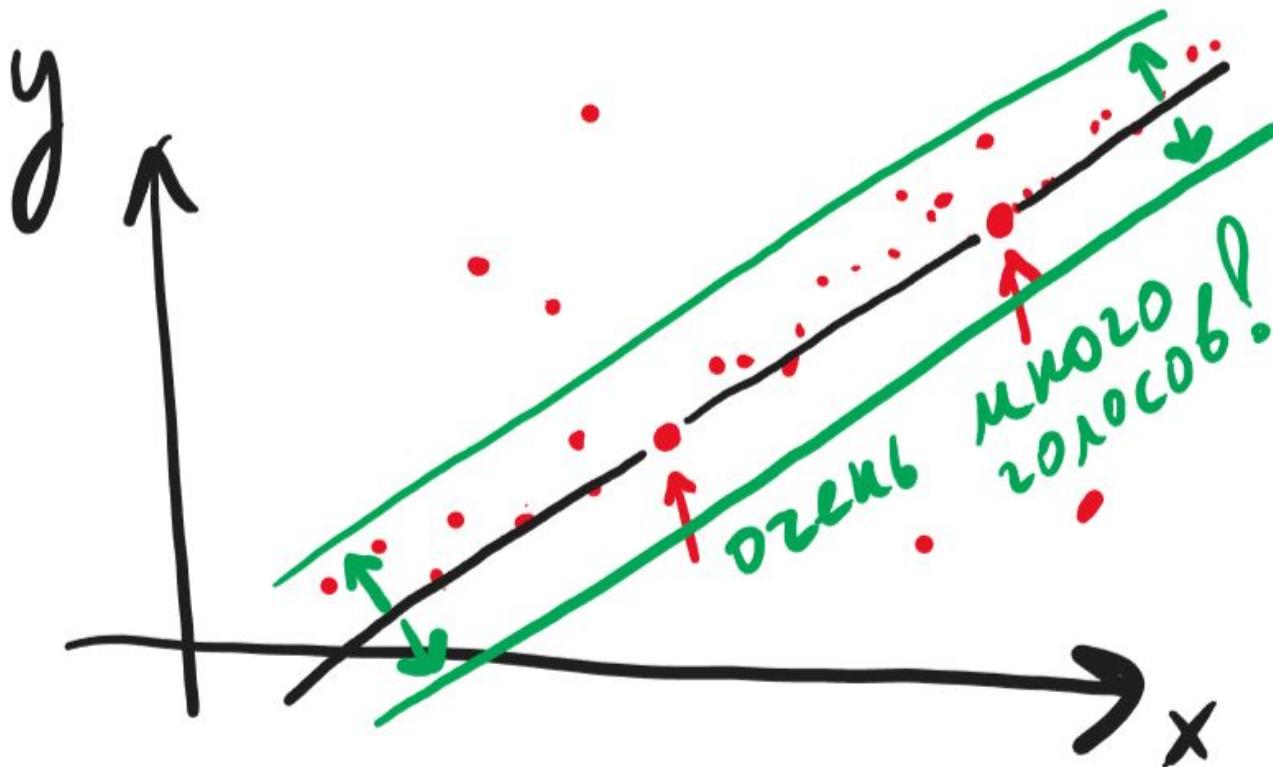
RANSAC для поиска прямой среди точек



RANSAC для поиска прямой среди точек



RANSAC для поиска прямой среди точек



RANSAC для поиска прямой среди точек

```
result_line = None
```

```
result_score = 0
```

RANSAC для поиска прямой среди точек

```
result_line = None
result_score = 0
// количество итераций можно высчитать на основе ожидаемого процента
// выбросов
1000 итераций делаем:
A, B = randomSamples(points)
```

RANSAC для поиска прямой среди точек

```
result_line = None
result_score = 0
// количество итераций можно высчитать на основе ожидаемого процента
// выбросов
1000 итераций делаем:
A, B = randomSamples(points)
line = fit(A, B)
```

RANSAC для поиска прямой среди точек

```
result_line = None
result_score = 0
// количество итераций можно высчитать на основе ожидаемого процента
// выбросов
1000 итераций делаем:
A, B = randomSamples(points)
line = fit(A, B)
score = estimateScore(points, line)
```

RANSAC для поиска прямой среди точек

```
result_line = None
result_score = 0
// количество итераций можно высчитать на основе ожидаемого процента
// выбросов
1000 итераций делаем:
A, B = randomSamples(points)
line = fit(A, B)
score = estimateScore(points, line)
if (score > result_score)
    result_line, result_score = line, score
```

RANSAC для поиска прямой среди точек

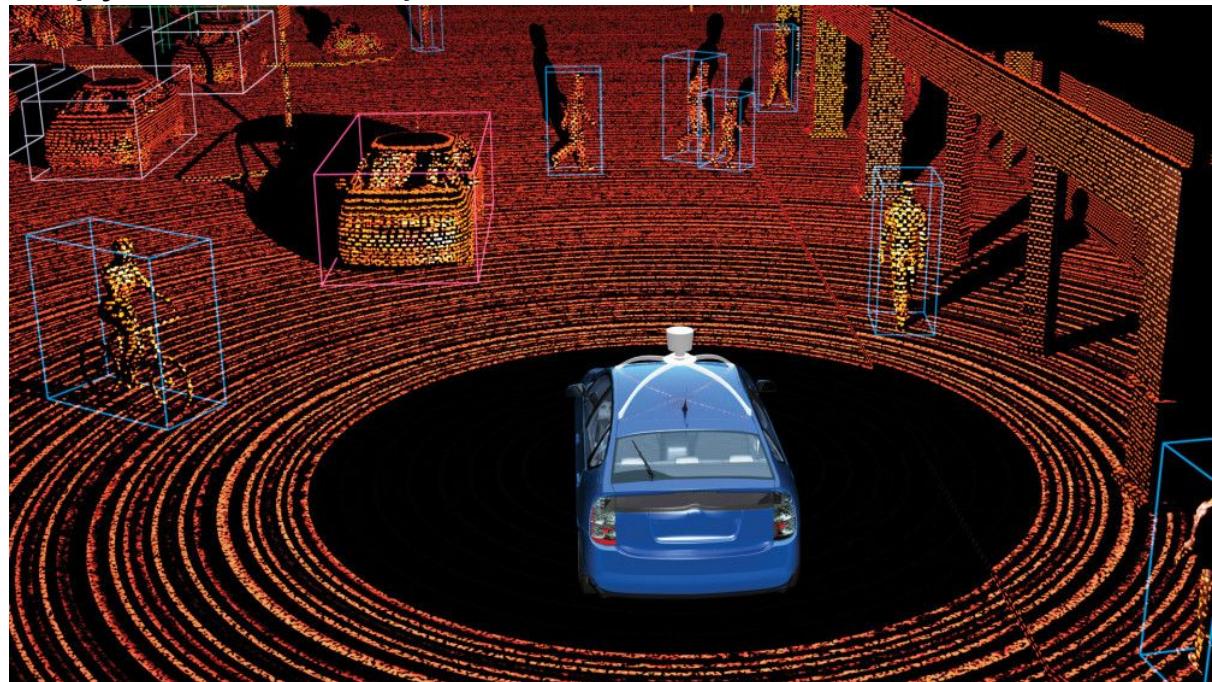
- 1) А что изменится если бы мы искали параболу?
(много точек лежащих около какой-то одной параболы + много выбросов)

RANSAC для поиска прямой среди точек

- 1) А что изменится если бы мы искали параболу?
(много точек лежащих около какой-то одной параболы + много выбросов)
- 2) А как решить задачу поиска двух парабол?
(40% точек на одной параболе, 40% точек на другой параболе
и 20% точек - выбросы)

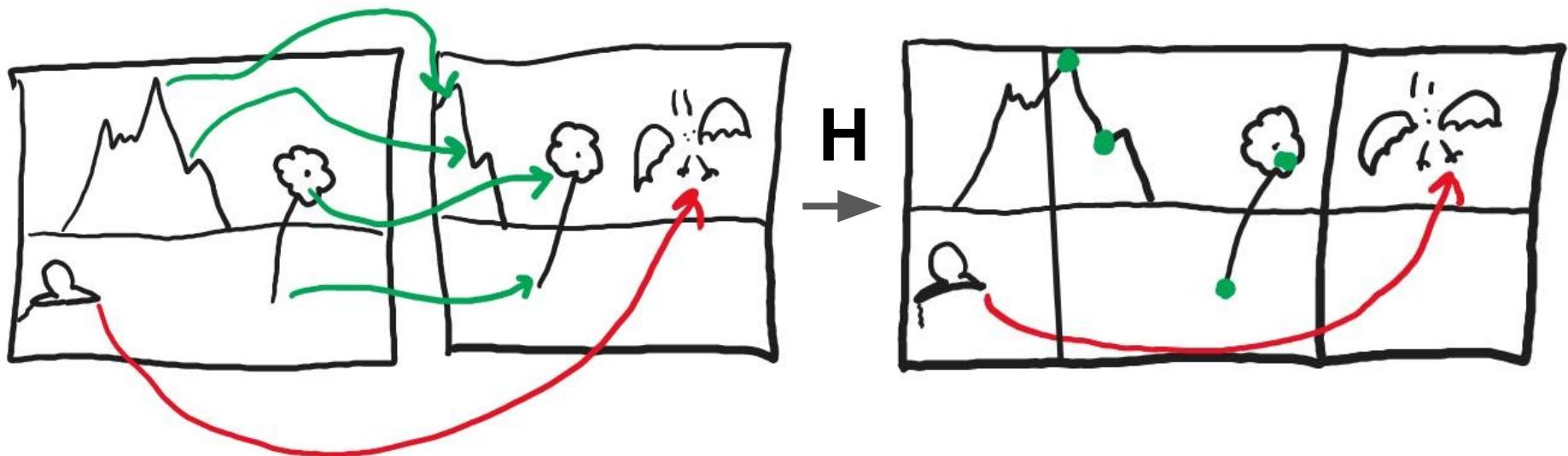
RANSAC для поиска плоскости

3) А что если точки в 3D и мы хотим найти плоскость на которой они лежат?
Например это LIDAR-скан окружающего мира с беспилотного автомобиля и мы
хотим найти где земля:



Фильтрация сопоставлений. RANSAC

- 1) На каждой итерации RANSAC по 4 матчам определяем гомографию и отфильтровываем выбросы
- 2) Оставляем матчи с итерации, где фильтрацию прошло наибольшее их количество



4. Cluster filtering

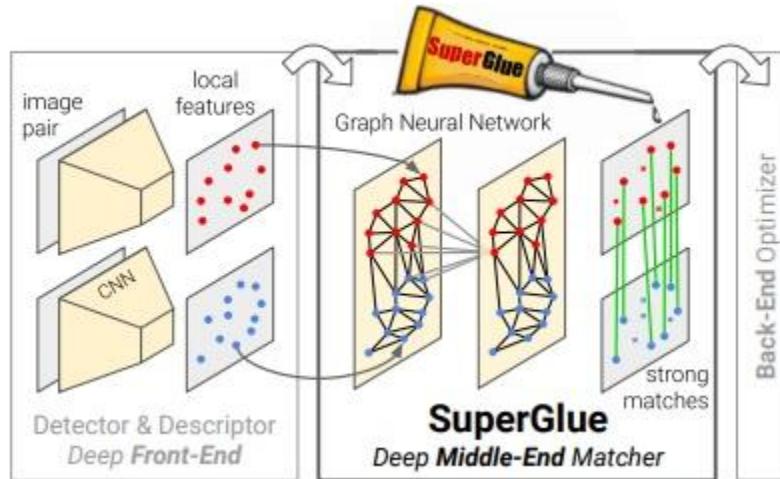


5. RANSAC



Ссылки

- 1) [GMS matcher](#) (похоже на cluster filtering)
- 2) [SuperGlue](#) (нейронка, расширение cluster filtering, следим за паттернами)
- 3) [Бенчмарк](#) современных детекторов и дескрипторов и сравнение с SIFT
- 4) [Ежегодный большой контест](#) по feature matching



Вопросы?



Симиутин Борис

simiyutin.boris@yandex.ru⁷⁴