

Алгоритм построения панорамы из фотографий

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

Источник картинки: С. Malin

Полярный Николай

polarnick239@gmail.com

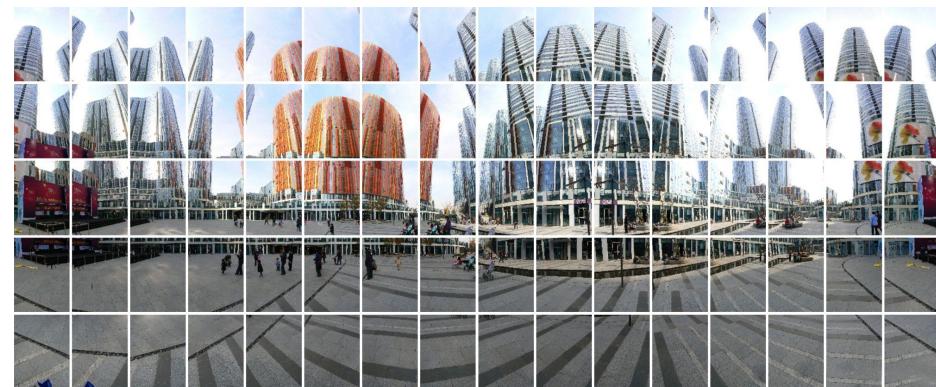
Постановка задачи

Входные данные:

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

Результат:

- одна большая фотография с этой же точки
но со **сферической** “камеры”

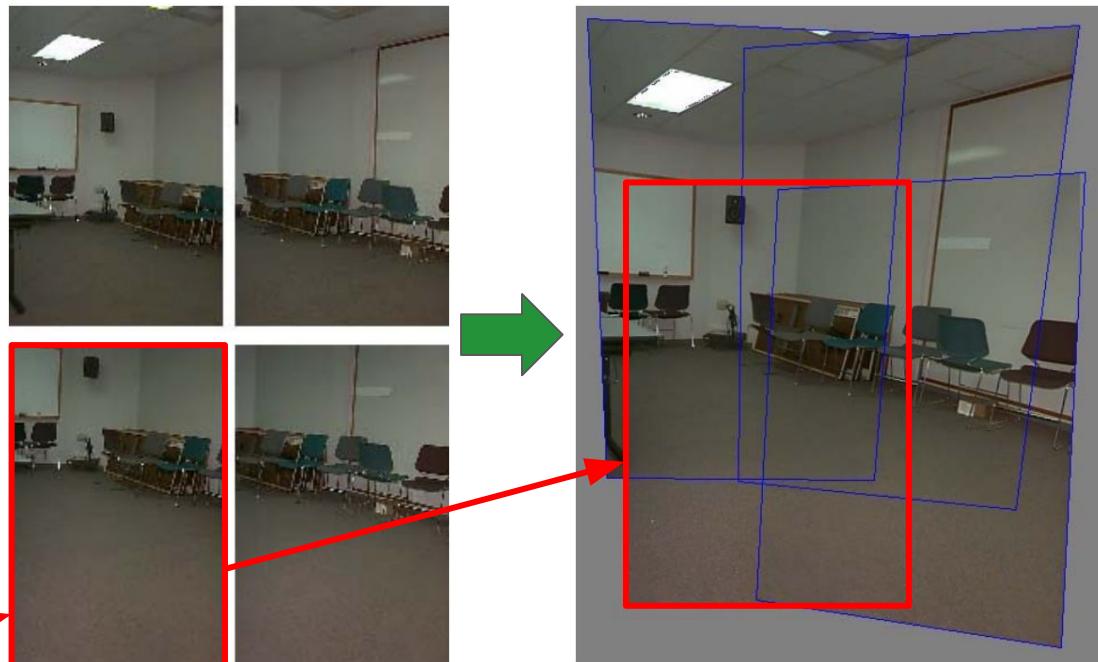


[Источник картинки](#)

Постановка задачи (упрощенная)

Входные данные:

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



Результат:

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

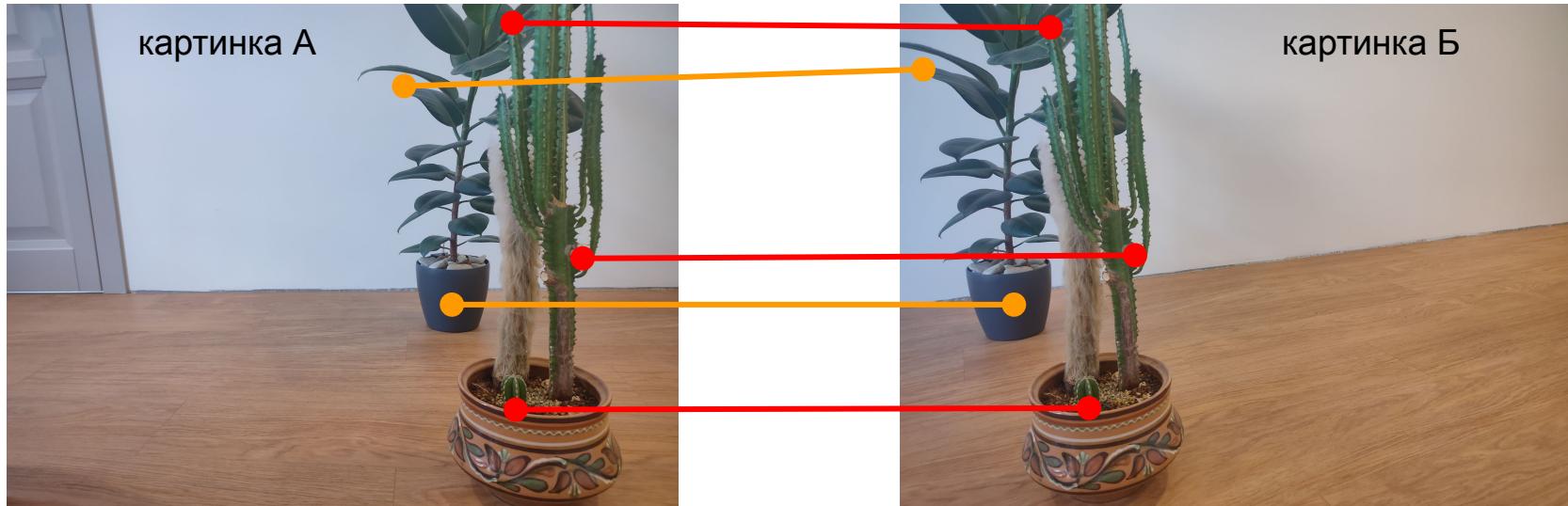
[Источник картинки](#)

Панорама полок магазинов (для мерчендайзеров)



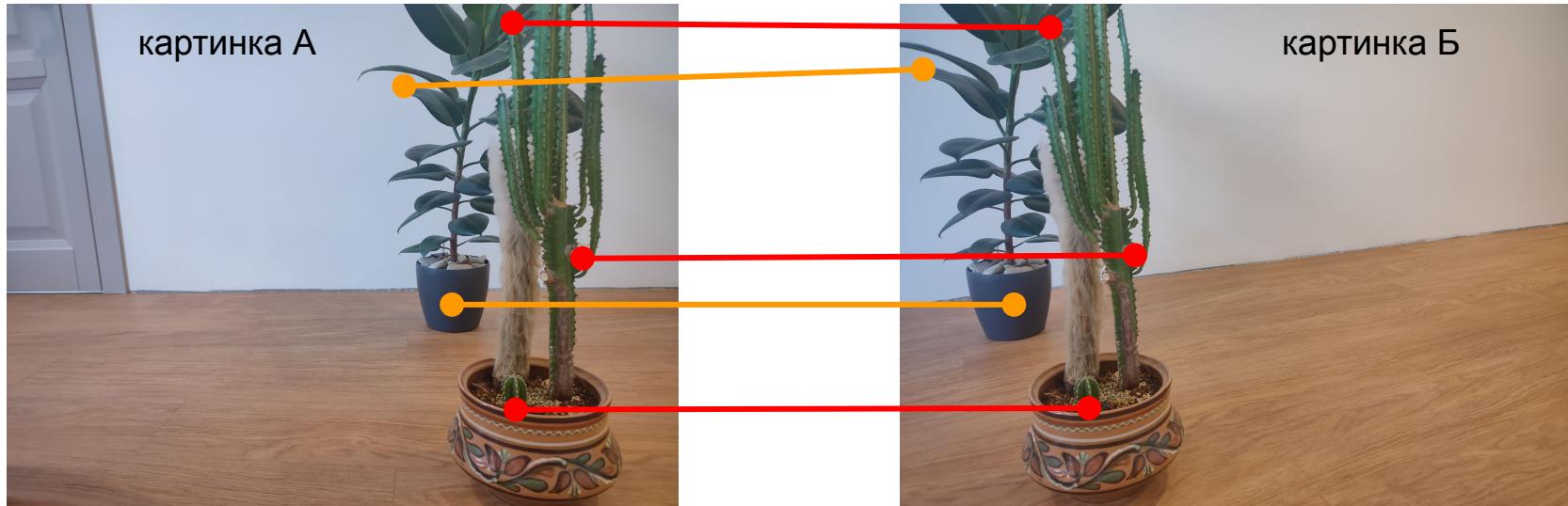
План решения

1. Найти как фотографии перекрываются (накладываются друг на друга)
т.е. сопоставили ключевые точки на фотографиях:



План решения

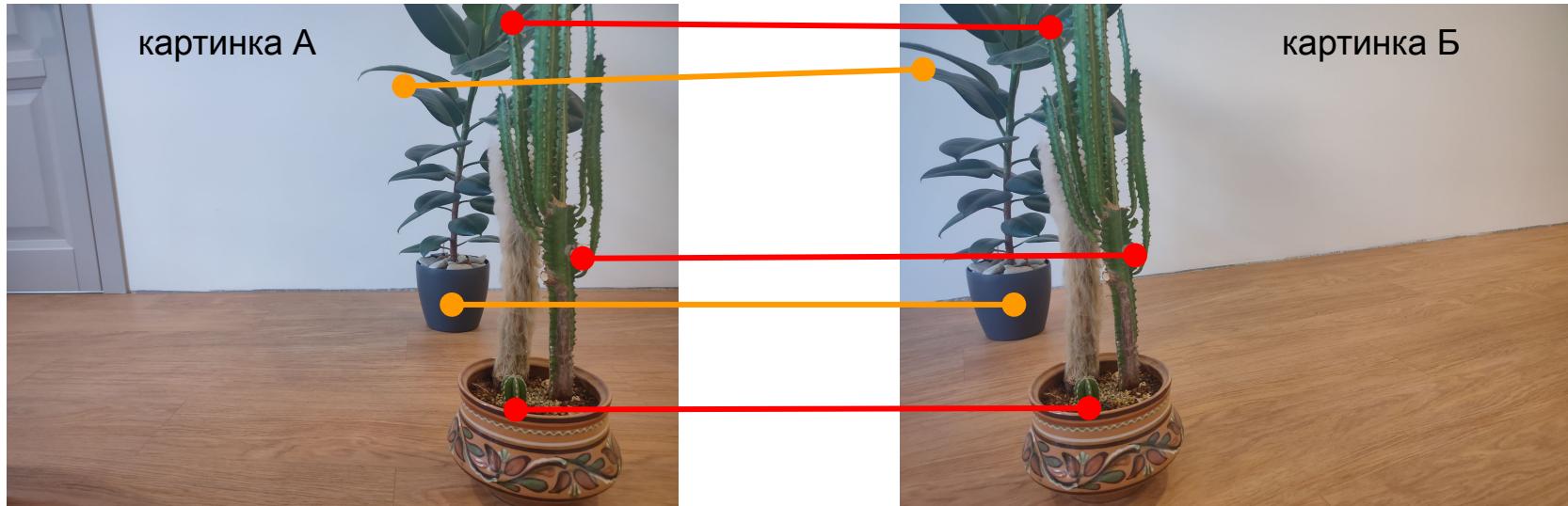
- Найти как фотографии перекрываются (накладываются друг на друга)
т.е. сопоставили ключевые точки на фотографиях:



Непрерывно ли это отображение плоскости изображения?

План решения

- Найти как фотографии перекрываются (накладываются друг на друга) т.е. сопоставили ключевые точки на фотографиях:



Непрерывно ли это отображение плоскости изображения?
Можно ли из подобных двух кадров сделать мини-панораму?

План решения

- Найти как фотографии перекрываются (накладываются друг на друга) т.е. сопоставили ключевые точки на фотографиях:



Непрерывно ли это отображение плоскости изображения?
Можно ли из подобных двух кадров сделать мини-панораму?

План решения

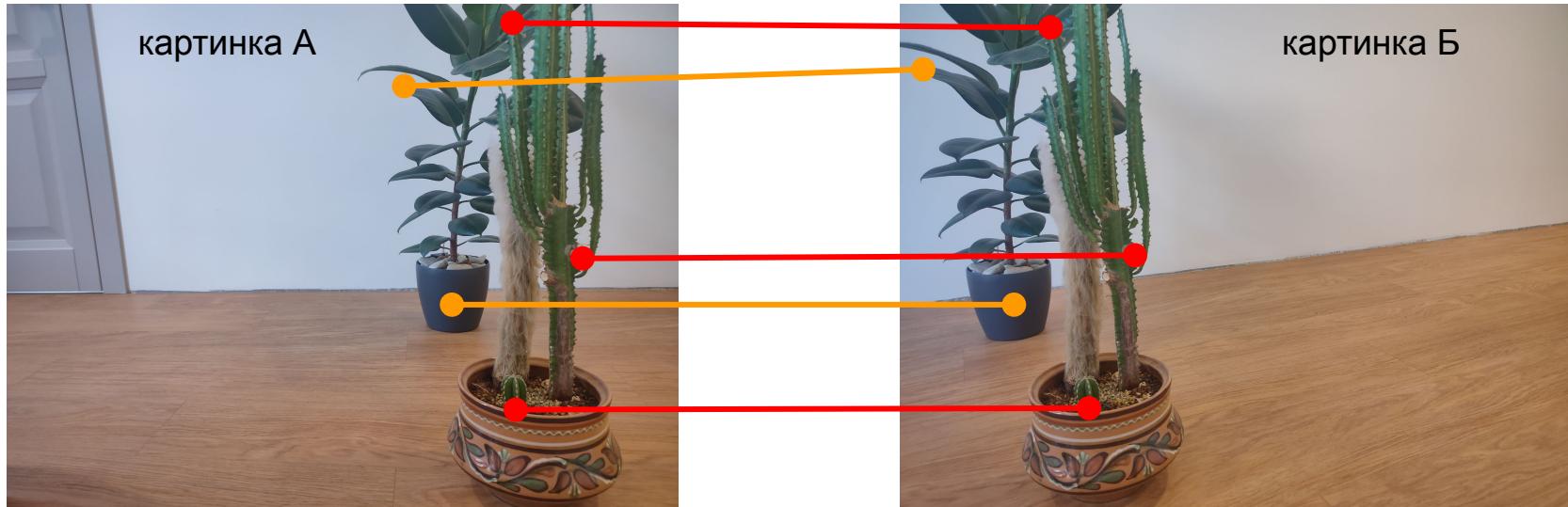
- Найти как фотографии перекрываются (накладываются друг на друга) т.е. сопоставили ключевые точки на фотографиях:



Непрерывно ли это отображение плоскости изображения?
Можно ли из подобных двух кадров сделать мини-панораму?

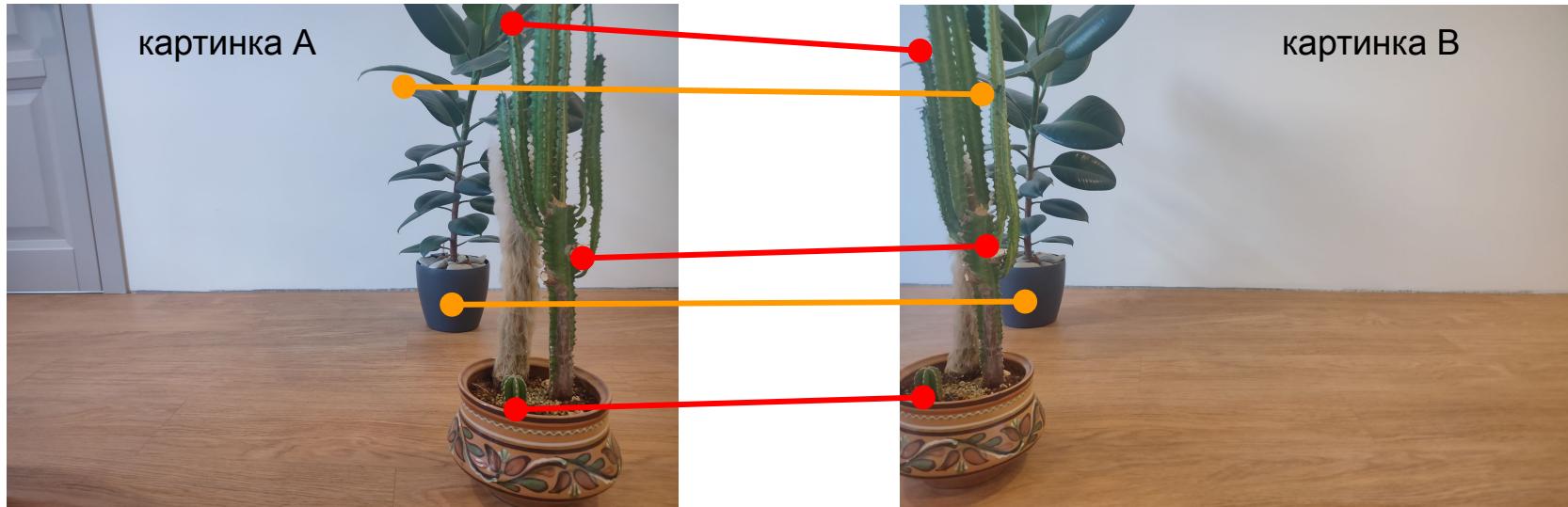
План решения

1. Найти как фотографии перекрываются (накладываются друг на друга)
т.е. сопоставили ключевые точки на фотографиях:



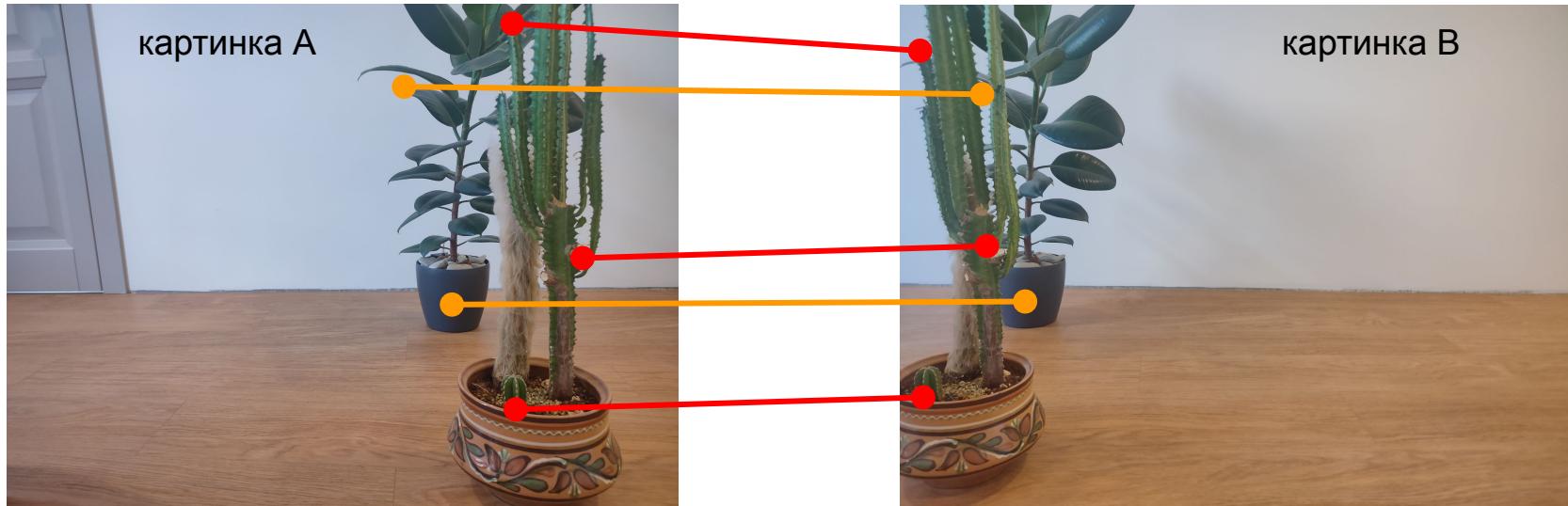
План решения

- Найти как фотографии перекрываются (накладываются друг на друга)
т.е. сопоставили ключевые точки на фотографиях:



План решения

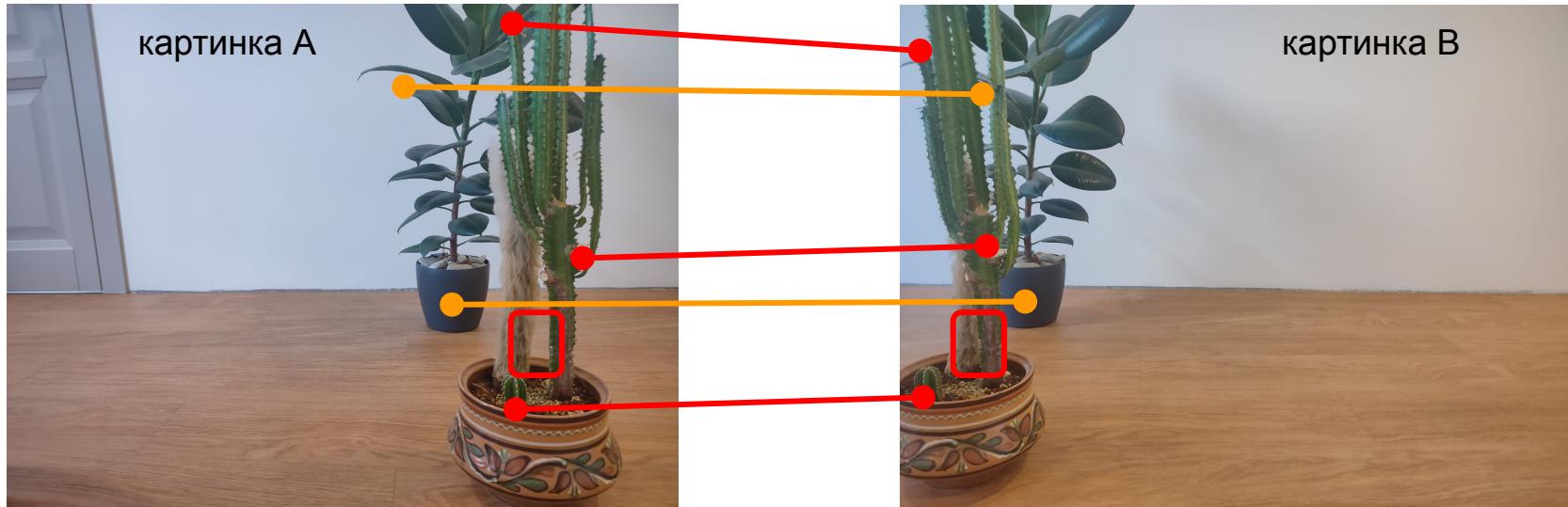
- Найти как фотографии перекрываются (накладываются друг на друга)
т.е. сопоставили ключевые точки на фотографиях:



Непрерывно ли это отображение плоскости изображения?

План решения

- Найти как фотографии перекрываются (накладываются друг на друга) т.е. сопоставили ключевые точки на фотографиях:



Непрерывно ли это отображение плоскости изображения?
Можно ли из подобных двух кадров сделать мини-панораму?

План решения

- Найти как фотографии перекрываются (накладываются друг на друга)
т.е. сопоставили ключевые точки на фотографиях:



**Непрерывно ли это отображение плоскости изображения?
Можно ли из подобных двух кадров сделать мини-панораму?**

Панорама полок магазинов (для мерчендайзеров)



Непрерывно ли это отображение в данном случае?
Можно ли из подобных кадров сделать панораму?

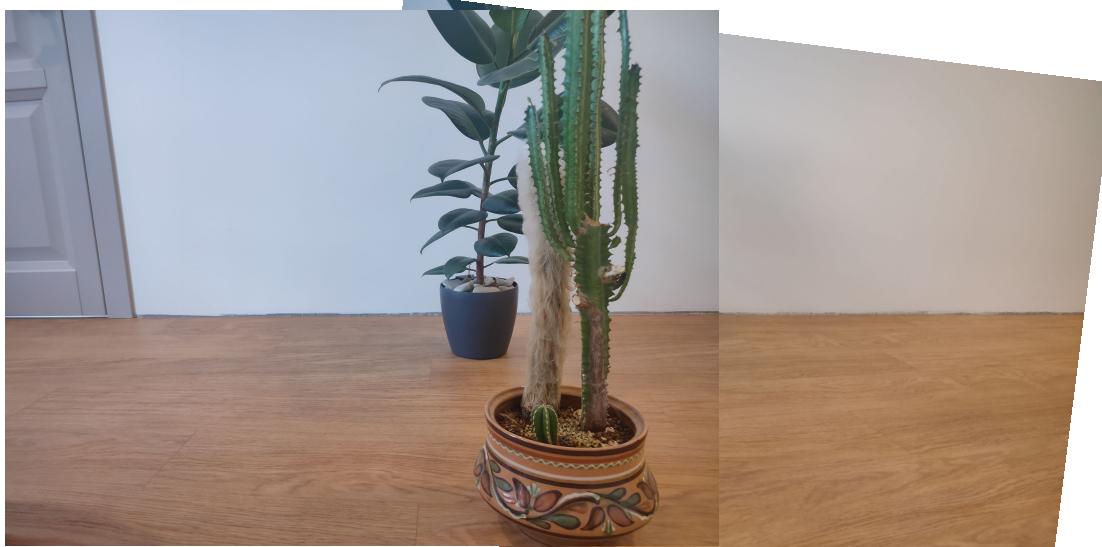
План решения

1. Найти как фотографии перекрываются (накладываются друг на друга) т.е. сопоставили ключевые точки на фотографиях.
2. Наложили картинки переходом в единую плоскость.
(натянули картинки начиная с первой так чтобы точки совпали)



План решения

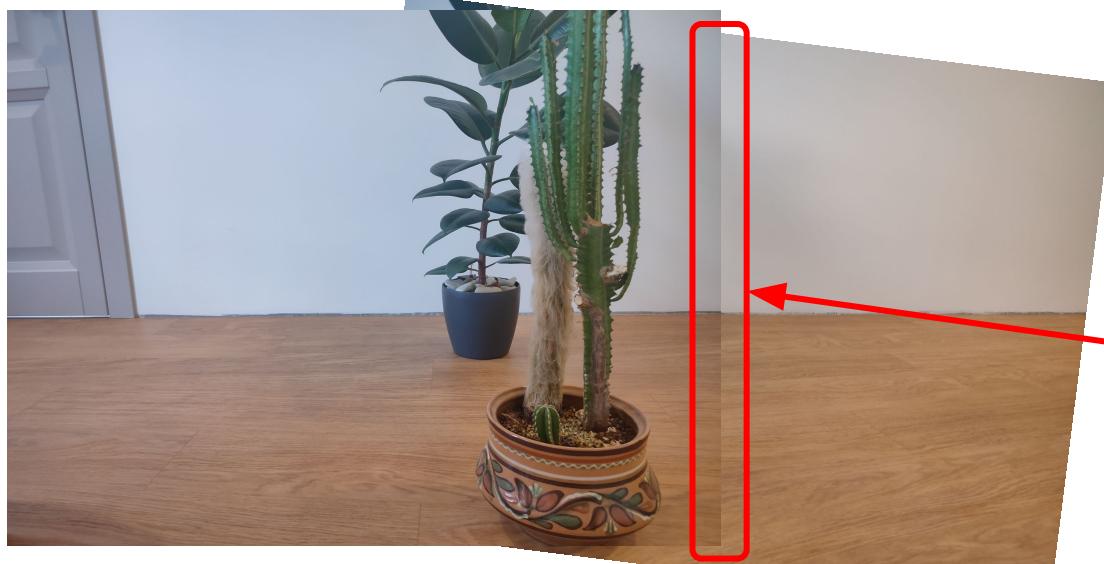
1. Найти как фотографии перекрываются (накладываются друг на друга) т.е. сопоставили ключевые точки на фотографиях.
2. Наложили картинки переходом в единую плоскость.
(натянули картинки начиная с первой так чтобы точки совпали)



Что дальше?
Какие есть дефекты?

План решения

1. Найти как фотографии перекрываются (накладываются друг на друга) т.е. сопоставили ключевые точки на фотографиях.
2. Наложили картинки переходом в единую плоскость.
(натянули картинки начиная с первой так чтобы точки совпали)



Что дальше?
Какие есть дефекты?
Очень заметный шов на стыке!

План решения

1. Найти как фотографии перекрываются (накладываются друг на друга)
т.е. сопоставили ключевые точки на фотографиях.
2. Наложили картинки переходом в единую плоскость.
(натянули картинки начиная с первой так чтобы точки совпали)
3. Бесшовное смещивание картинок чтобы швы были меньше заметны.
(борьба с перепадом яркости изображения)

Достаточно ли выравнивать яркость на стыках?

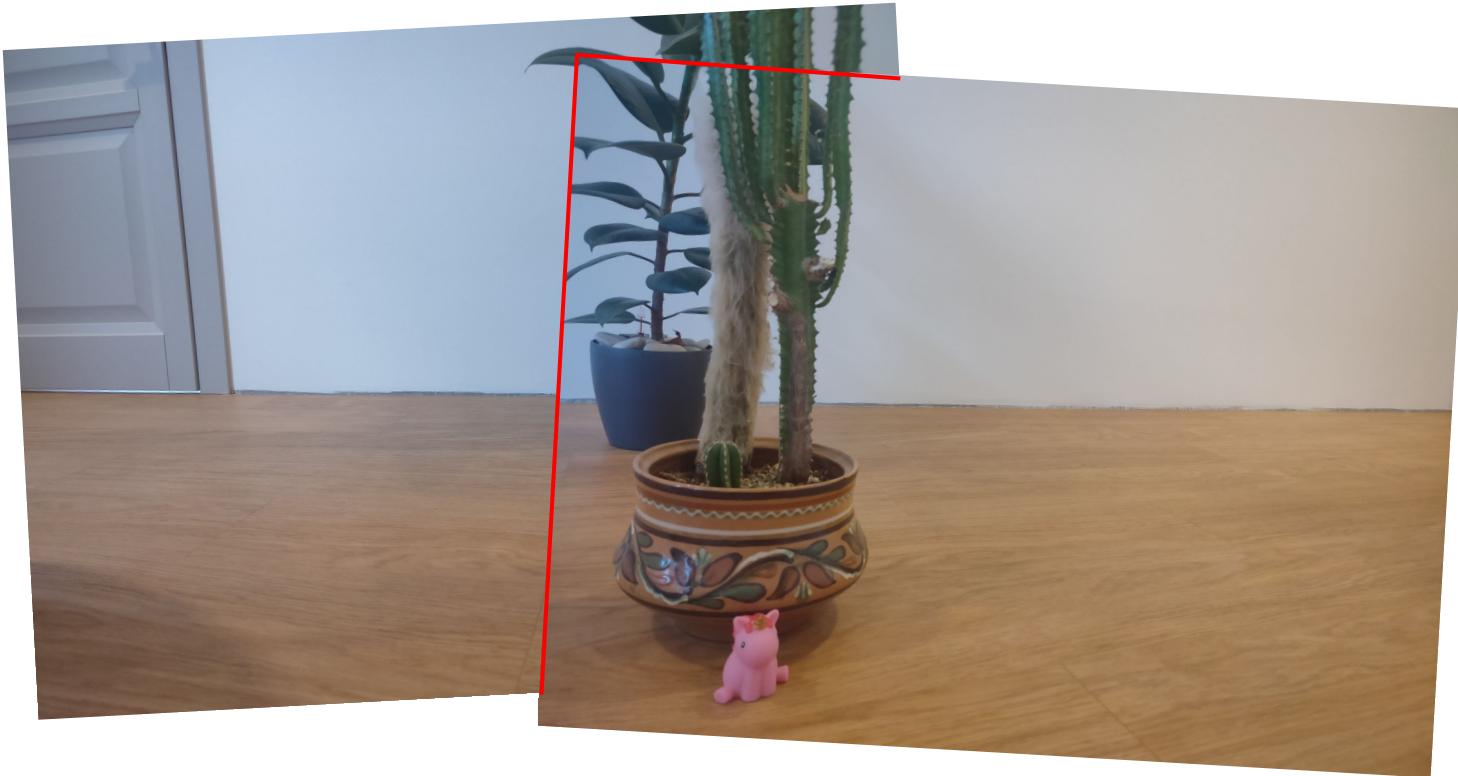


Достаточно ли выравнивать яркость на стыках?



Непрерывно ли это отображение плоскости изображения?
Можно ли из подобных двух кадров сделать мини-панораму?

Достаточно ли выравнивать яркость на стыках?



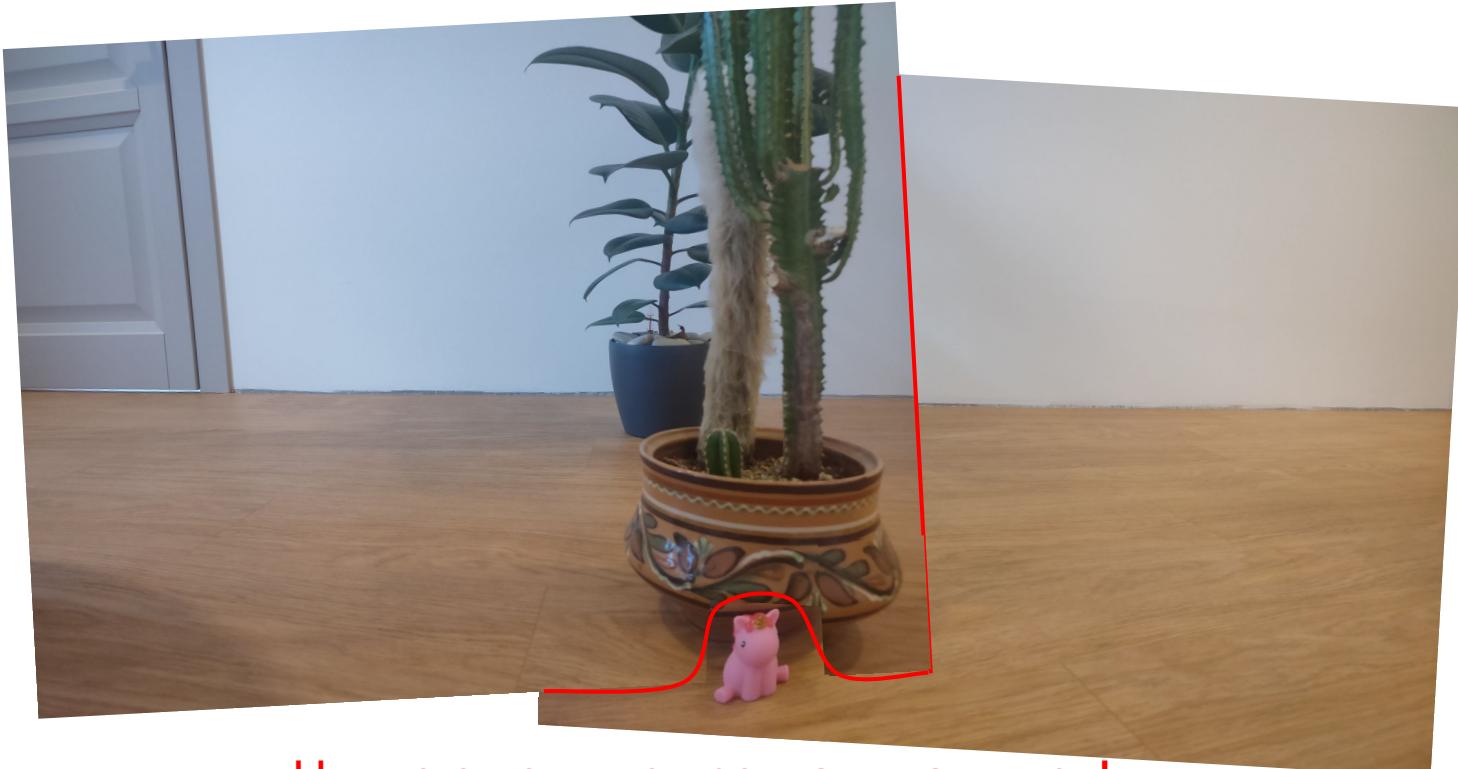
Достаточно ли выравнивать яркость на стыках?



Достаточно ли выравнивать яркость на стыках?



Достаточно ли выравнивать яркость на стыках?



Нужно еще умно прокладывать шов!
Чтобы ни один Единорог не пострадал!

План решения

1. Найти как фотографии перекрываются (накладываются друг на друга)
т.е. сопоставили ключевые точки на фотографиях.
2. Наложили картинки переходом в единую плоскость.
(натянули картинки начиная с первой так чтобы точки совпали)
3. Бесшовное смешивание картинок чтобы швы были меньше заметны.
(борьба с перепадом яркости изображения)
4. Умная прокладка швов. Чтобы не прокладывать швы через “призраков”:
 - проходящие туристы
 - проезжающие машины
 - пробегающие Единороги

План решения

1. Найти как фотографии перекрываются (накладываются друг на друга) т.е. сопоставили ключевые точки на фотографиях.
2. Наложили картинки переходом в единую плоскость.
(натянули картинки начиная с первой так чтобы точки совпали)
3. Бесшовное смешивание картинок чтобы швы были меньше заметны.
(борьба с перепадом яркости изображения)
4. Умная прокладка швов. Чтобы не прокладывать швы через “призраков”:
 - проходящие туристы
 - проезжающие машины
 - пробегающие Единороги

Все или ничего: призрак должен быть либо целиком, либо не быть вообще.
Но никак нельзя его разрезать пополам.

План решения

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

План решения

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

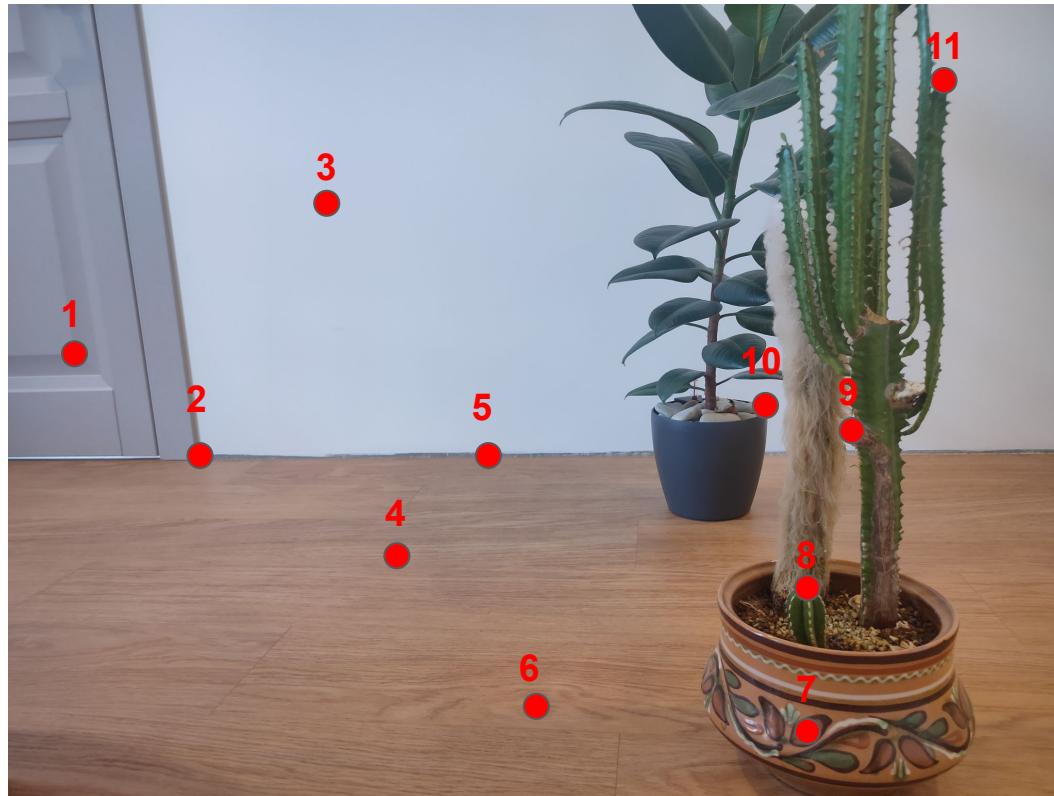
1. Создание ключевых точек: Detection

Какие точки удобны для сопоставления? Какие к ним требования?



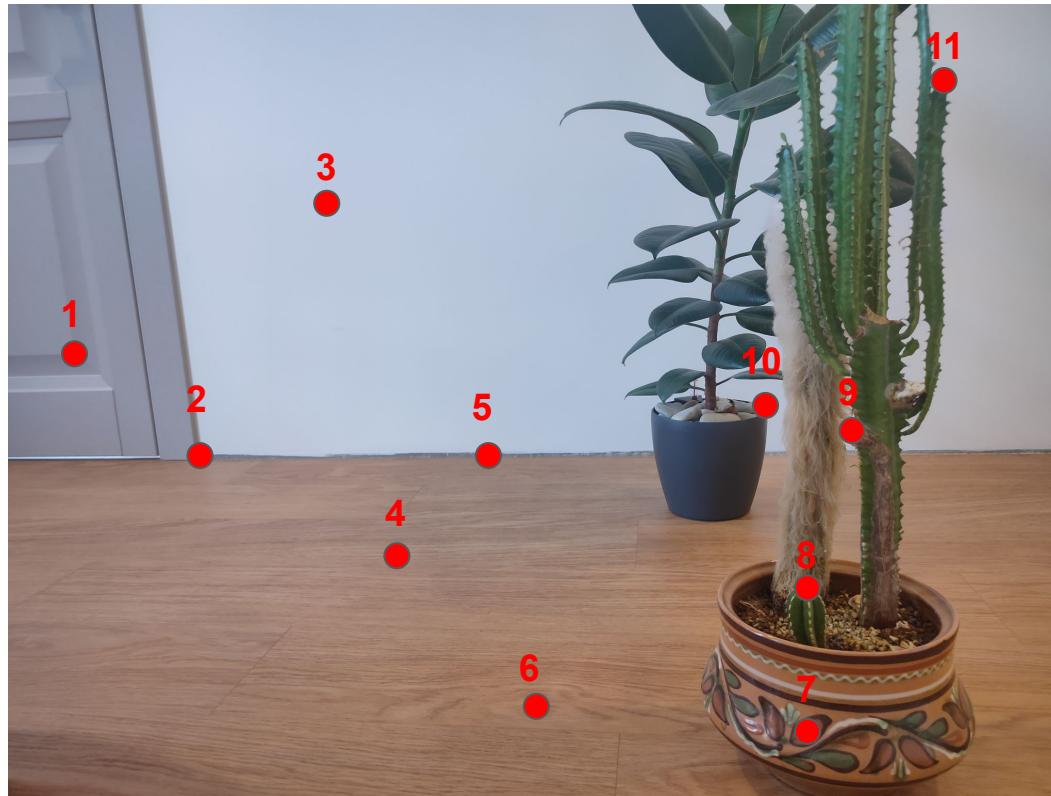
1. Создание ключевых точек: Detection

Какие точки удобны для сопоставления? Какие к ним требования?



1. Создание ключевых точек: Detection

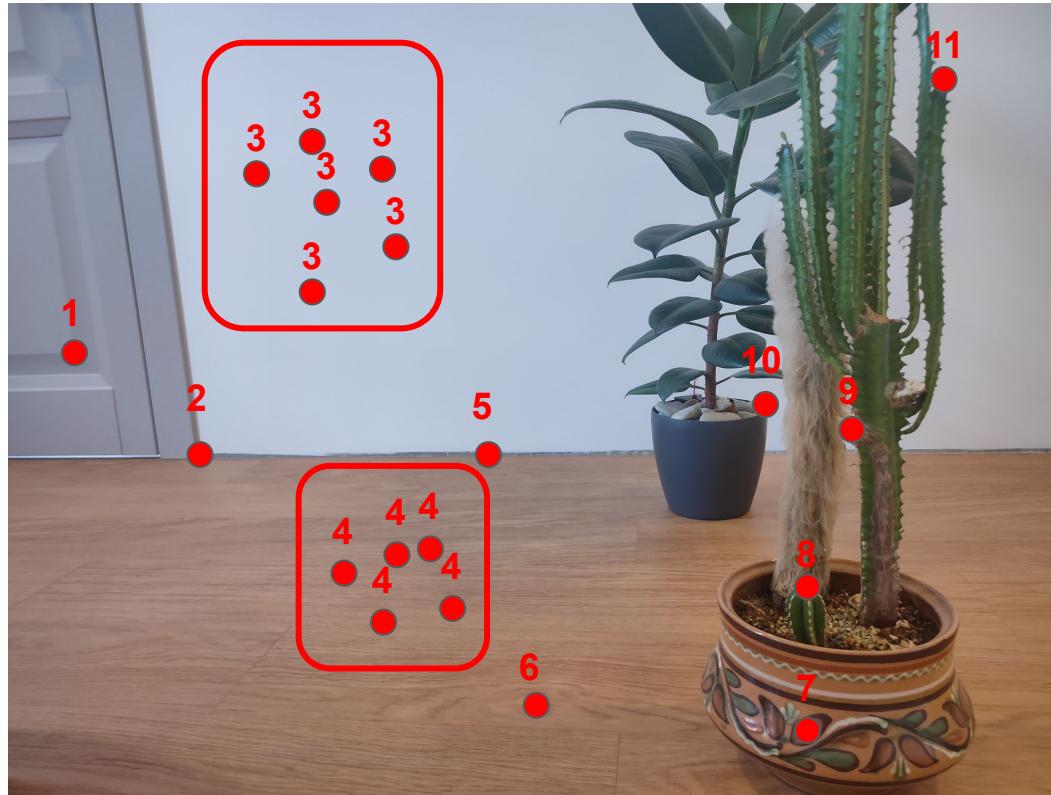
Какие точки удобны для сопоставления? Какие к ним требования?



Инвариантность: фотография с другого ракурса должна обнаружить те же точки.

1. Создание ключевых точек: Detection

Какие точки удобны для сопоставления? Какие к ним требования?



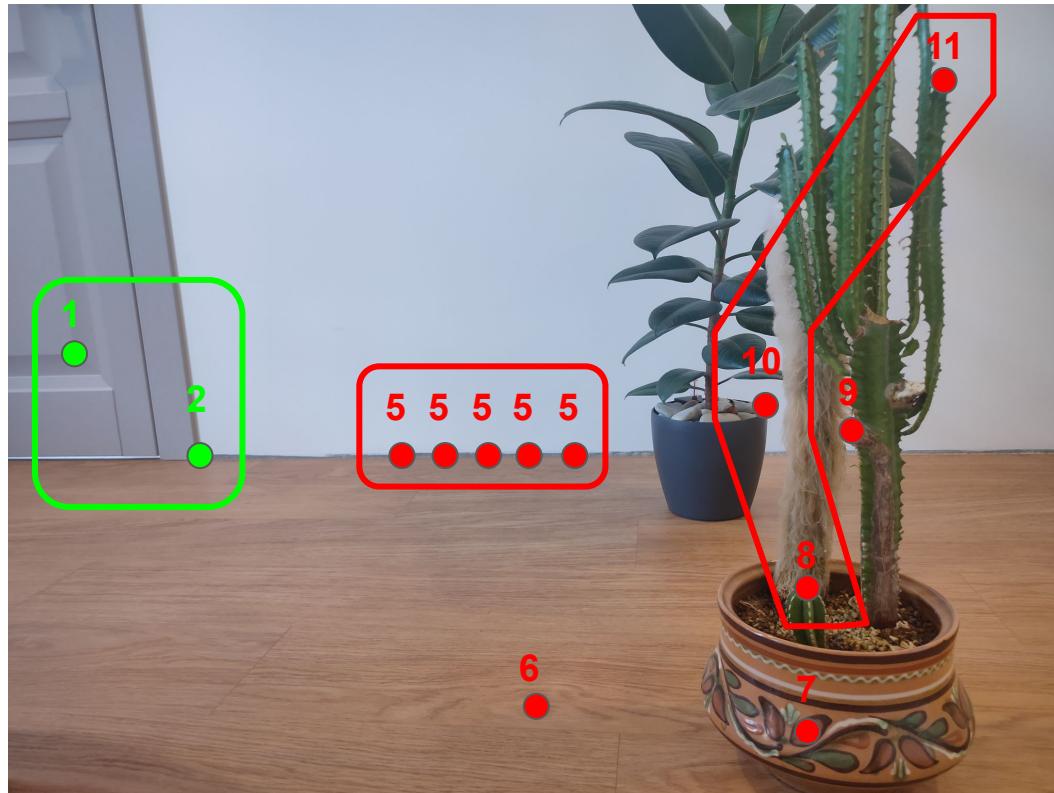
Инвариантность: фотография с другого ракурса должна обнаружить те же точки.

Точка 3: стена белая, зацепиться не за что, текстуры нет, по ней легко “скользить”. Точка не однозначная и большой риск ошибки сопоставления.

Точка 4: зависит, может за текстуру дерева можно однозначно зацепиться, но если кадр размытый - тоже неоднозначность сопоставления.

1. Создание ключевых точек: Detection

Какие точки удобны для сопоставления? Какие к ним требования?



Инвариантность: фотография с другого ракурса должна обнаружить те же точки.

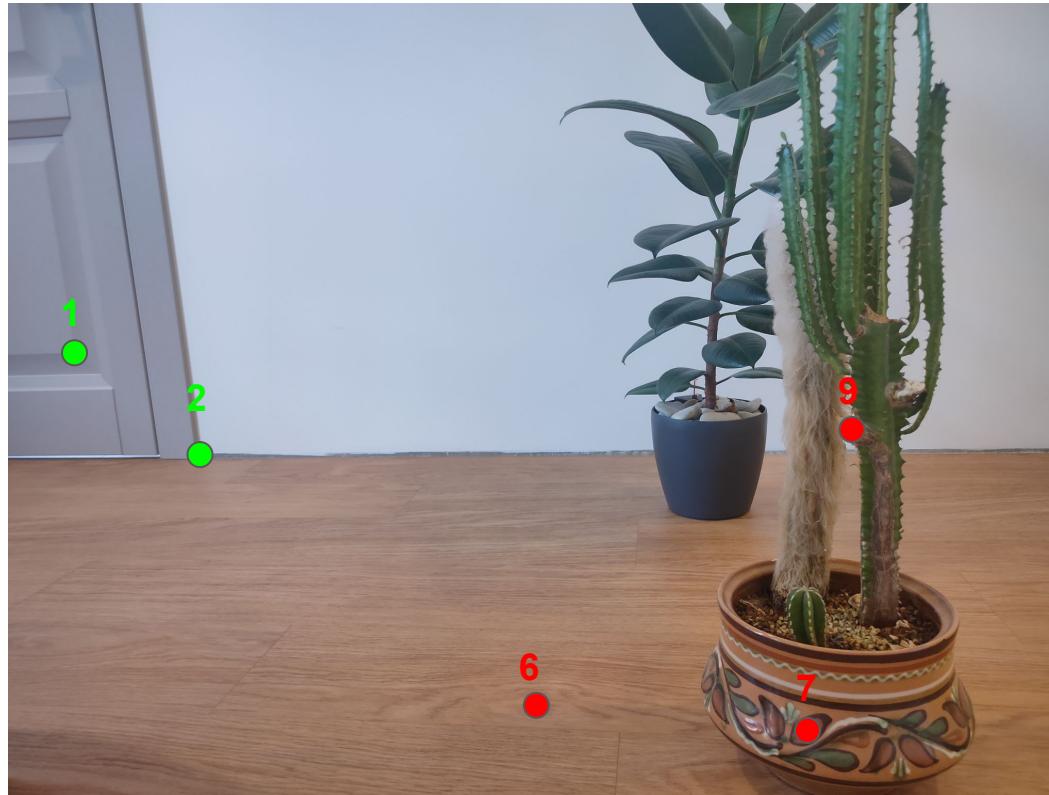
Точка 5: скользит по краю пола.

Точки 1 и 2: хорошие, они на углу. Причем **на углу плоскости**. А не на углу чье пространственное положение зависит от положения камеры (см. **точки 8,10,11**), а значит может не воспроизвестись (**точка 10**) или воспроизвестись неточно (**точки 8,11**).

Углы - критерий ключевых точек в **Harris Corner Detector**.

1. Создание ключевых точек: Detection

Какие точки удобны для сопоставления? Какие к ним требования?



Инвариантность: фотография с другого ракурса должна обнаружить те же точки.

Точки 6, 7 и 9: инвариантны или нет?

1. Создание ключевых точек: Detection

Какие точки удобны для сопоставления? Какие к ним требования?

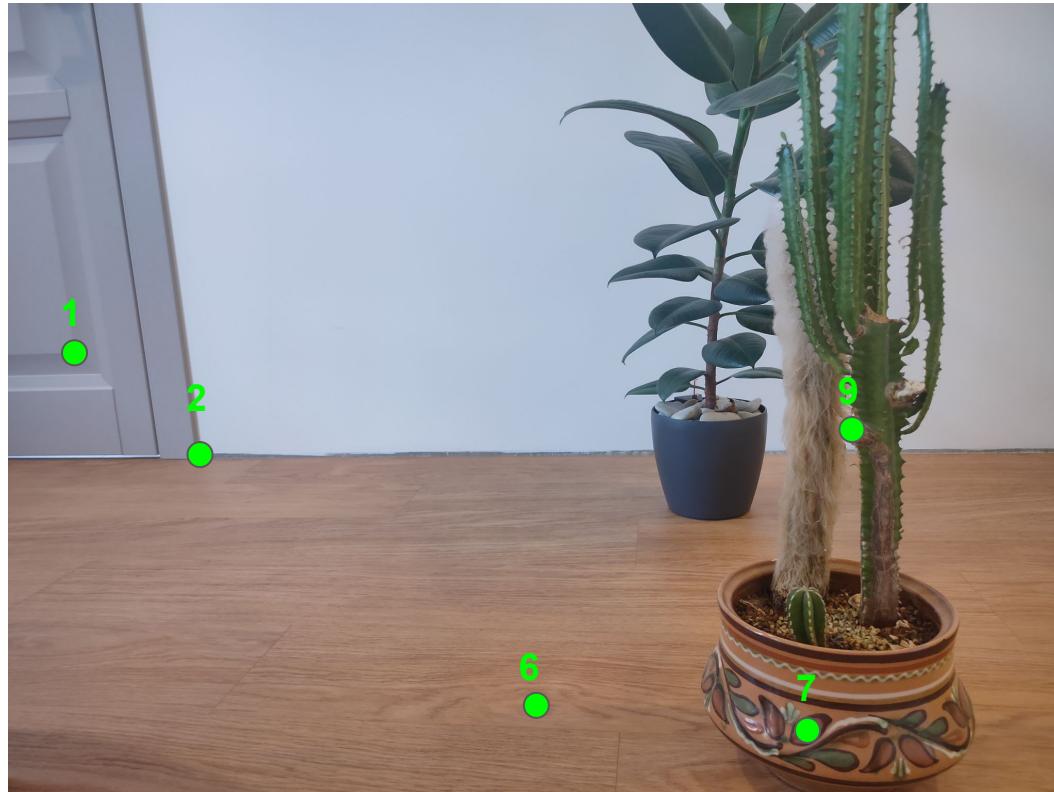


Инвариантность: фотография с другого ракурса должна обнаружить те же точки.

Точки 6, 7 и 9: инвариантны или нет?

1. Создание ключевых точек: Detection

Какие точки удобны для сопоставления? Какие к ним требования?



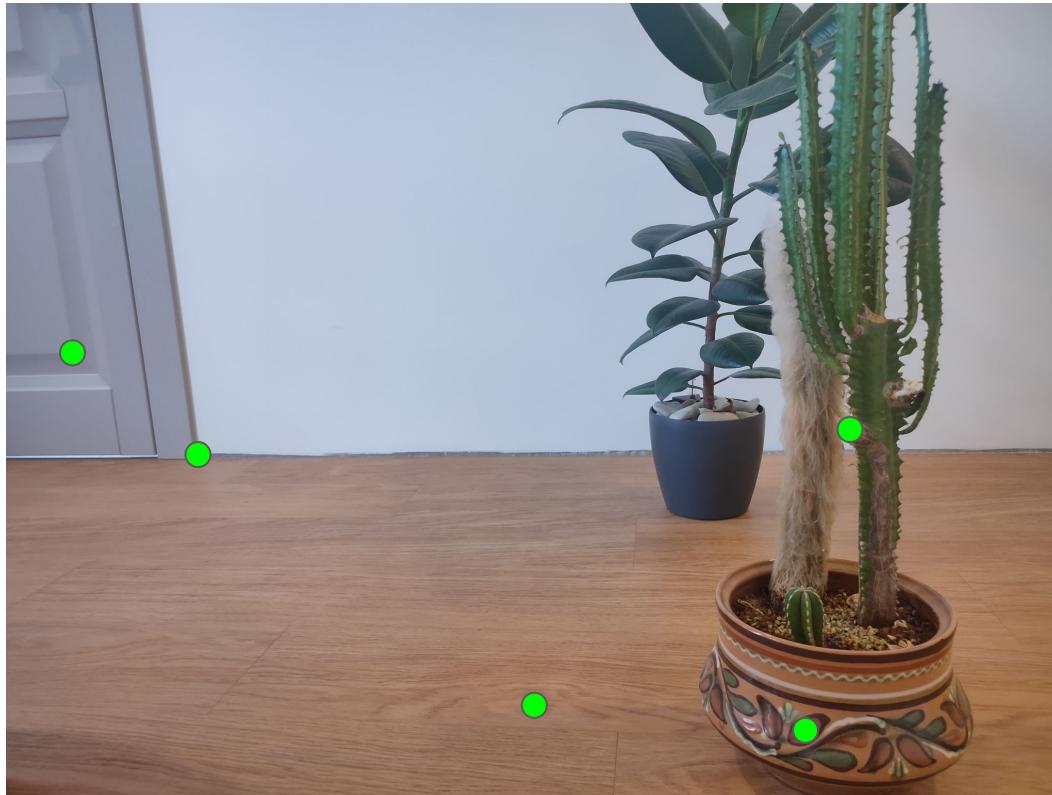
Инвариантность: фотография с другого ракурса должна обнаружить те же точки.

Точки 6, 7 и 9: находятся в центре пятна. Пятно лежащее на плоскости **инвариантно** т.к. его центр можно выбрать точно независимо от ракурса камеры.

Пятна - критерий ключевых точек в **SIFT Detector**.

1. Создание ключевых точек: SIFT Descriptor

Как сопоставлять точки между двумя картинками? Построить им компактное описание - Descriptor.



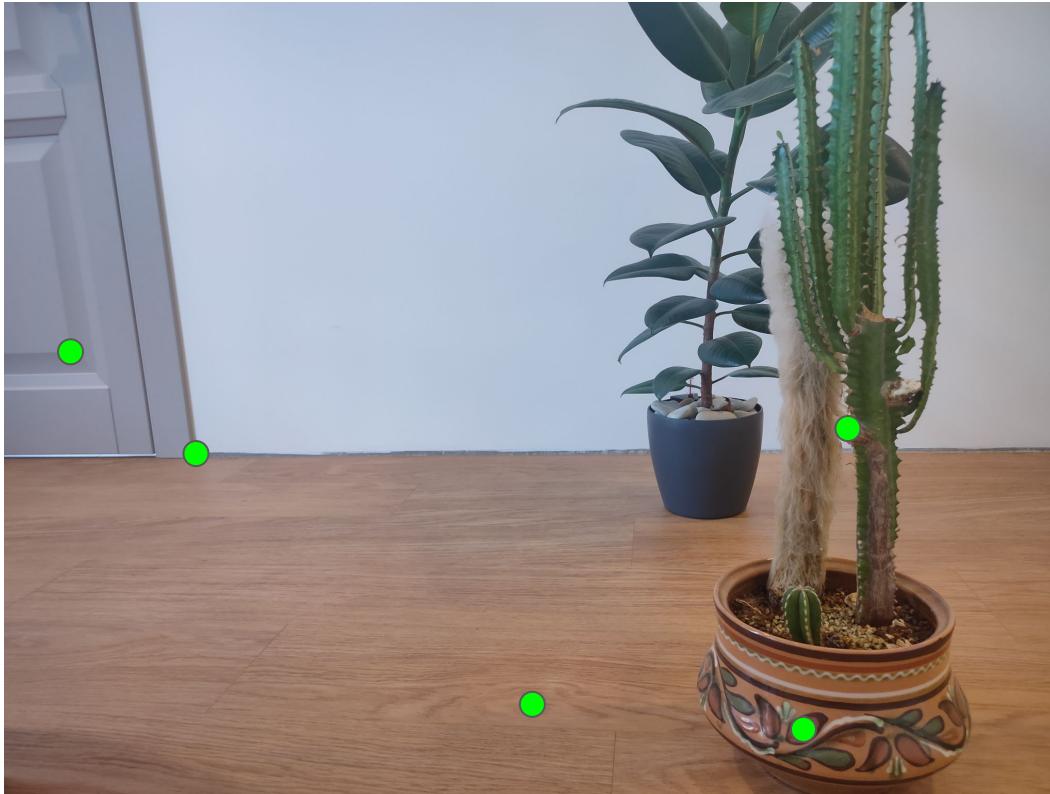
SIFT Descriptor: 128 Float = 512 bytes

Как оценить похожесть двух точек?

Как оценить похожесть двух
дескрипторов А и В?

1. Создание ключевых точек: SIFT Descriptor

Как сопоставлять точки между двумя картинками? Построить им компактное описание - Descriptor.



SIFT Descriptor: 128 Float = 512 bytes

Как оценить похожесть двух точек?

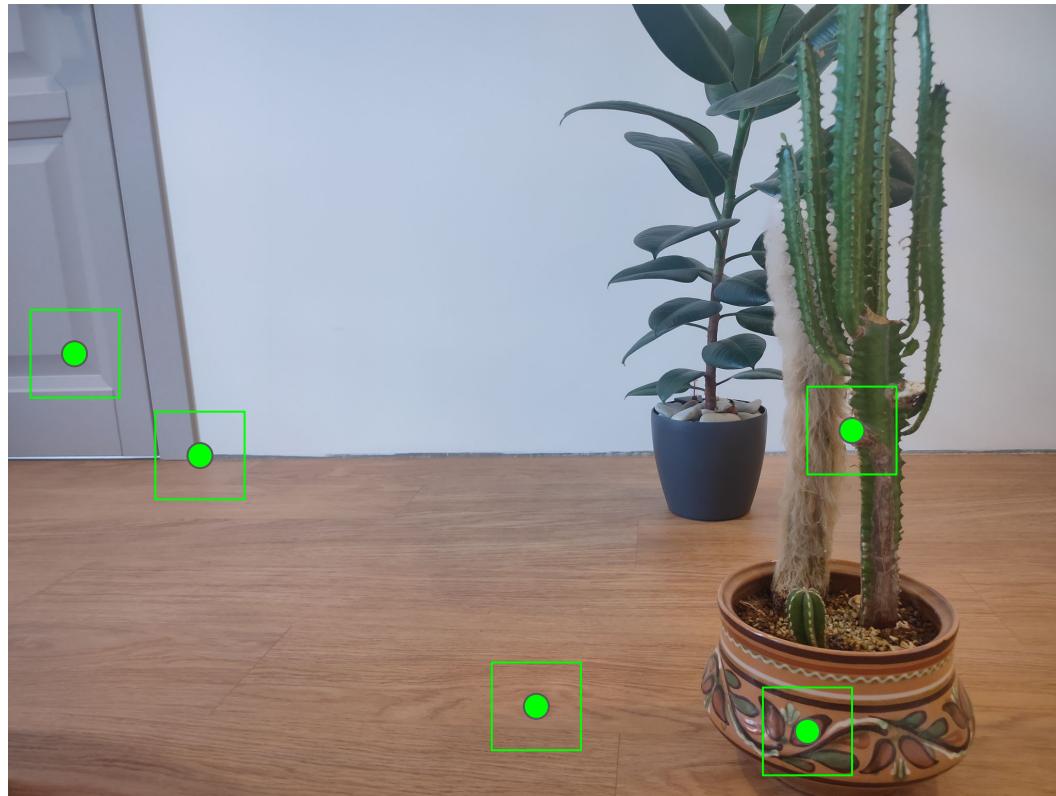
Как оценить похожесть двух дескрипторов А и В?

Посчитать между ними евклидово расстояние:

$$dist(A, B) = \sqrt{\sum_{i=1}^{128} (A_i - B_i)^2}$$

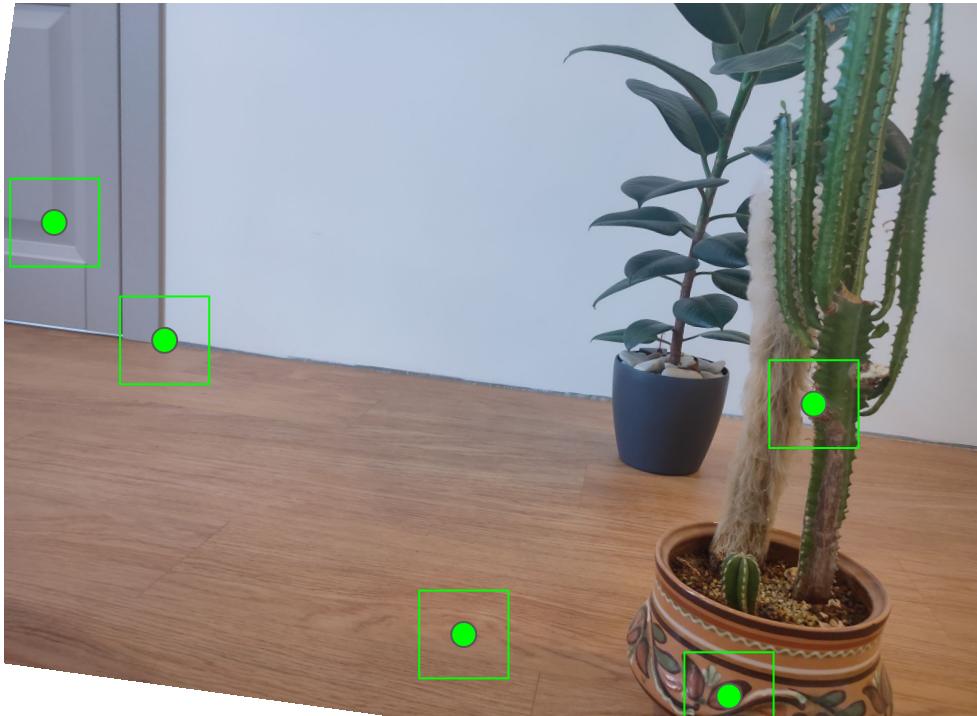
1. Создание ключевых точек: SIFT Descriptor

Какие есть проблемы с **инвариантностью Descriptor-а?**



1. Создание ключевых точек: SIFT Descriptor

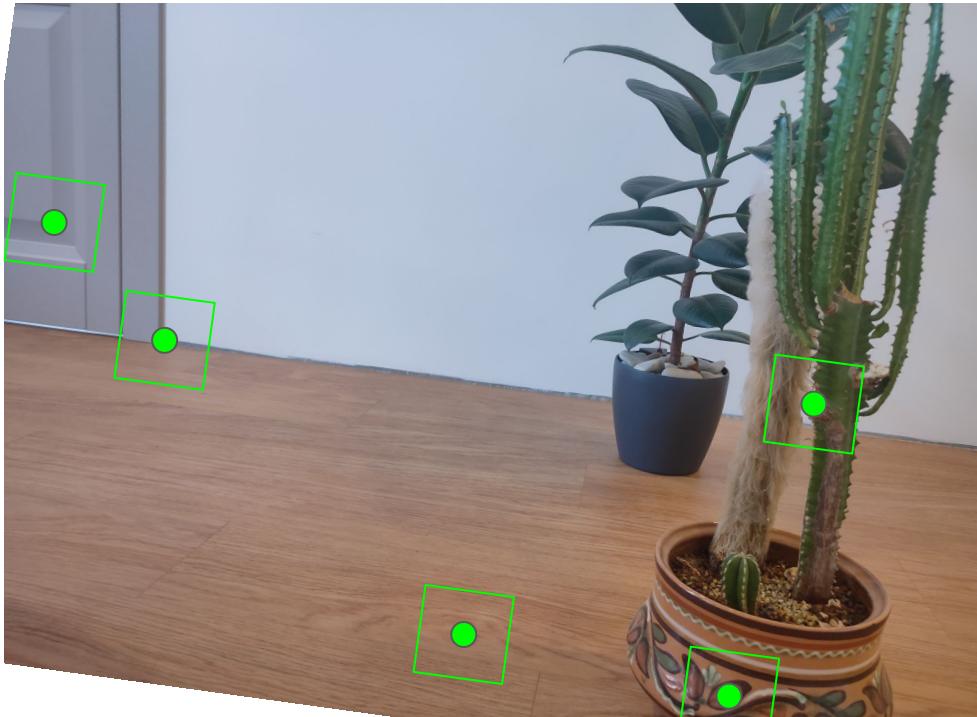
Какие есть проблемы с **инвариантностью Descriptor-a?**



Если другая фотография под углом - информация в локальной окрестности (**патч**) одних и тех же ключевых точек сильно меняется. Т.е. просто брать квадрат вокруг точки = **неинвариантный дескриптор**.

1. Создание ключевых точек: SIFT Descriptor

Какие есть проблемы с **инвариантностью Descriptor-а?**

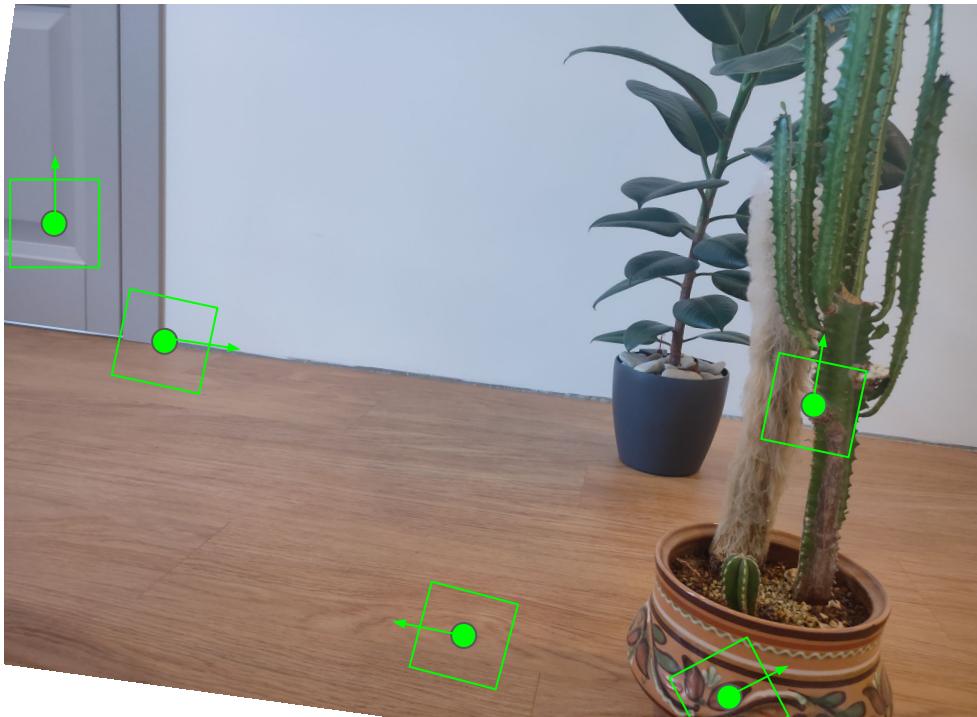


Если другая фотография под углом - информация в локальной окрестности (**патч**) одних и тех же ключевых точек сильно меняется. Т.е. просто брать квадрат вокруг точки = **неинвариантный дескриптор**.

Значит было бы хорошо повернуть **патчи** чтобы они были под тем же углом что и в оригинальной картинке. Но мы не знаем какой угол поворота!

1. Создание ключевых точек: SIFT Descriptor

Какие есть проблемы с **инвариантностью Descriptor-a?**



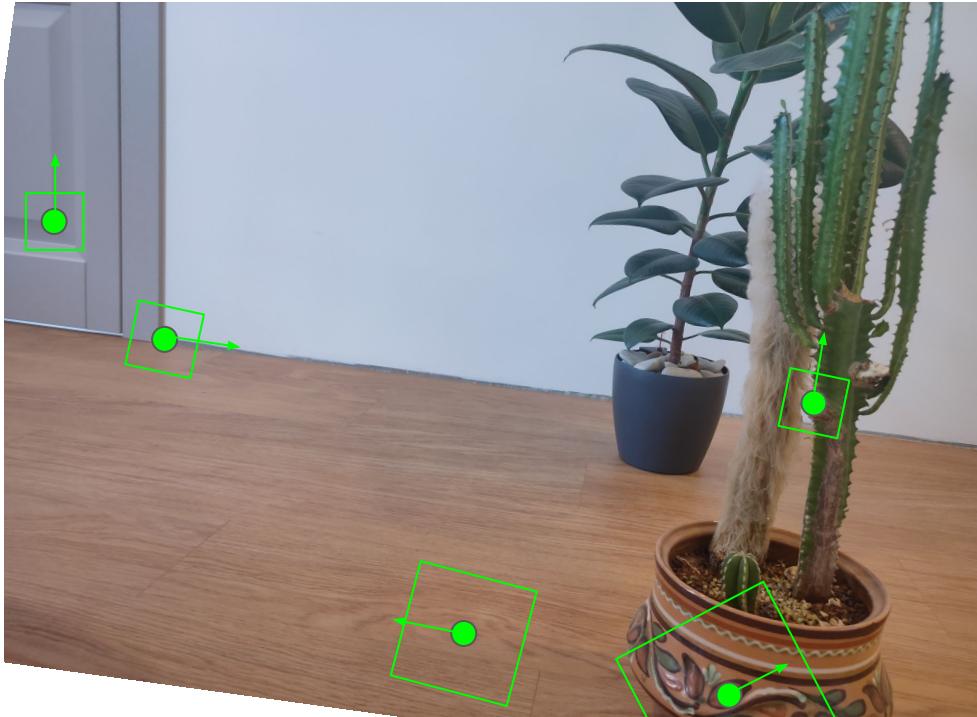
- 1) **Инвариантность по повороту:** ключевая точка выбирает направление (например самое популярное направление градиента в локальной окрестности).

Локальный патч повернут соответственно.

1. Создание ключевых точек: SIFT Descriptor

Какие есть проблемы с **инвариантностью Descriptor-a?**

- 1) Инвариантность по повороту
- 2) Инвариантность по масштабу

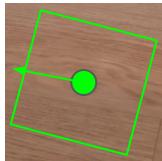


1. Создание ключевых точек: SIFT Descriptor

Какие есть проблемы с **инвариантностью Descriptor-a?**

- 1) Инвариантность по повороту
- 2) Инвариантность по масштабу
- 3) Инвариантность по описанию

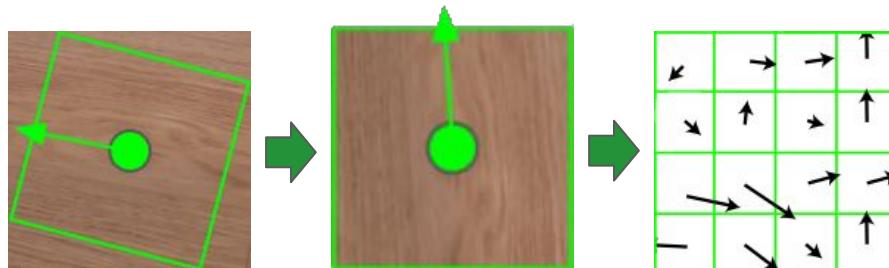
Теперь надо придумать алгоритм который представляет в виде 128 вещественных чисел локальный патч (поворнутый и “правильного” размера).



1. Создание ключевых точек: SIFT Descriptor

Какие есть проблемы с **инвариантностью Descriptor-a?**

- 1) Инвариантность по повороту
- 2) Инвариантность по масштабу
- 3) Инвариантность по описанию



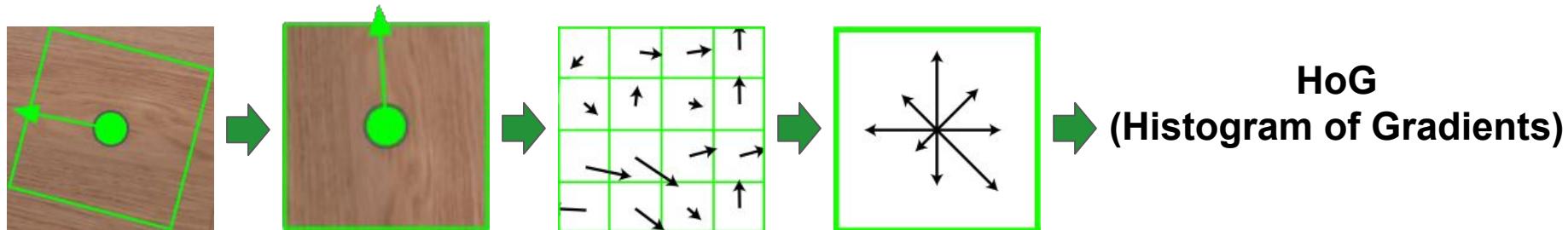
1. Создание ключевых точек: SIFT Descriptor

Какие есть проблемы с **инвариантностью Descriptor-a?**

HoG = N чисел: одно число - одно направление.

Число получается суммой длин градиентов
которые указывают в этом направлении.

- 1) Инвариантность по повороту
- 2) Инвариантность по масштабу
- 3) Инвариантность по описанию



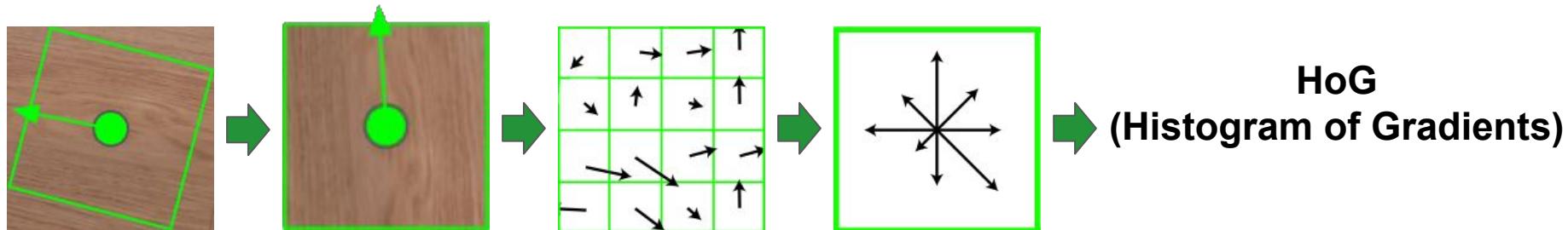
1. Создание ключевых точек: SIFT Descriptor

Какие есть проблемы с **инвариантностью Descriptor-a?**

HoG = N чисел: одно число - одно направление.

Число получается суммой длин градиентов
которые указывают в этом направлении.

- 1) Инвариантность по повороту
- 2) Инвариантность по масштабу
- 3) Инвариантность по описанию:
**Какие изменения картинки
почти не меняют HoG?**



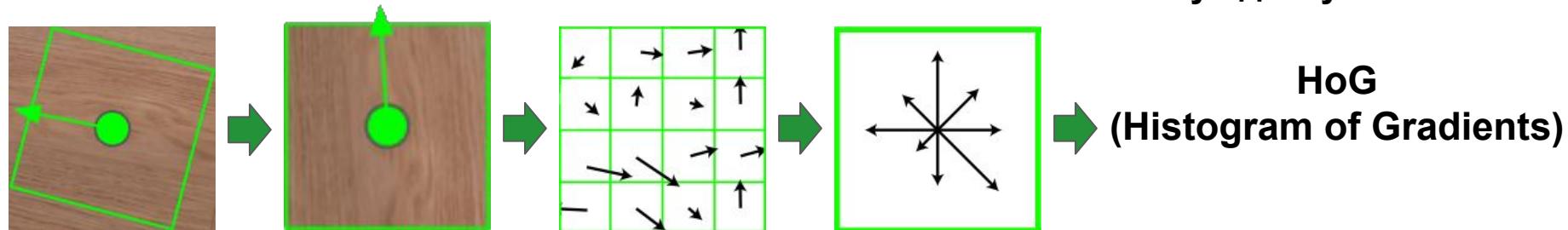
1. Создание ключевых точек: SIFT Descriptor

Какие есть проблемы с **инвариантностью Descriptor-a?**

HoG = N чисел: одно число - одно направление.

Число получается суммой длин градиентов
которые указывают в этом направлении.

- 1) Инвариантность по повороту
- 2) Инвариантность по масштабу
- 3) Инвариантность по описанию:
 - К изменению яркости
 - К малому сдвигу



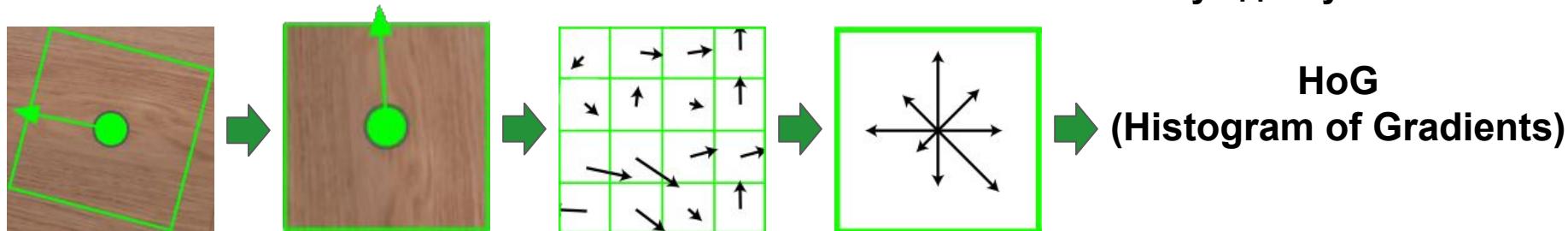
1. Создание ключевых точек: SIFT Descriptor

Какие есть проблемы с **инвариантностью Descriptor-a?**

HoG = N чисел: одно число - одно направление.

Число получается суммой длин градиентов
которые указывают в этом направлении.

- 1) Инвариантность по повороту
- 2) Инвариантность по масштабу
- 3) Инвариантность по описанию:
 - К изменению яркости
 - К малому сдвигу



SIFT Descriptor = HoG, т.е. 128-и корзинная гистограмма. В каждой корзине записано
насколько популярно это направление.

P.S. на самом деле локальный патч бьется на мини-патчи, и строится **несколько HoG** - для
каждого мини-патча. [Оригинальная статья](#). [Подробная лекция про SIFT](#).

2. Сопоставление ключевых точек

Пусть у нас есть две картинки.

Пусть в каждой **SIFT** выбрал ключевые
точки и построил для них дескрипторы.

2. Сопоставление ключевых точек

Пусть у нас есть две картинки.

Пусть в каждой **SIFT** выбрал ключевые точки и построил для них дескрипторы.

Как нам сопоставить точки картинки А с точками картинки Б?

SIFT Descriptor: 128 Float = 512 bytes

Как оценить похожесть двух точек?

Как оценить похожесть двух дескрипторов А и В?

Посчитать между ними евклидово расстояние:

$$dist(A, B) = \sqrt{\sum_{i=1}^{128} (A_i - B_i)^2}$$





2. Сопоставление ключевых точек

Пусть у нас есть две картинки.

Пусть в каждой **SIFT** выбрал ключевые точки и построил для них дескрипторы.

Как нам сопоставить точки картинки А с точками картинки Б?

SIFT Descriptor: 128 Float = 512 bytes

Как оценить похожесть двух точек?

Как оценить похожесть двух дескрипторов А и В?

Посчитать между ними евклидово расстояние:

$$dist(A, B) = \sqrt{\sum_{i=1}^{128} (A_i - B_i)^2}$$

2. Сопоставление ключевых точек

Пусть у нас есть две картинки.

Пусть в каждой **SIFT** выбрал ключевые точки и построил для них дескрипторы.

1) **Nearest Neighbor (NN):**

Для каждой точки картинки **A** найдем ближайшую точку картинки **B**
(по похожести дескрипторов)

SIFT Descriptor: 128 Float = 512 bytes

Как оценить похожесть двух точек?

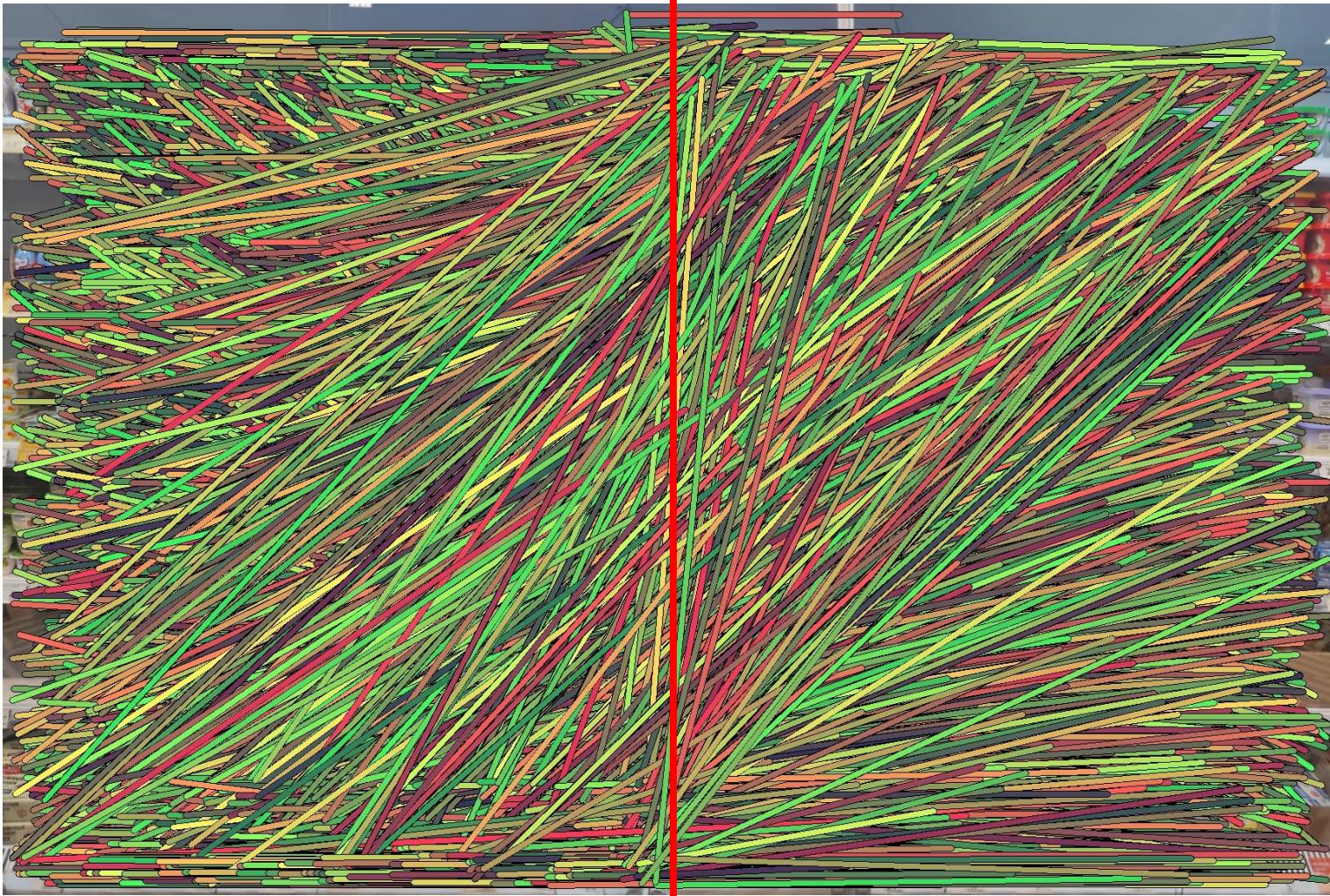
Как оценить похожесть двух дескрипторов **A** и **B**?

Посчитать между ними евклидово расстояние:

$$dist(A, B) = \sqrt{\sum_{i=1}^{128} (A_i - B_i)^2}$$

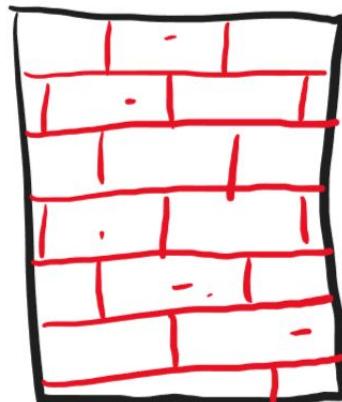
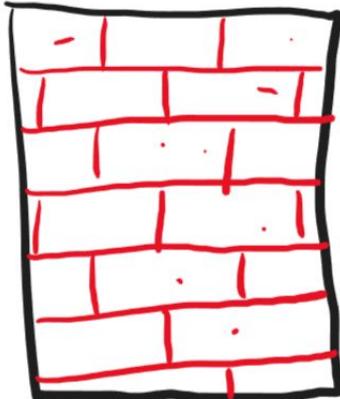


1. Nearest Neighbors



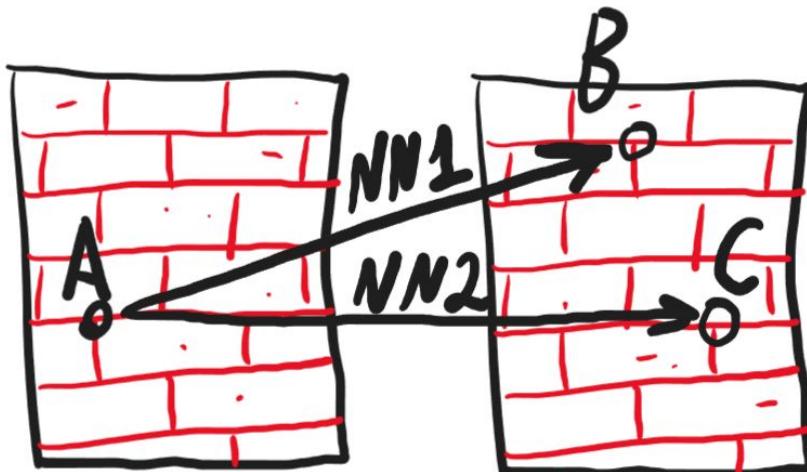
2. Сопоставление ключевых точек

- 1) **Nearest Neighbor (NN)**
- 2) **Как избавиться от очевидно плохих сопоставлений?**



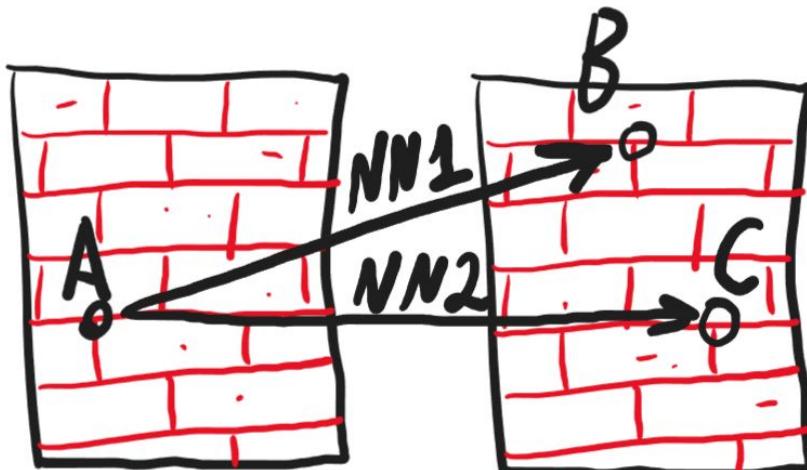
2. Сопоставление ключевых точек

- 1) **Nearest Neighbor (NN)**
- 2) Пусть мы искали KNN ($K=2$ Nearest Neighbors), т.е. два ближайших сопоставления.



2. Сопоставление ключевых точек

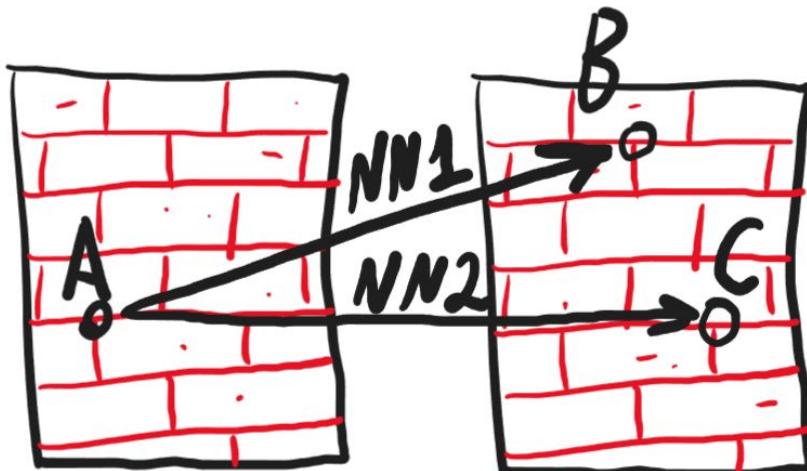
- 1) **Nearest Neighbor (NN)**
- 2) Пусть мы искали KNN (**K=2 Nearest Neighbors**), т.е. два ближайших сопоставления.



$$\text{dist}(A, B) = x$$
$$\text{dist}(A, C) = y$$

2. Сопоставление ключевых точек

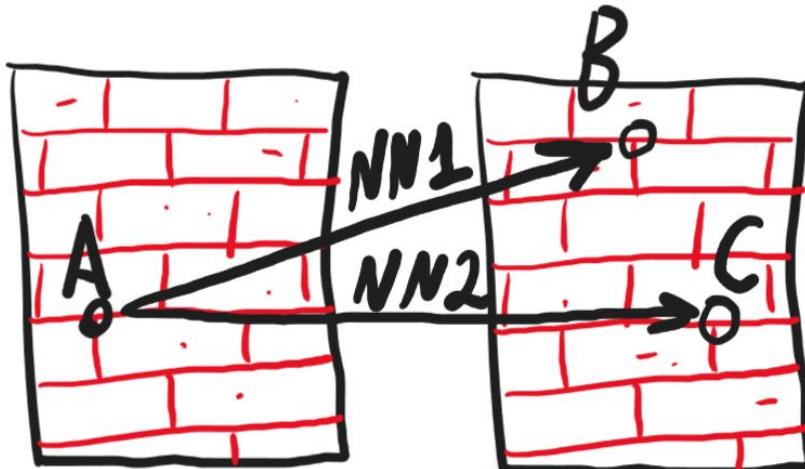
- 1) **Nearest Neighbor (NN)**
- 2) Пусть мы искали KNN (**K=2 Nearest Neighbors**), т.е. два ближайших сопоставления.



$$\text{dist}(A, B) = x$$
$$\text{dist}(A, C) = y$$

2. Сопоставление ключевых точек

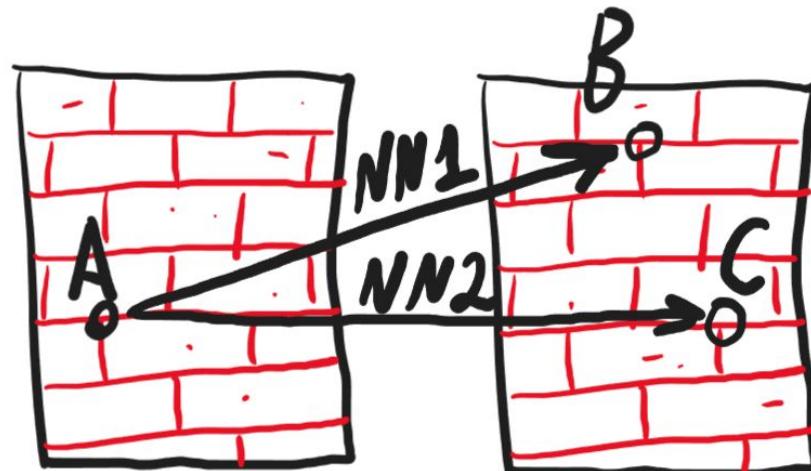
- 1) **Nearest Neighbor (NN)**
- 2) Пусть мы искали KNN (**K=2 Nearest Neighbors**), т.е. два ближайших сопоставления.



$$\text{dist}(A, B) = x$$
$$\text{dist}(A, C) = y$$

2. Сопоставление ключевых точек

- 1) **Nearest Neighbor (NN)**
- 2) Пусть мы искали KNN (**K=2 Nearest Neighbors**), т.е. два ближайших сопоставления.



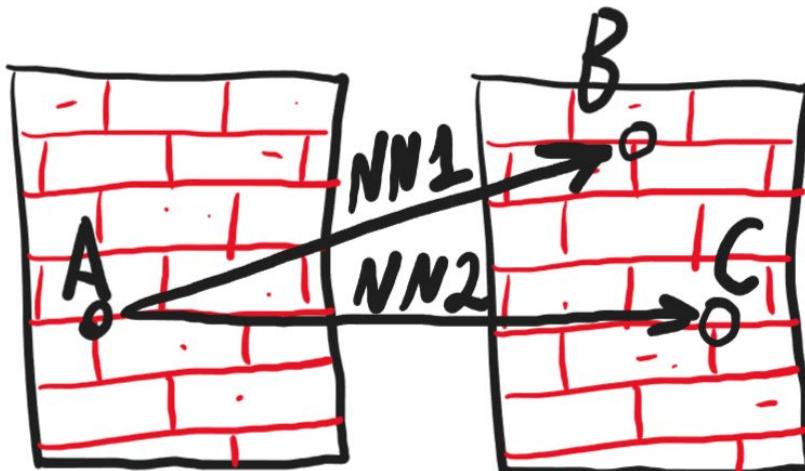
$$k = 0,9 \quad x < k \cdot y$$
$$x > k \cdot y$$

$$\text{dist}(A, B) = x$$

$$\text{dist}(A, C) = y$$

2. Сопоставление ключевых точек

- 1) **Nearest Neighbor (NN)**
- 2) Пусть мы искали KNN (**K=2 Nearest Neighbors**), т.е. два ближайших сопоставления.



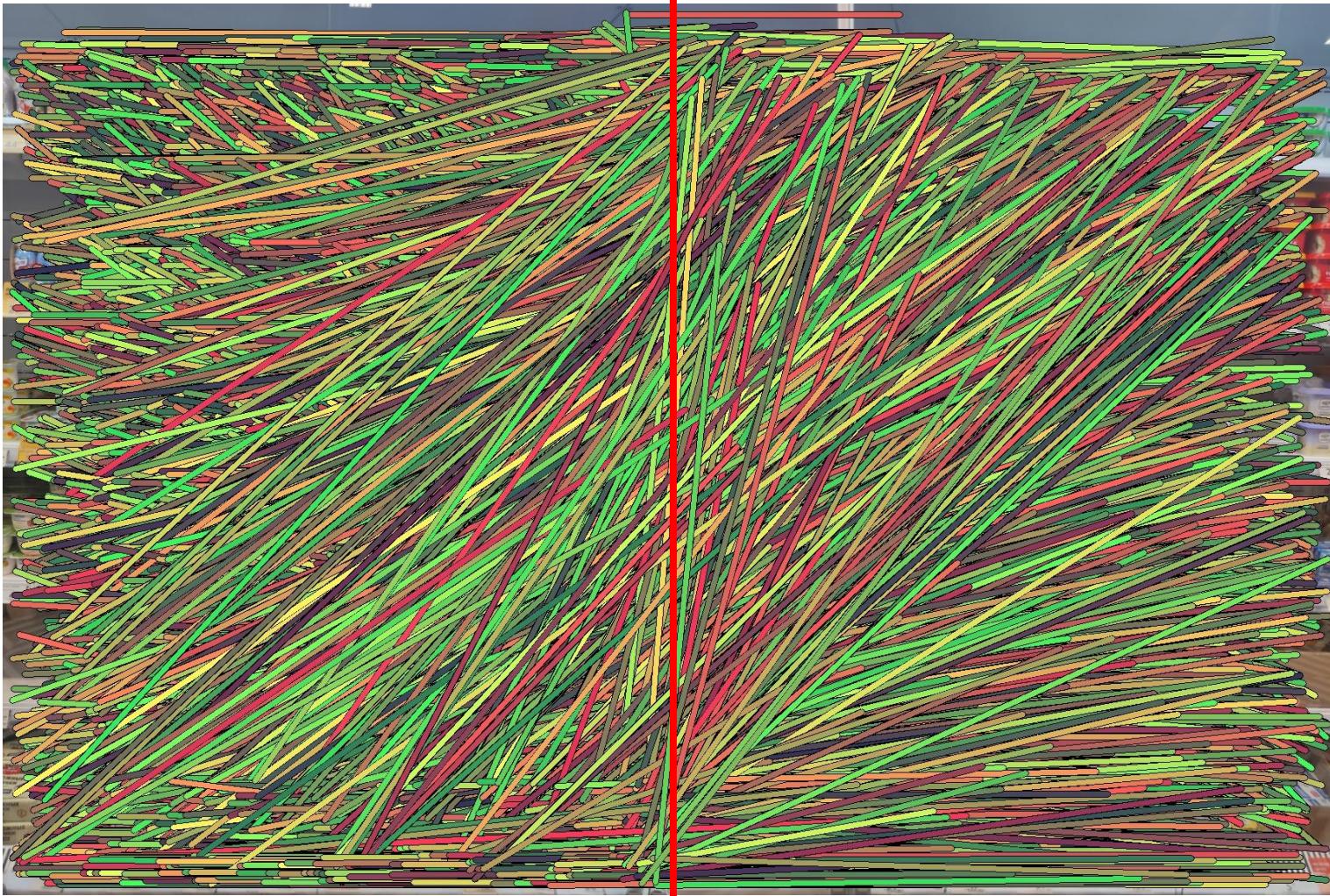
$k = 0,9$ $x < k \cdot y$ $x > k \cdot y$

$$\text{dist}(A, B) = x$$

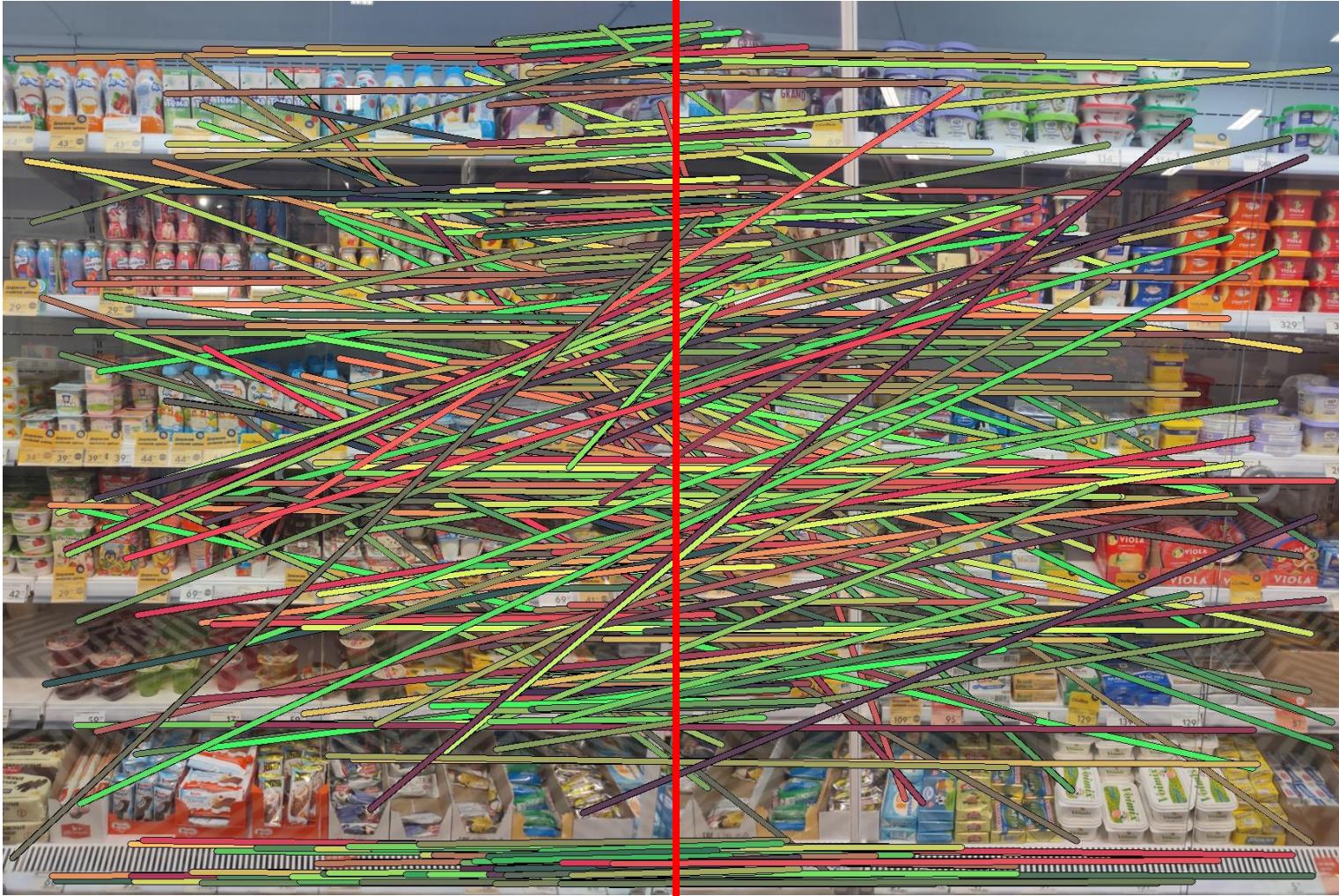
↗

$$\text{dist}(A, C) = y$$

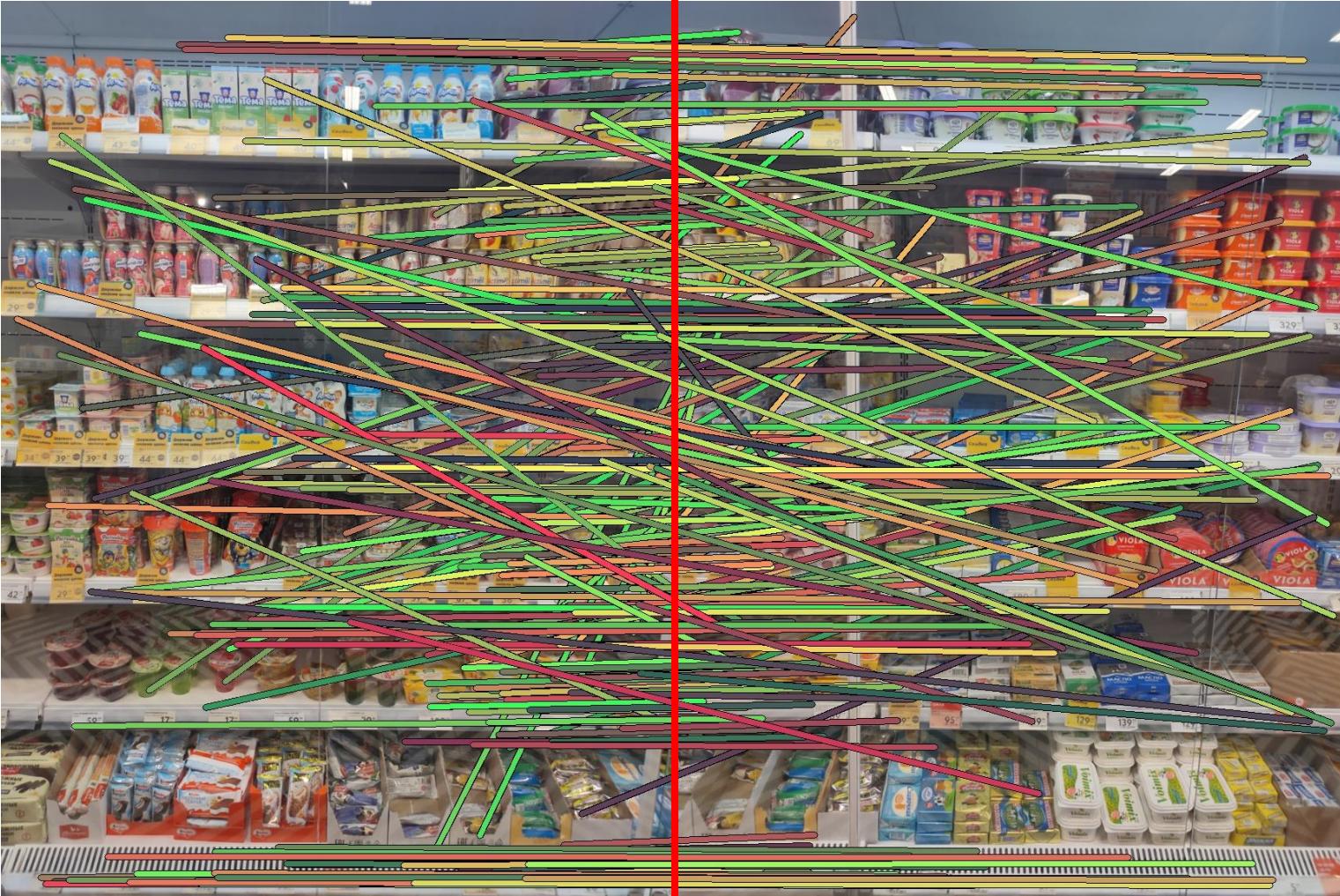
1. Nearest Neighbors



2. K-ratio test



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

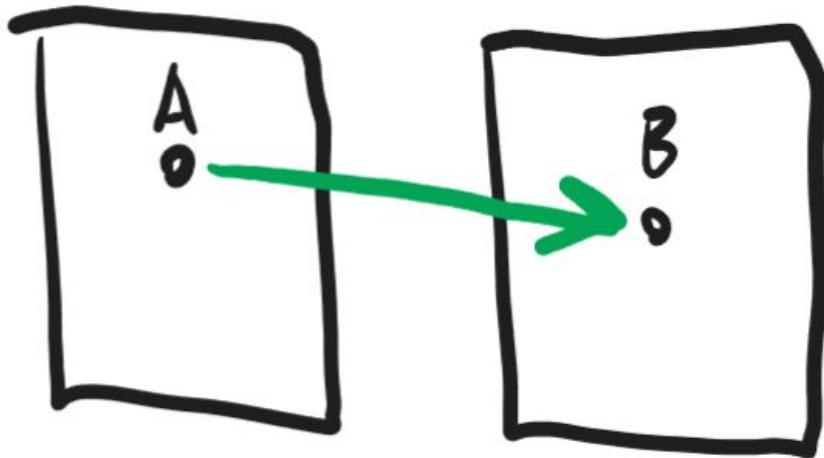


2. Сопоставление ключевых точек

- 1) **Nearest Neighbor (NN)**
- 2) **K-ratio test**
- 3) **А нельзя ли как-то воспользоваться тем что мы сопоставляем как слева направо, так и справа налево?**
Какие разумные требования можно предъявить к такой паре?

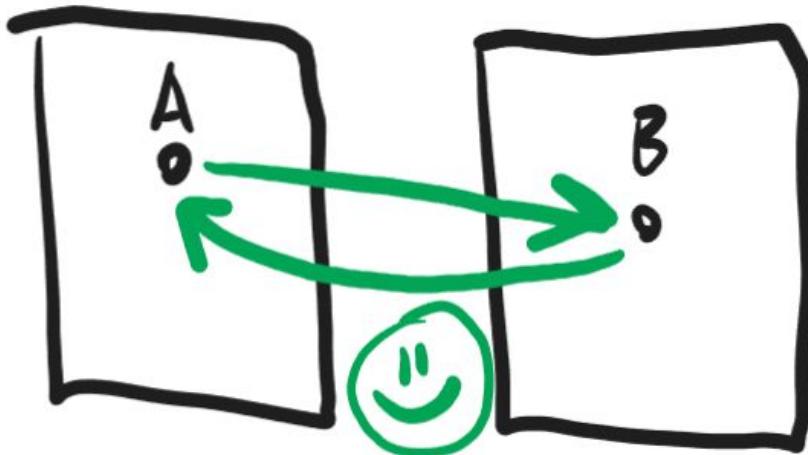
2. Сопоставление ключевых точек

- 1) Nearest Neighbor (NN)
- 2) K-ratio test
- 3) А нельзя ли как-то воспользоваться тем что мы сопоставляем как слева направо, так и справа налево?
Какие разумные требования можно предъявить к такой паре?



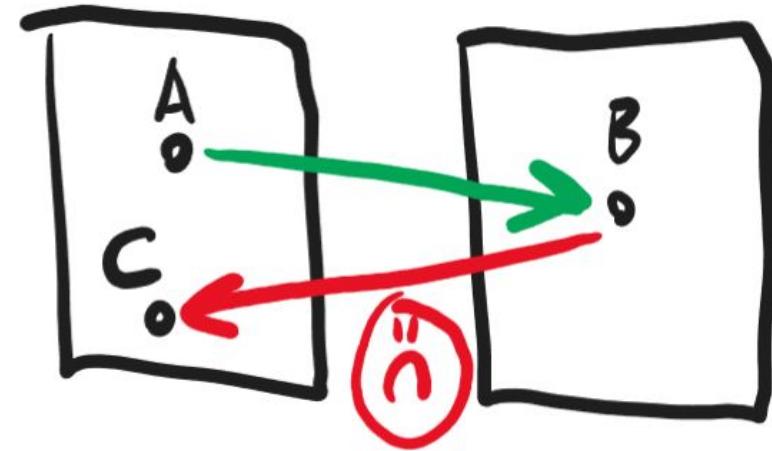
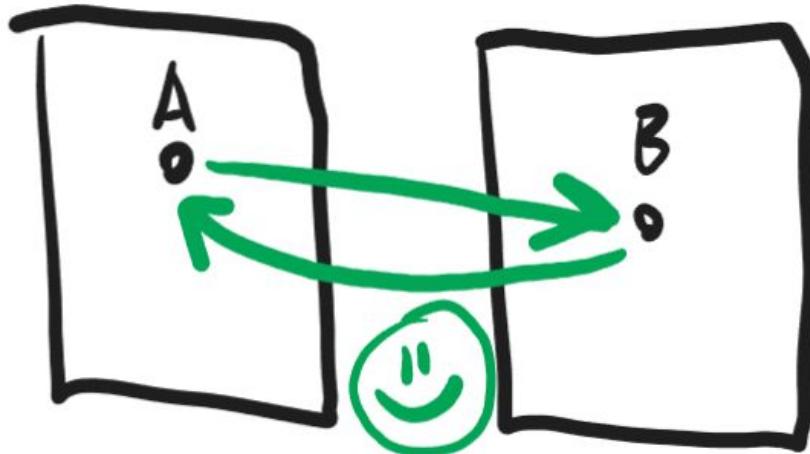
2. Сопоставление ключевых точек

- 1) Nearest Neighbor (NN)
- 2) K-ratio test
- 3) А нельзя ли как-то воспользоваться тем что мы сопоставляем как слева направо, так и справа налево?
Какие разумные требования можно предъявить к такой паре?



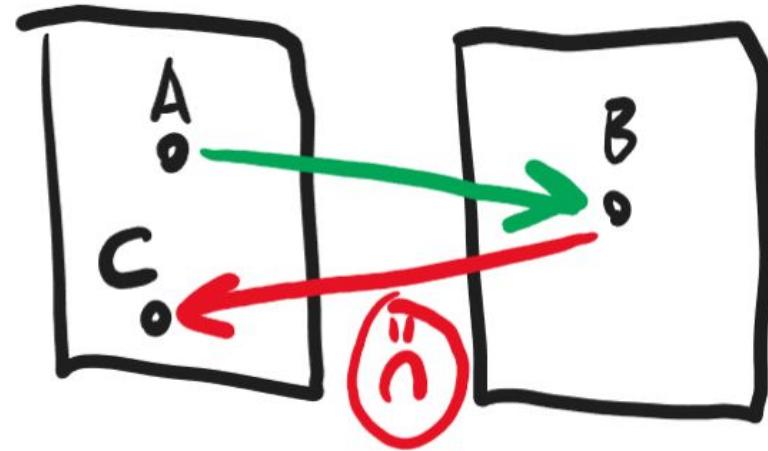
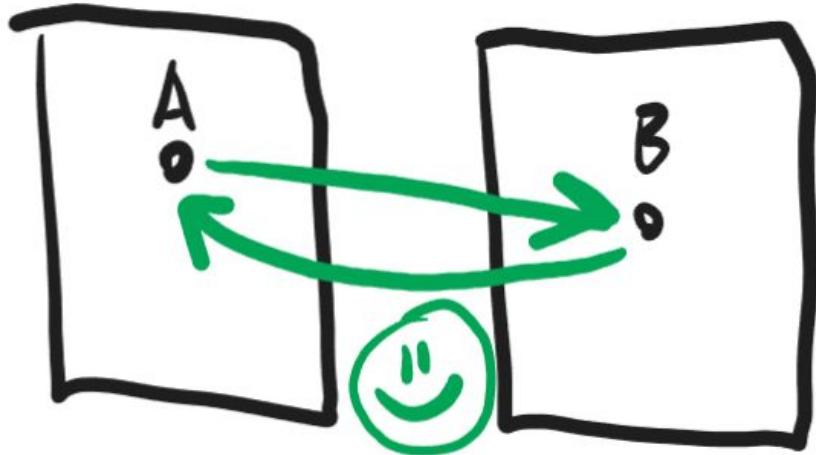
2. Сопоставление ключевых точек

- 1) Nearest Neighbor (NN)
- 2) K-ratio test
- 3) А нельзя ли как-то воспользоваться тем что мы сопоставляем как слева направо, так и справа налево?
Какие разумные требования можно предъявить к такой паре?

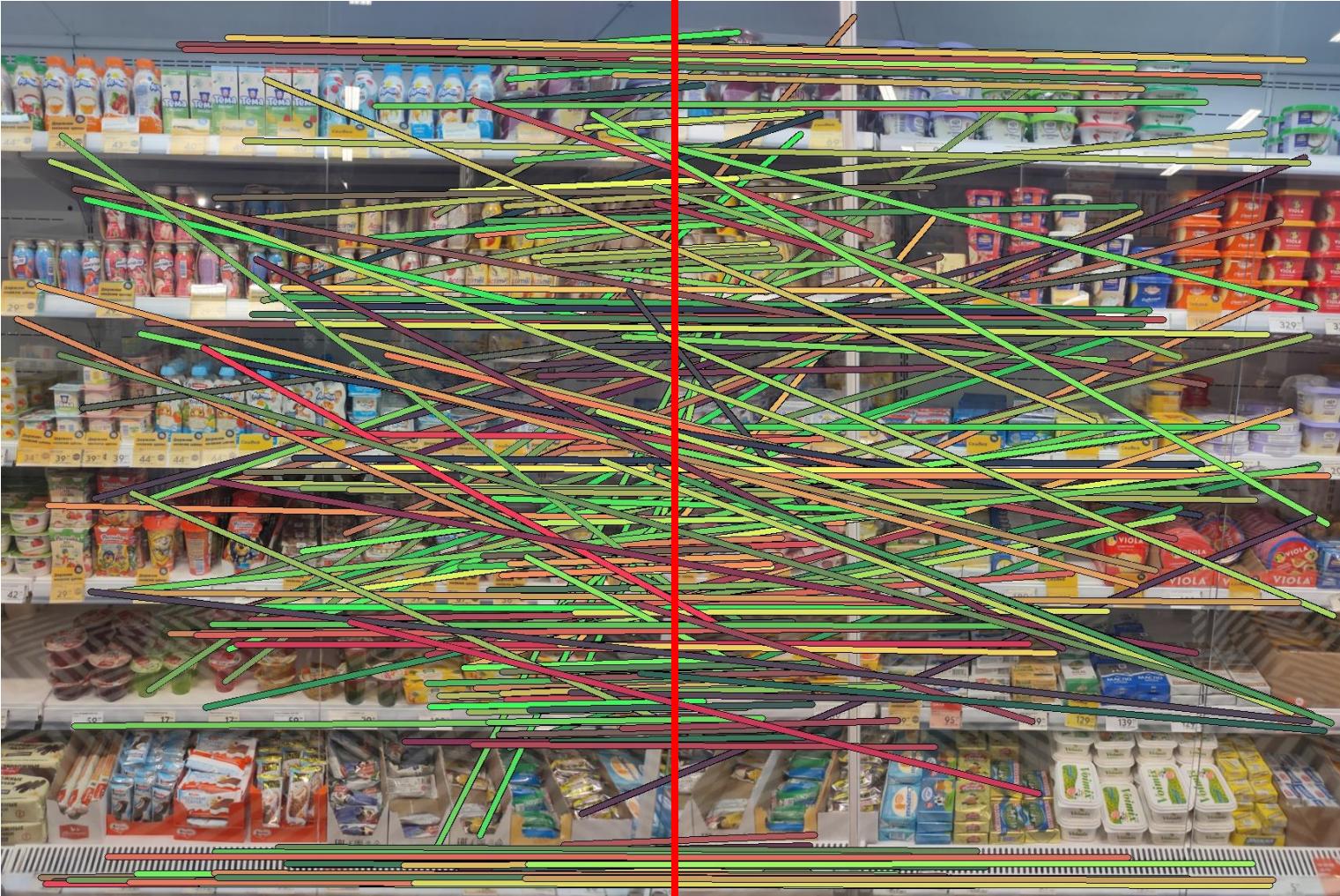


2. Сопоставление ключевых точек

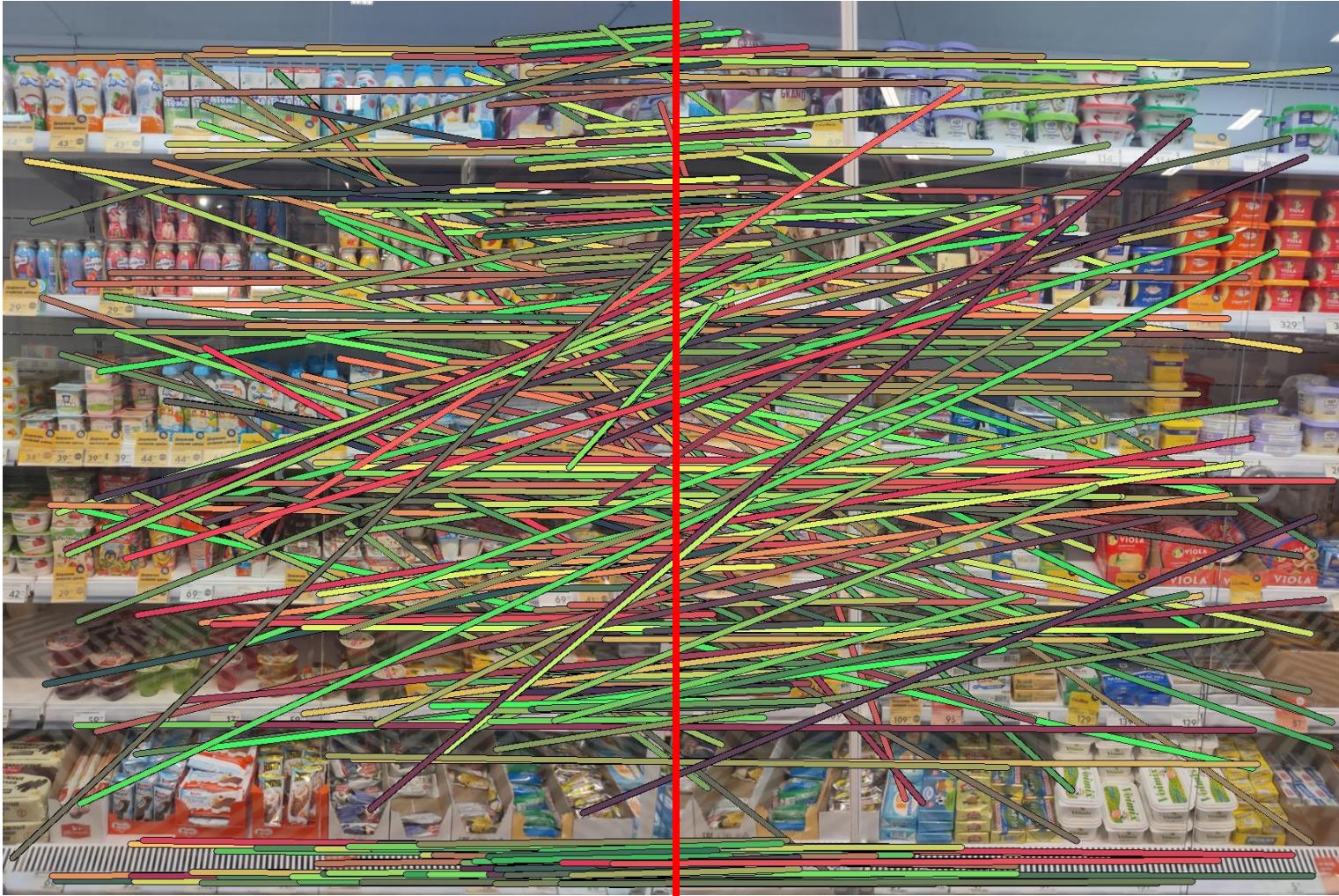
- 1) Nearest Neighbor (NN)
- 2) K-ratio test
- 3) Left-right check



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



2. K-ratio test

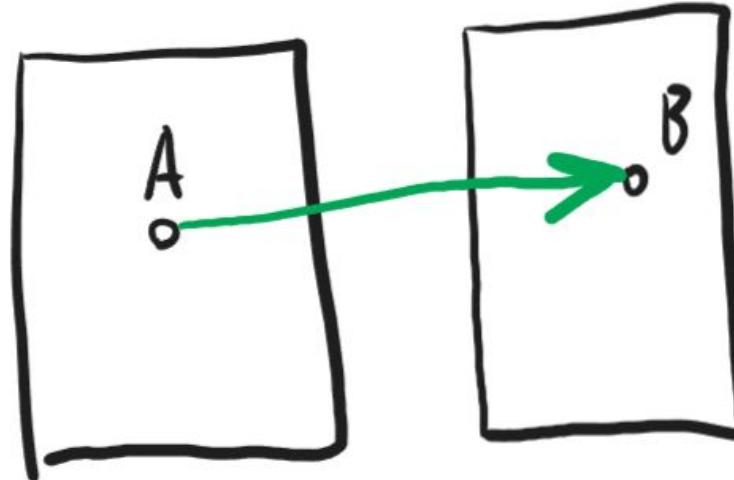


3. Left-right check



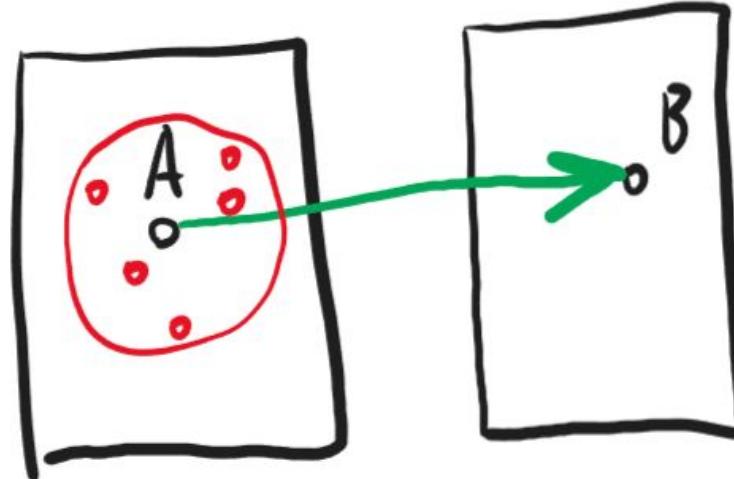
2. Сопоставление ключевых точек

- 1) Nearest Neighbor (NN)
- 2) K-ratio test
- 3) Left-right check
- 4) Какие другие требования разумности можно предъявить?



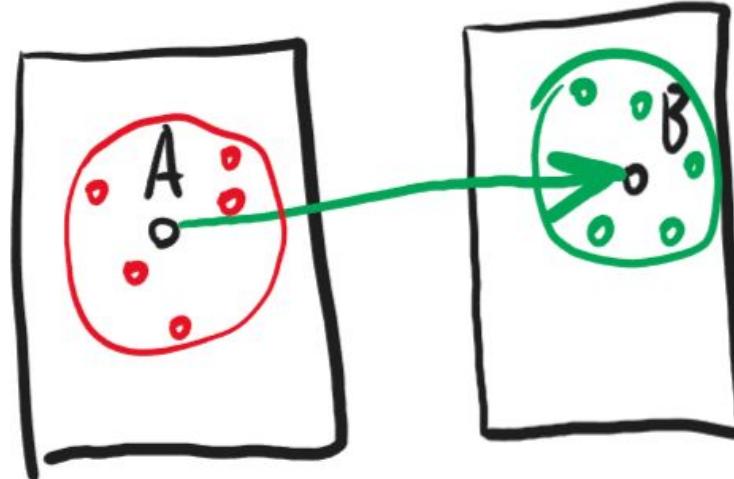
2. Сопоставление ключевых точек

- 1) Nearest Neighbor (NN)
- 2) K-ratio test
- 3) Left-right check
- 4) Какие другие требования разумности можно предъявить?



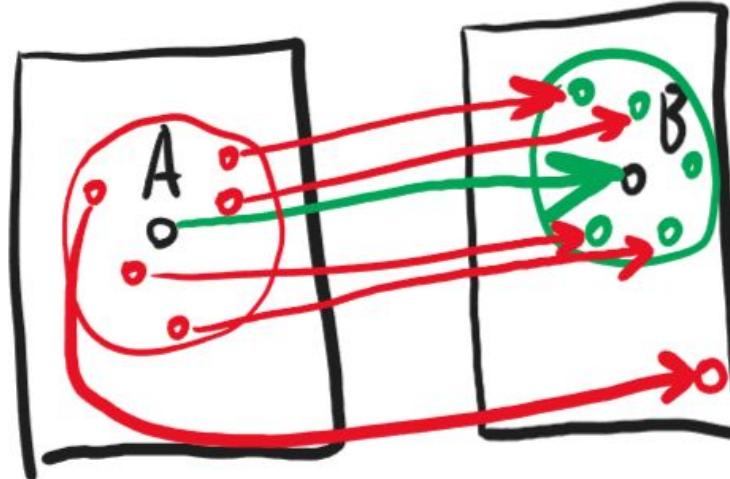
2. Сопоставление ключевых точек

- 1) Nearest Neighbor (NN)
- 2) K-ratio test
- 3) Left-right check
- 4) Какие другие требования разумности можно предъявить?



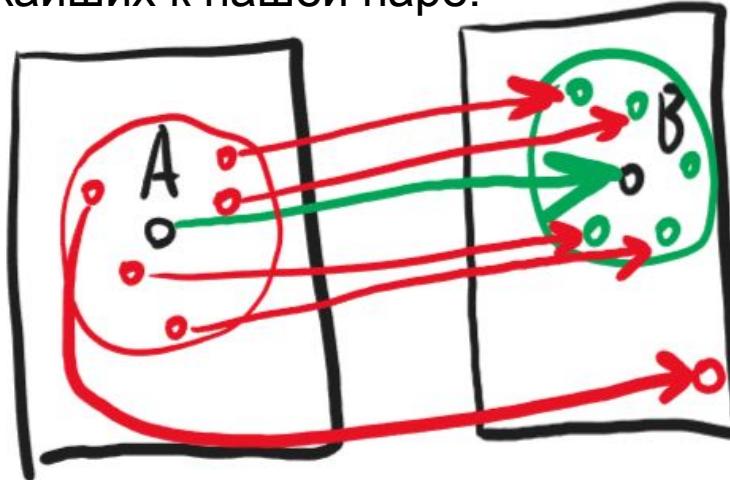
2. Сопоставление ключевых точек

- 1) Nearest Neighbor (NN)
- 2) K-ratio test
- 3) Left-right check
- 4) Какие другие требования разумности можно предъявить?



2. Сопоставление ключевых точек

- 1) **Nearest Neighbor (NN)**
- 2) **K-ratio test**
- 3) **Left-right check**
- 4) **Cluster filtering:** из k ближайших к нам **больше половины** парных должно быть среди k ближайших к нашей паре.



3. Left-right check



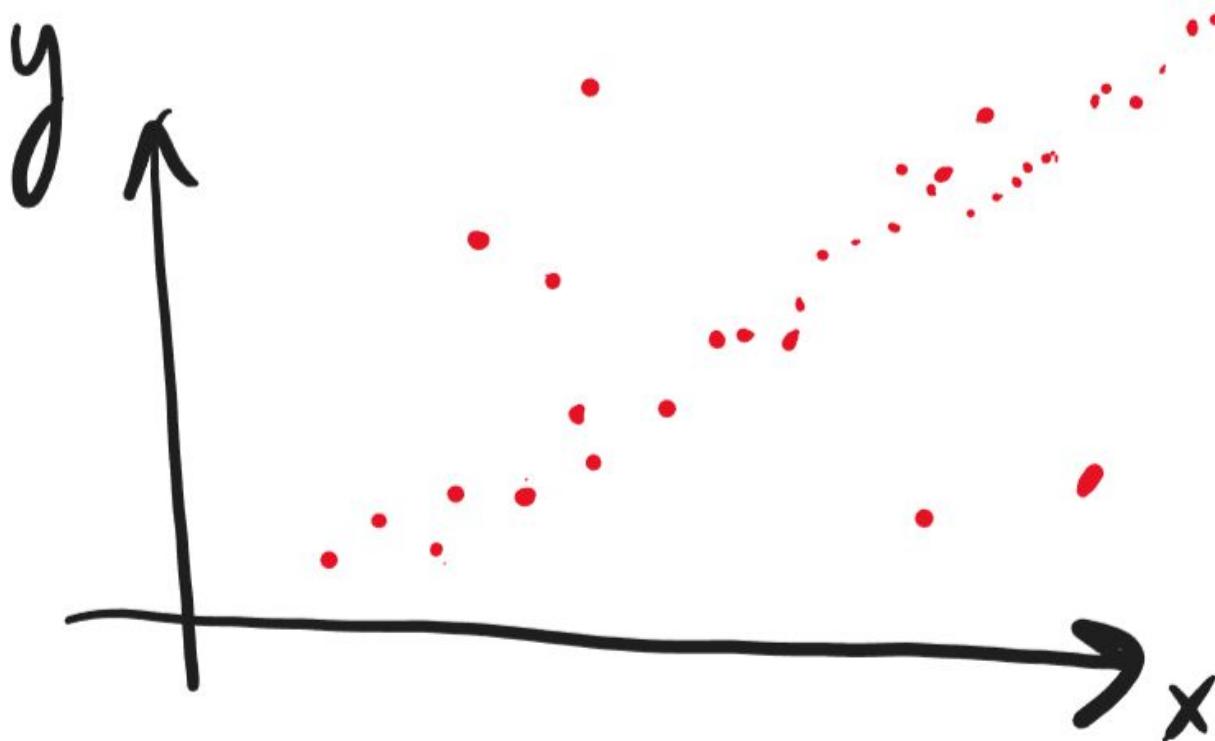
4. Cluster filtering



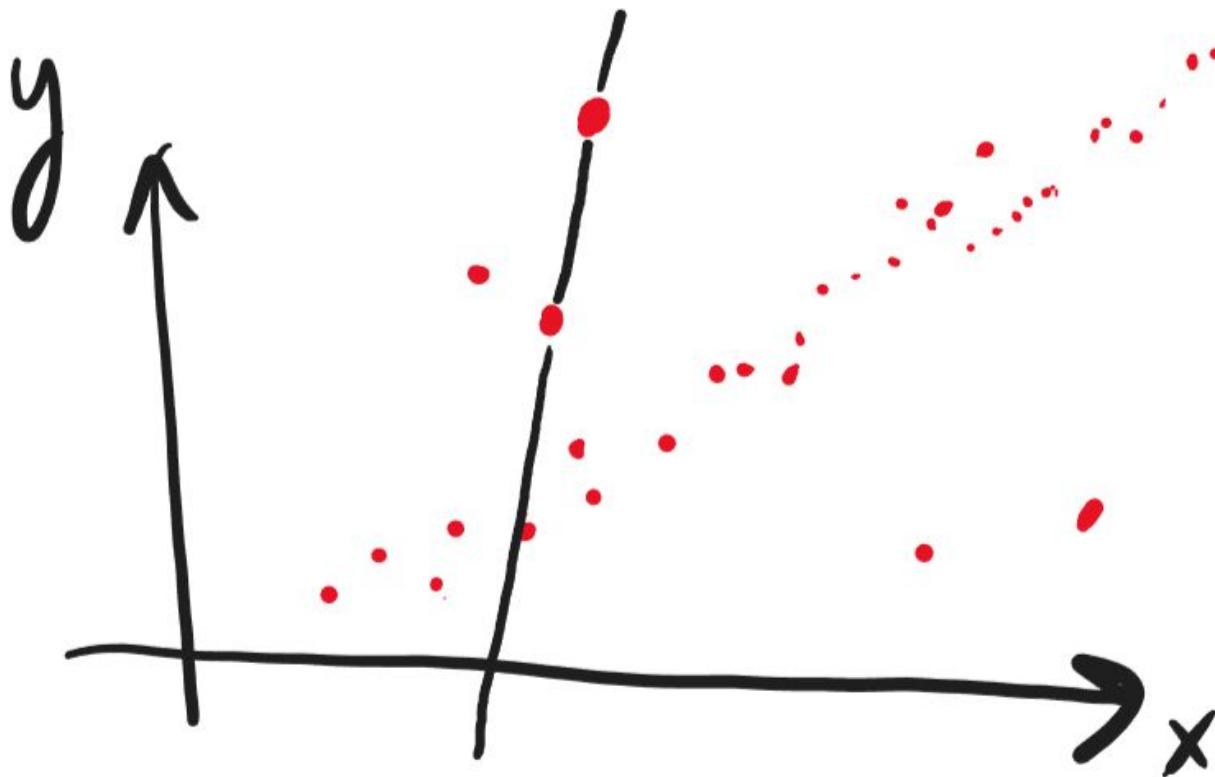
2. Сопоставление ключевых точек

- 1) **Nearest Neighbor (NN)**
- 2) **K-ratio test**
- 3) **Left-right check**
- 4) **Cluster filtering**
- 5) **А для чего мы все это задумали? Для наложения картинок!**

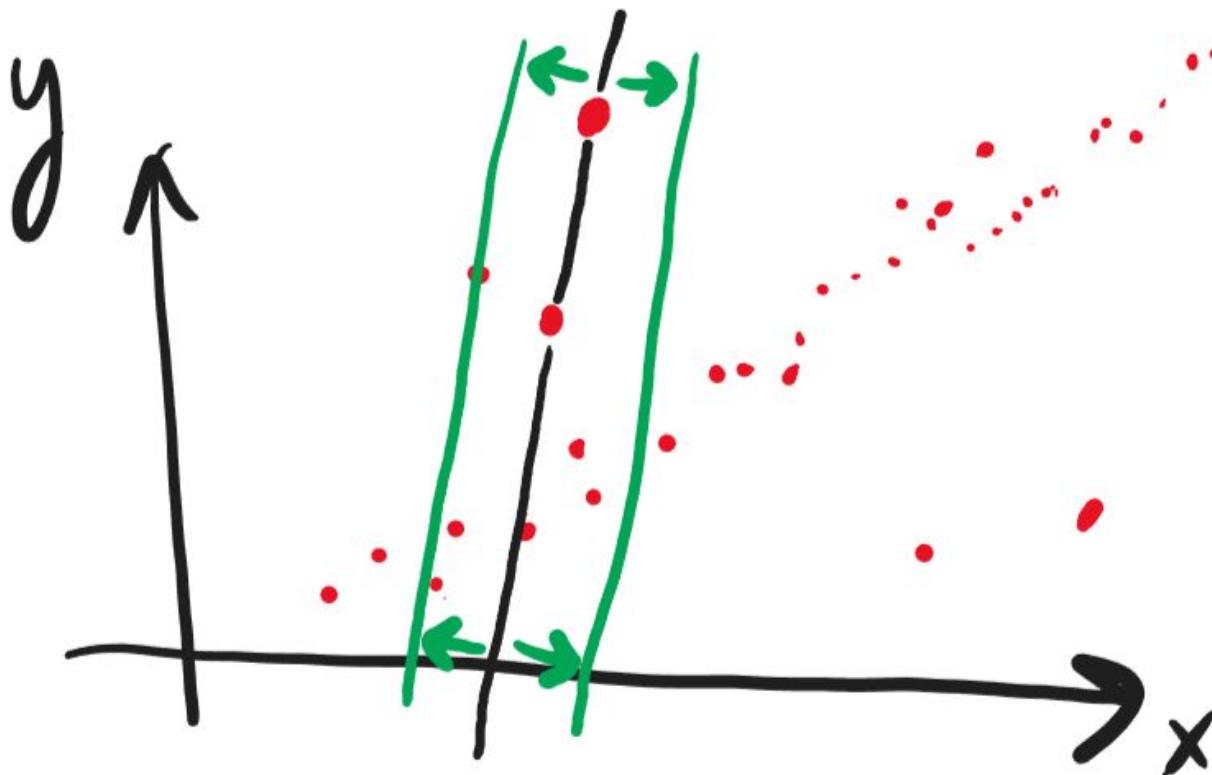
RANSAC для поиска прямой среди точек



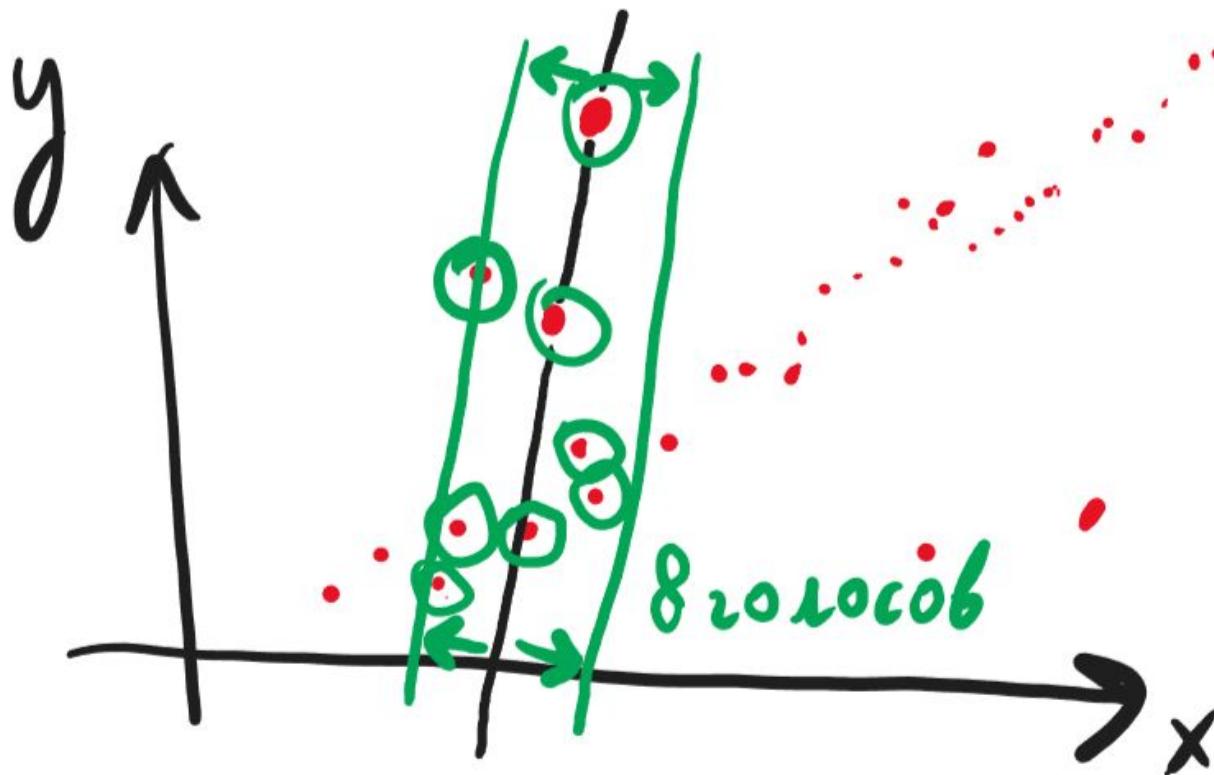
RANSAC для поиска прямой среди точек



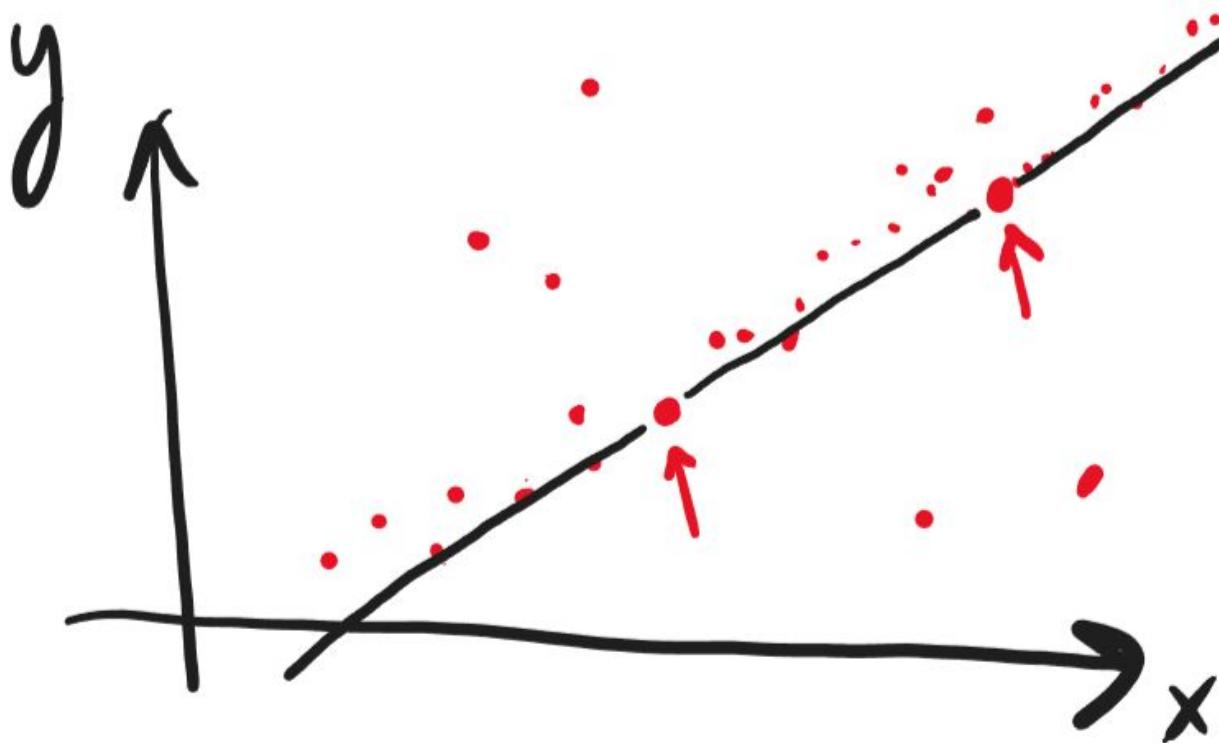
RANSAC для поиска прямой среди точек



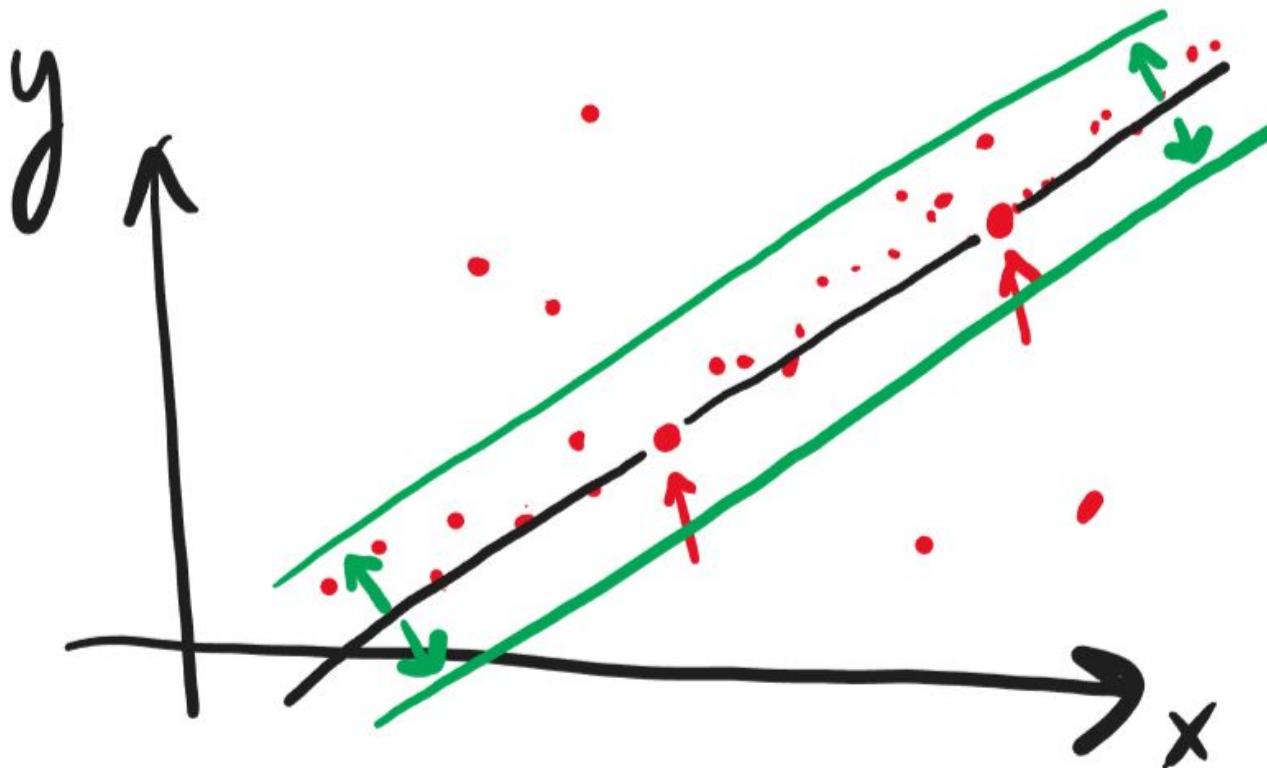
RANSAC для поиска прямой среди точек



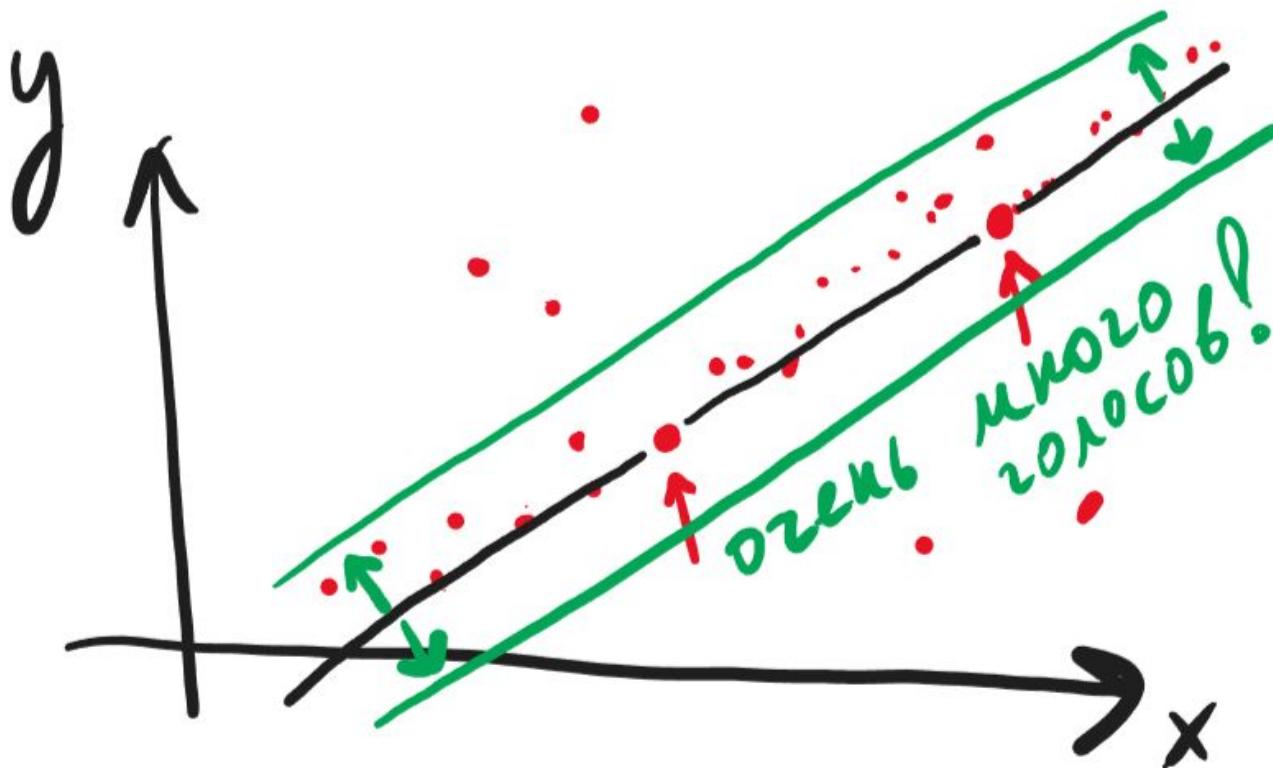
RANSAC для поиска прямой среди точек



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

1000 итераций делаем:

A, B = randomSamples(points)

RANSAC для поиска прямой среди точек

```
result_line = None
```

```
result_score = 0
```

1000 итераций делаем:

```
A, B = randomSamples(points)
```

```
line = fit(A, B)
```

RANSAC для поиска прямой среди точек

```
result_line = None
```

```
result_score = 0
```

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
```

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

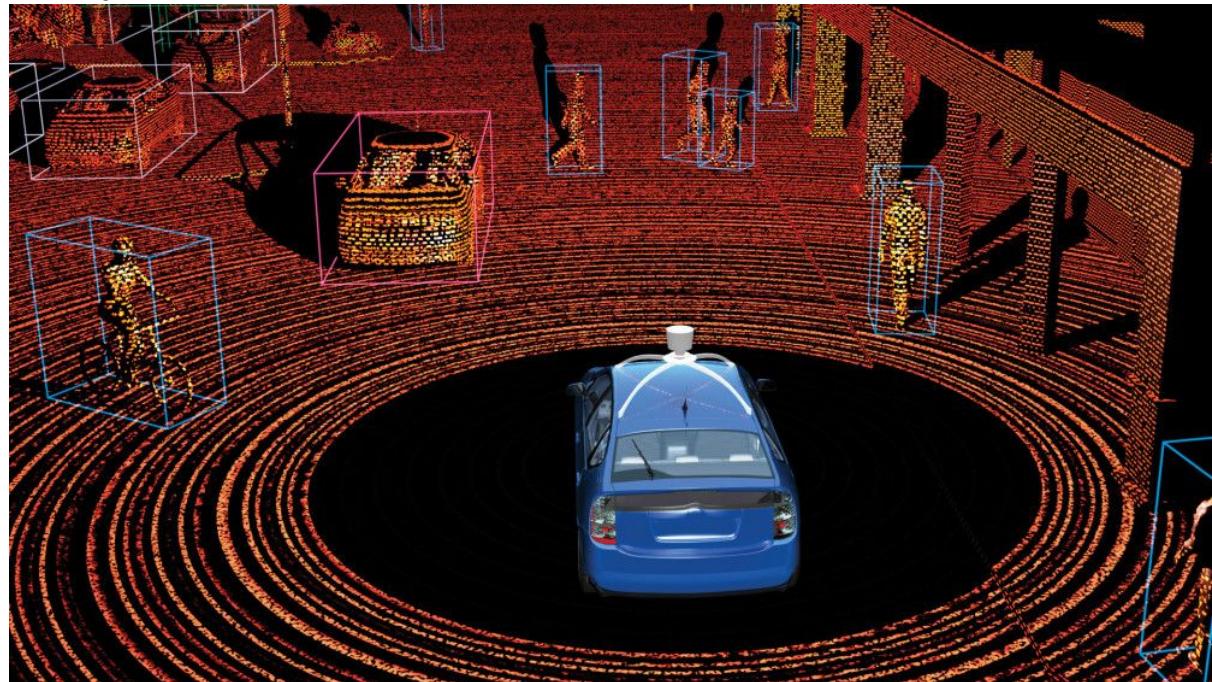
- 1) А что изменится если бы мы искали параболу?
(много точек лежащих около какой-то одной параболы + много выбросов)

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 что 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:



2. Сопоставление ключевых точек

- 1) **Nearest Neighbor (NN)**
- 2) **K-ratio test**
- 3) **Left-right check**
- 4) **Cluster filtering**
- 5) **А для чего мы все это задумали? Для наложения картинок!**
Как сюда применить RANSAC?

4. Cluster filtering



5. RANSAC



Что на следующей лекции?

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

Вопросы?



Полярный Николай

polarnick239@gmail.com