ИТМО

Поиск изображений

Техническое зрение

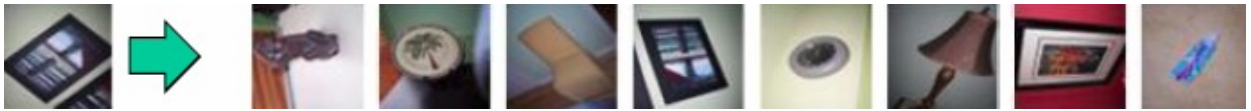
Поиск изображений

Задача

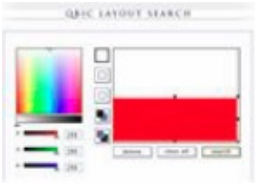
- Поиск изображений с определенным содержанием в базе изображений.
- Схоже с распознаванием, но фокус на масштабировании.

Формирование запроса на поиск

1. Текстовый запрос (аннотация изображения) – требуется категоризация.
2. Изображение-пример (найти такое же).

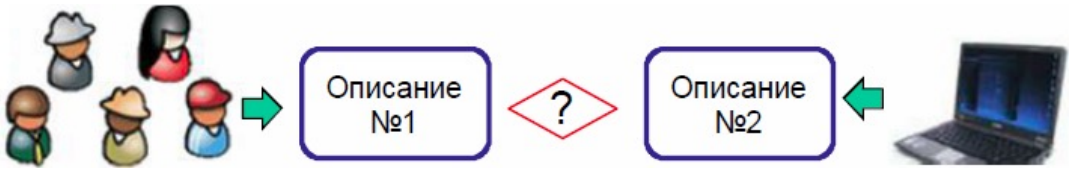


3. Запрос в виде характеристики содержимого (гистограмма цветов, например).



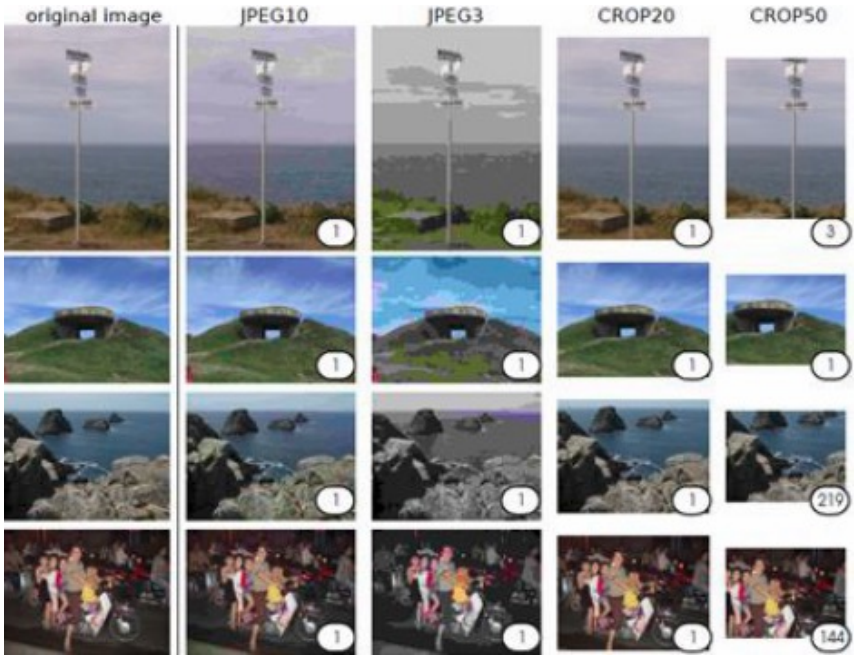
«Похожее изображение» – что это?

- **Семантический разрыв (semantic gap)** – несовпадение информации, которую можно извлечь из визуальных данных, и интерпретацией этих данных со стороны пользователя.



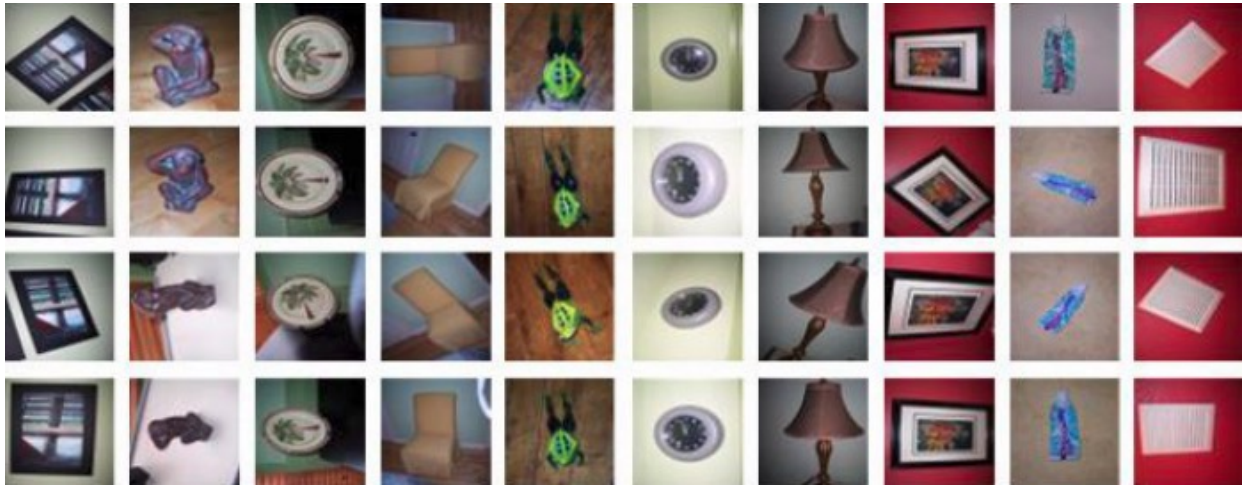
Похожее 1: Полудубликаты

- Near-duplicates
- Немного измененная версия изображения (ракурс, цвета, размер, и т.д.).



Похожее 2: Тот же объект или сцена

- Object retrieval
- Сильные вариации ракурсов, фона и других изменений по сравнению с полудубликатами.



Похожее 3: Похожие сцены по конфигурации

- Могут быть разными по назначению.

Кухня

Приемная

Бар

Автобус

Самолет

Зал

Похожее 4: Изображения из одного класса

- Category-level classification.
- Сцены или объекты.

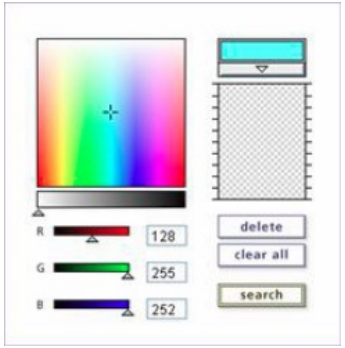


Банкетный зал

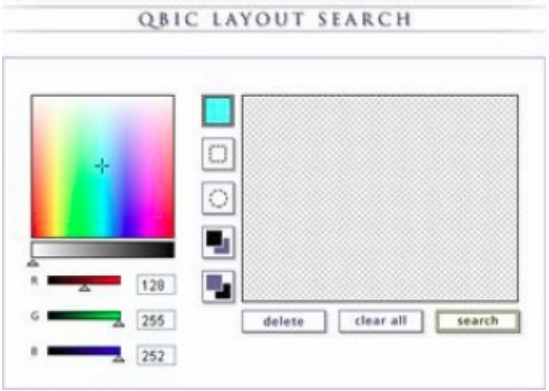
Постановка задачи

- Все 4 задачи имеют разные постановки.
- Для их решения требуются разные алгоритмы.
- Рассмотрим самую первую систему поиска по содержанию изображения – **QBIC «Query By Image Content»** (1995):
  - Вычисляет наборы признаков объектов:
    - Цветовые гистограммы признаков;
    - Площадь, периметр и др.
  - Для описания объектов используется бинарная маска.
  - Сегментация объектов проводится вручную или автоматически:
    - Выделение контрастных объектов на фоне (музейные экспонаты);
    - Заливка для автоматизированной разметки.
  - В базе изображений порядка 10 000 штук.

Пример работы QBIC

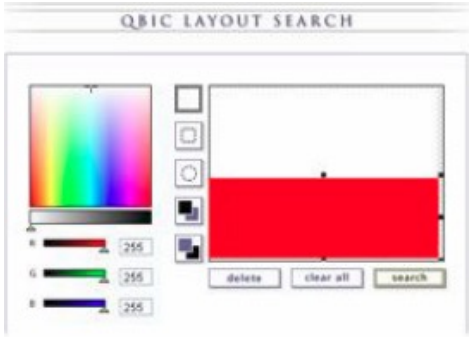


Гистограмма



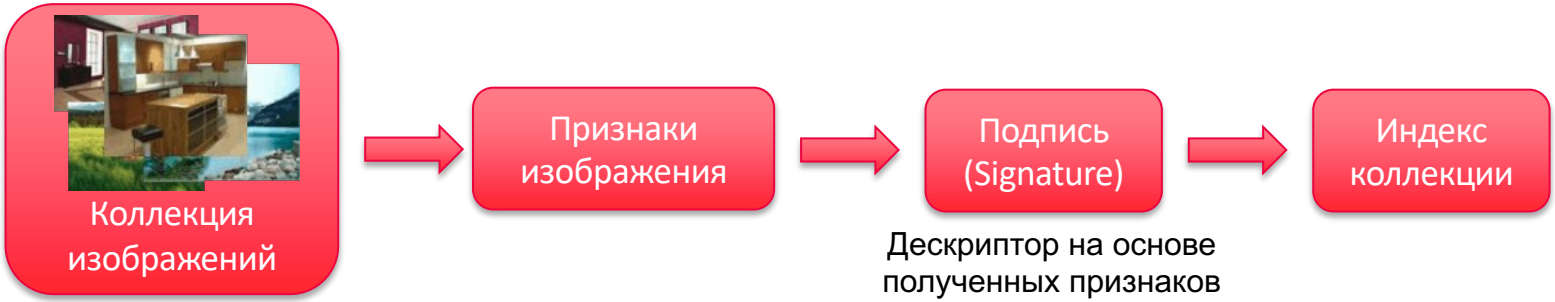
Пространственное распределение цветов

Пример работы QBIC

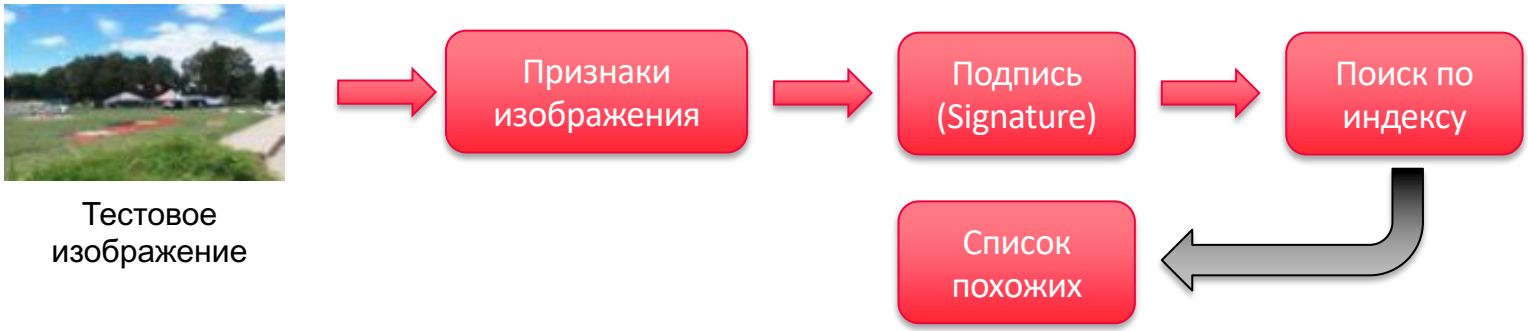


Общая схема поиска изображений

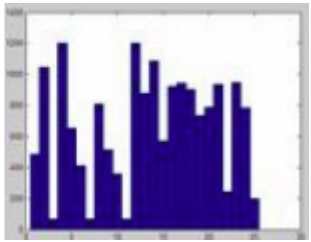
1. Построение индекса



2. Тестирование



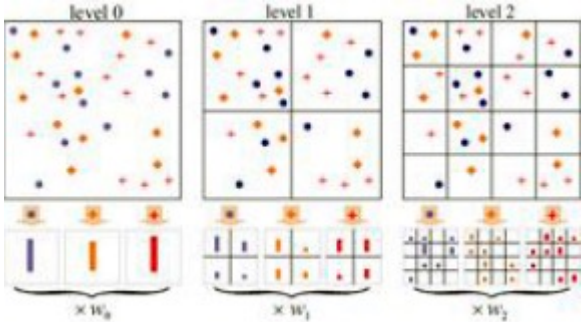
Дескрипторы

Гистограммы цветов


Гистограммы градиентов (GIST / HOG)


Мешок слов



Мешок слов по пространству
и отдельным особенностям

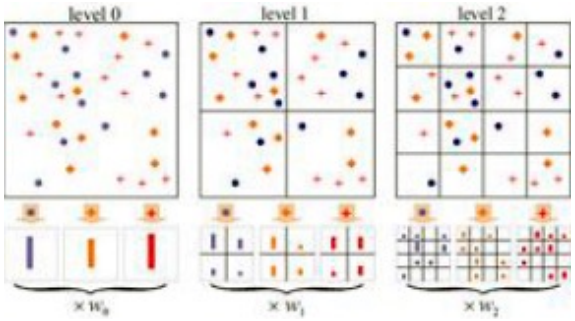
Ресурсы для решения задачи

ИТМО

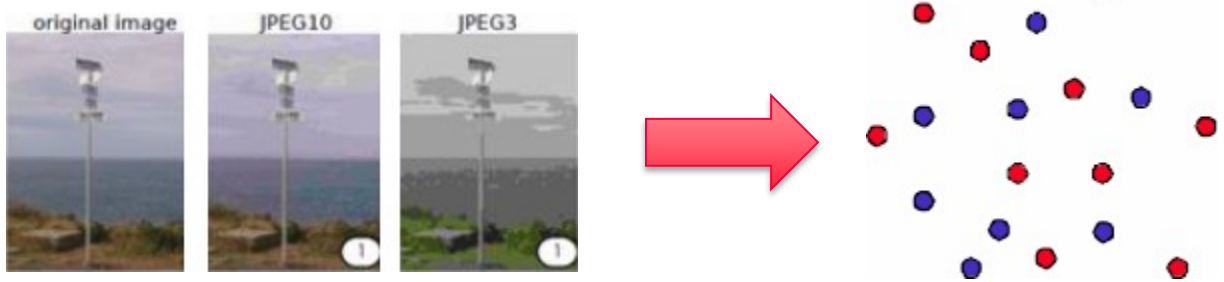
- Желаемое число изображений в коллекции: миллиарды.
- Пример – дескриптор GIST:
  - Решетка 4 x 4 * 8 ориентаций * 4 масштаба = 512 параметров;
  - При 4-х байтах на параметр: 2048 байт;
  - При коллекции в 1 млн. изображений: дескрипторы 2 Гбайта.
- Вывод: простой дескриптор требует большого объема памяти.

Ресурсы для решения задачи

- Пример – мешок слов:
  - Размер словаря от 200 до 1 млн слов;
  - До 1 млн параметров в одной гистограмме;
  - До 4 Мб на одно изображение.
- Пример – мешок слов с пирамидой:
  - Пирамида из трех уровней – 21 гистограмма;
  - Мешок слов * 21 = до 80 Мб на одно изображение.
- Пример – фильтры объектов:
  - 200 основных классов;
  - Пирамида из трех уровней – 21 гистограмма;
  - 21 * 200 * 4 Байта = 16 килобайт;
  - 1 млн изображений: дескрипторы 16 Гб.

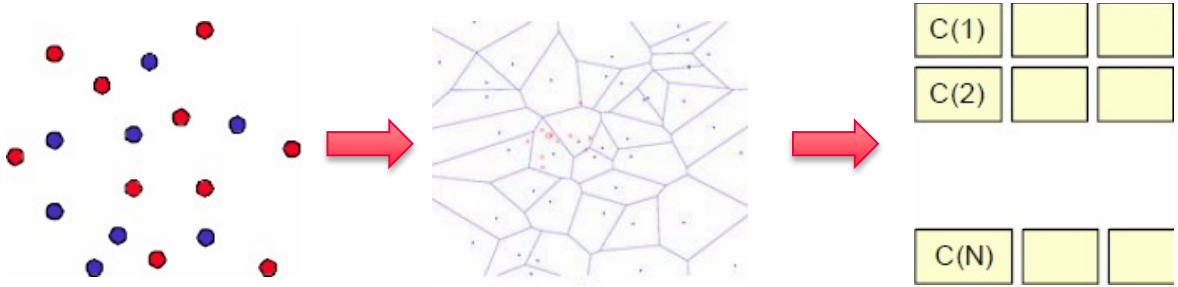


Поиск ближайшего изображения



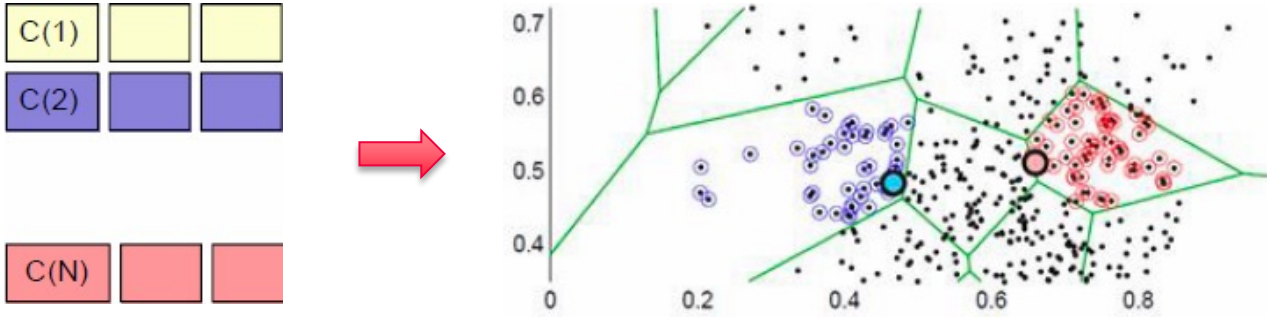
- Необходимо найти ближайших соседей по дескриптору во всей коллекции (Nearest Neighbor)
- Простейший индекс – линейный список всех дескрипторов
- Полный поиск – сравнение тестового вектора с каждым примером из коллекции
  - C = 1M изображений, GIST = 512 параметров, C * GIST = 512M операций
- Нужны приближенные методы поиска соседей
  - Approximate nearest neighbor

Инвертированный индекс



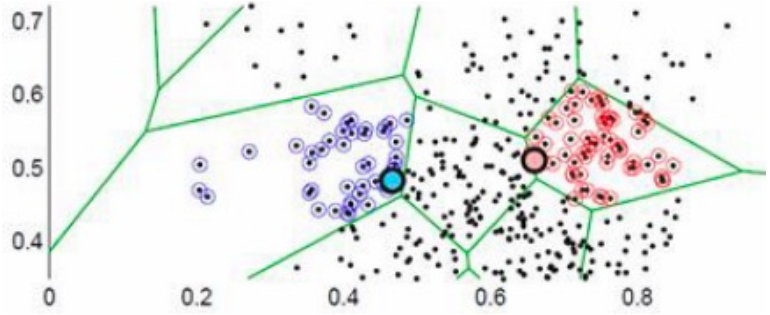
- Кластеризуем все дескрипторы, получим K-кластеров;
- Разделим всю коллекцию по кластерам;
- Составим «инвертированный индекс»:
  - Список кластеров (с приписанными GIST);
  - В каждом кластере – номера изображений, которые относятся к кластеру.

Инвертированный индекс



- Поиск по индексу:
  - Просмотрим инвертированный индекс, найдём ближайший кластер;
  - Все элементы в списке – ближайшие вектора (приближенно).

Ранжирование результатов



- Для повышения точности будем выдавать результаты из кластера упорядоченно:
  - Рассчитаем расстояния до каждого элемента списка по полному дескриптору;
  - Упорядочим результаты (re-ranking) по близости (первые – ближайшие).

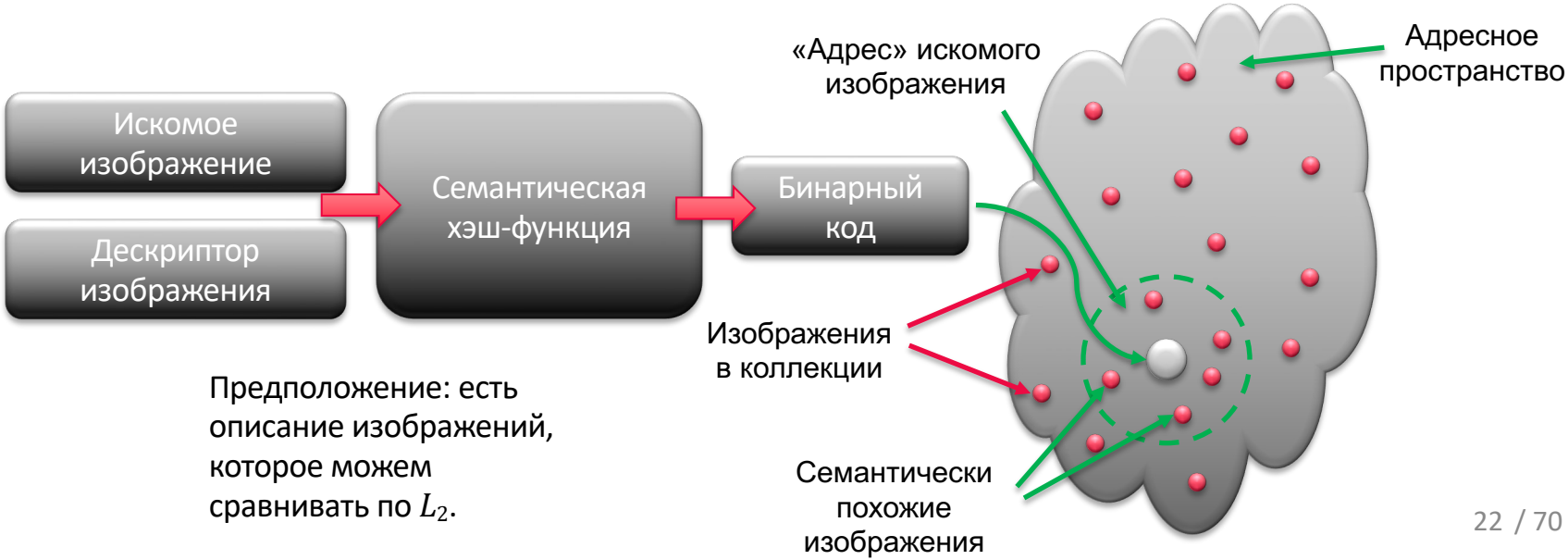
Простой метод

ИТМО

- Индексирование:
  - Вычисляем GIST для каждого изображения;
  - Кластеризуем дескрипторы, получаем 200 кластеров;
  - Строим инвертированный индекс по кластерам;
  - В памяти храним весь GIST.
- Поиск по коллекции:
  - Вычисляем GIST для изображения;
  - Ищем ближайший кластер в инвертированном индексе, сравнивая по GIST;
  - Сортируем список из индекса по GIST.

Семантическое хеширование

- Позволяет ускорить простой алгоритм и увеличить размер коллекции.
- Идея: построить такие короткие бинарные подписи для изображений, что для близких в L_2 изображений они будут близки.



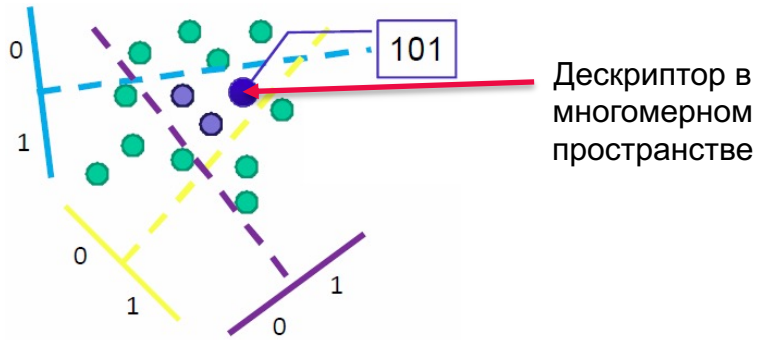
Предположение: есть описание изображений, которое можем сравнивать по L_2.

Формализация

- Имеются дескрипторы x, y (векторы);
- Требуется получить бинарный код $h(x)$, причем для поиска изображений: $h(x) \approx h(y)$, где $h(x)$ – семантическая хэш-функция (*бинарная подпись*).

Locality Sensitive Hashing

- LSH, Хеширование с учетом местоположения;
- Возьмем случайную проекцию данных на прямую;
- Случайно выберем порог, пометив проекции 0 или 1 (1 бит подписи);
- С увеличением числа бит подпись приближает L_2 – метрику в исходных дескрипторах.



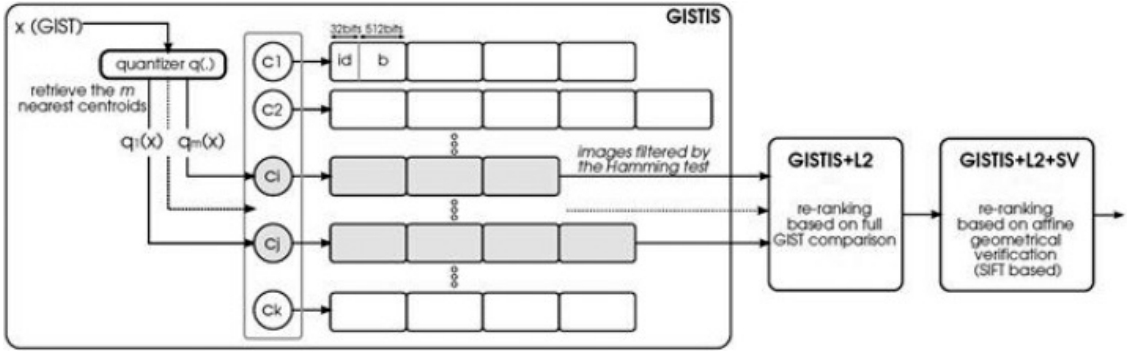
- Недостатки:
  - Приближение L_2 – асимптотическое.
  - При реализации может потребоваться слишком много бит для подписи.

GIST Indexing Structure (GISTIS)

ИТМО

- Строим GIST для каждого изображения;
- Кластеризуем все дескрипторы с помощью метода k-средних на $k = 200$ слов;
- Для каждого кластера считаем с помощью LSH бинарную подпись;
- Идентификатор картинки и бинарная подпись (512 бит) хранится в индексе в оперативной памяти (RAM);
- GIST хранится на жестком диске;
- Можем проводить сортировку несколько раз:
  - Сначала по бинарным подписям,
  - Затем по GIST с жёсткого диска.

Схема работы



- Результаты

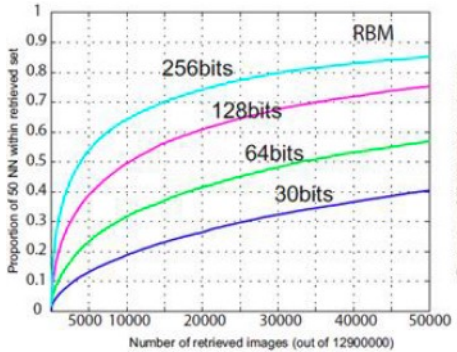
Эксперименты на 110M изображений

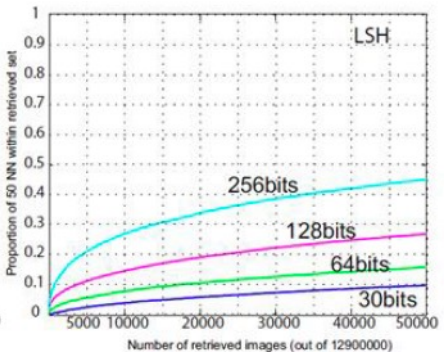
method	bytes (RAM) per image	time per query image *fingerprint*	*search*
SV [11]	501,816	440 ms	13 h
HE [5]	35,844	780 ms	96 ms
BOF	11,948	775 ms	353 ms
GHE	35,844	780 ms	47 ms
GBOF	11,948	775 ms	67 ms
GIST	3840	35 ms	1.26 s
GISTIS	68	36 ms	2 ms
GISTIS+L2	68	36 ms	6/192 ms

Сравнение

- Сравнение методов на основе обучения (RBM) и LSH при различной длине кодов:

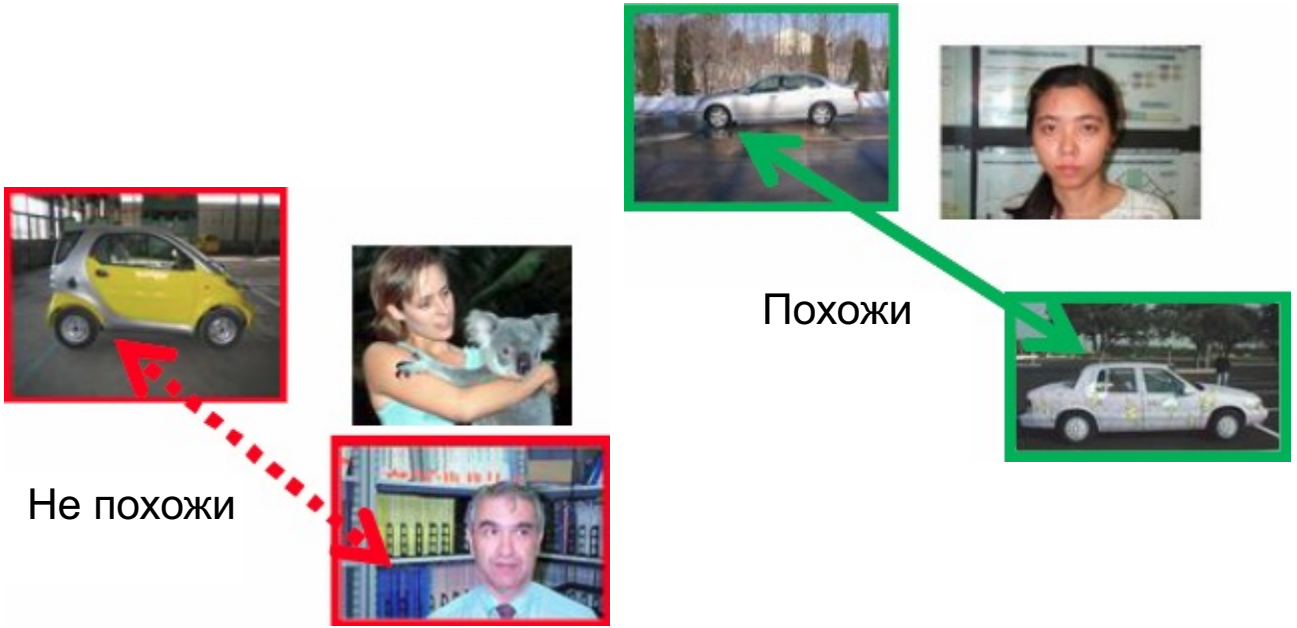
Сравнение на 12.9M изображений





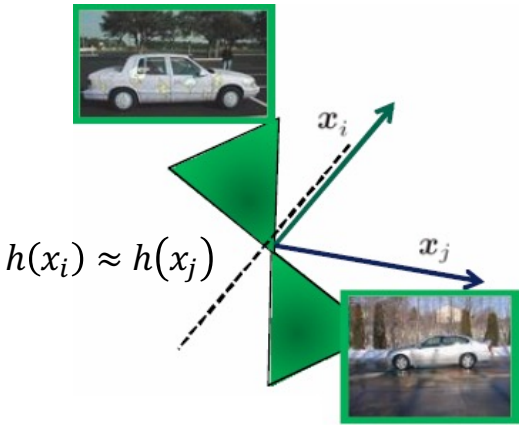
Обучаемые метрики

- В случае, если Евклидова метрика L_2 не подходит и сложно выбрать правильную метрику, то ее можно обучить.

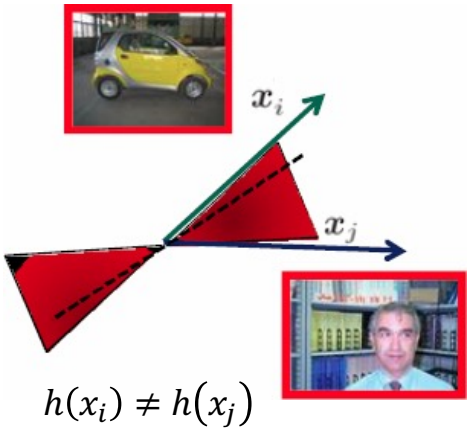


Обучаемые метрики

- Обучение расстояния через LSH:



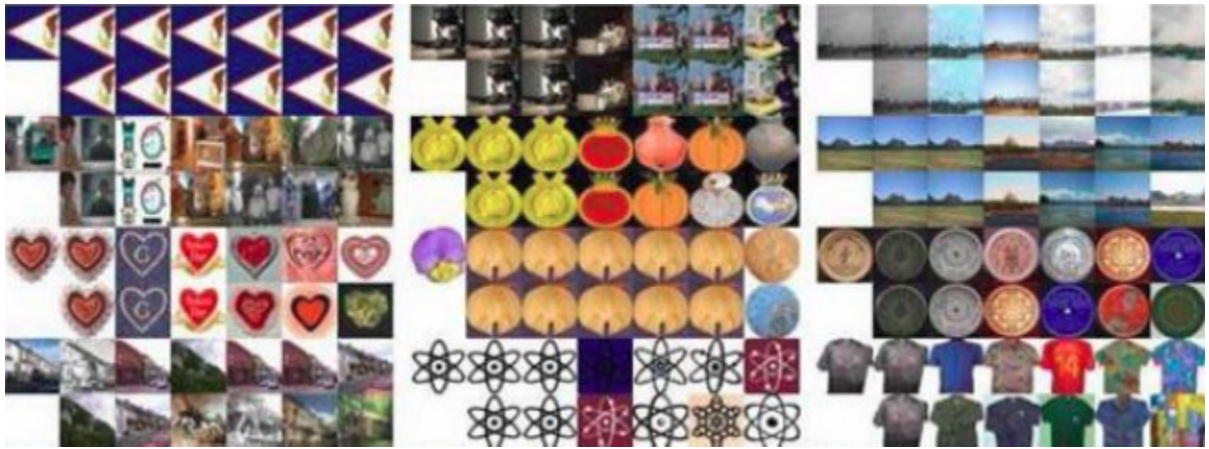
С меньшей вероятностью разбиваем подобные пары с ограничением сходства



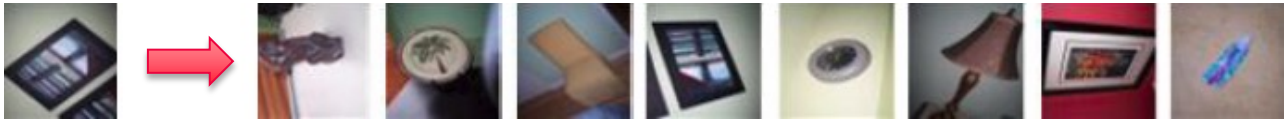
С большей вероятностью разобьем пары с ограничением несходства

Результат

- Сравнение на 80 млн маленьких картинок.
- Обученная метрика позволяет найти те же результаты, но просмотрев меньше 1% базы.
- Скорость – 0.5 с вместо 45 с.

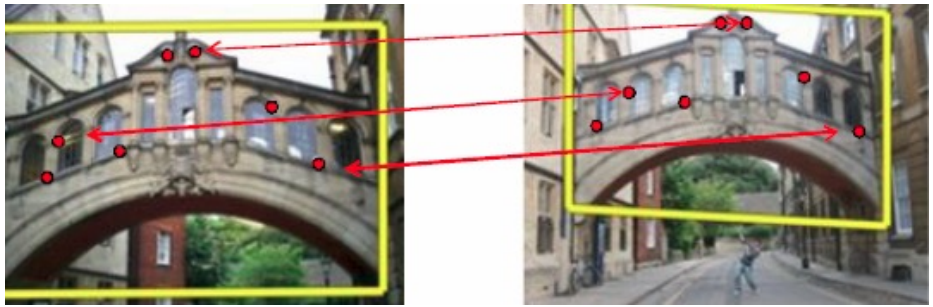


Поиск объектов или сцен



Поиск объектов или сцен

- Простейший способ:
  - Найдем локальные особенности (Harris, SIFT, SURF);
  - Вычислим дескрипторы для всех точек (SIFT);
  - Сопоставим точки по дескрипторам;
  - Вычислим робастным методом (RANSAC) преобразование, отфильтруем ложные соответствия;
  - Если соответствий больше K, тогда изображения считаем похожими.



Поиск объектов или сцен

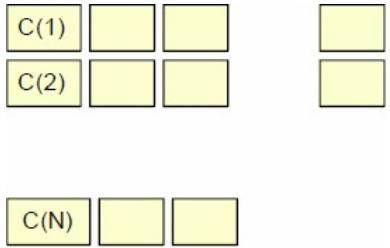
- Недостатки:
  - N точек, на каждую 2 (x, y) + 128 (SIFT) параметров – требуется очень много памяти;
  - Сопоставление дескрипторов всех точек по SIFT, например, по L_2 метрике займет очень много времени.
  - Достоинства: качество.

Ускорение сопоставления

- **Уменьшение размера индекса**
- Уменьшить размер дескриптора для описания локальной особенности;
- Квантование по словарю:
  - Составим словарь дескрипторов особенностей;
  - Квантуем особенности, т.е. заменяем дескриптор на номер в словаре;
  - Модифицируем метрику для сравнения дескрипторов:
    - Похожи (0), если одинаковый номер,
    - Непохожи (бесконечность), если номер разный.
- Можно ещё упростить сопоставление:
  - Опишем изображение «мешком слов»;
  - Качество сопоставления изображений можем считать как пересечение гистограмм – «мешков слов».

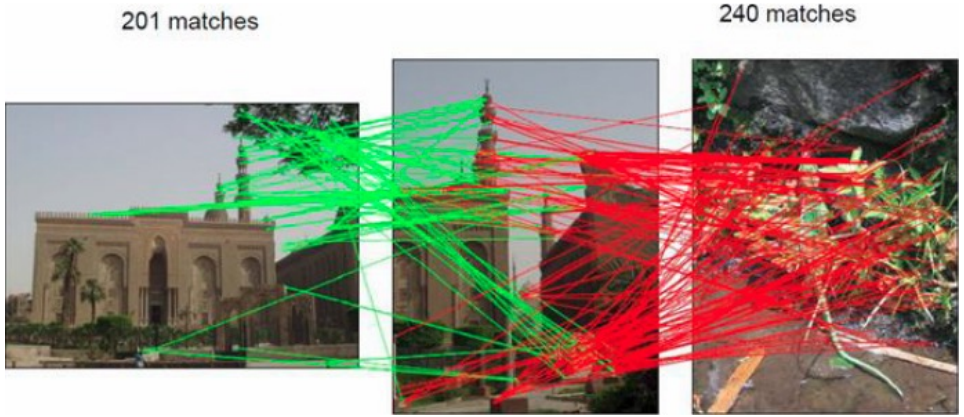
Инвертированный индекс

- Вектор слов в дескрипторе очень разреженный:
  - Например, 1k ненулевых элементов из 1M словаря.
- Удобно хранить его в инвертированном индексе:
  - Таблица (слова) x (изображения);
  - Список слов в словаре (терминов);
  - Для каждого слова храним список изображений, в котором слово встречается.
- Ускорение поиска:
  - Самые частые слова идут в начале списка.



Эффект квантования

- Словарь из 20 000 слов
- Сопоставим особенности в двух парах изображений по словарю, и посмотрим влияние размера словаря.
- Чем больше слов в словаре – тем точнее представление дескрипторов.



Эффект квантования

- Словарь из 200 000 слов

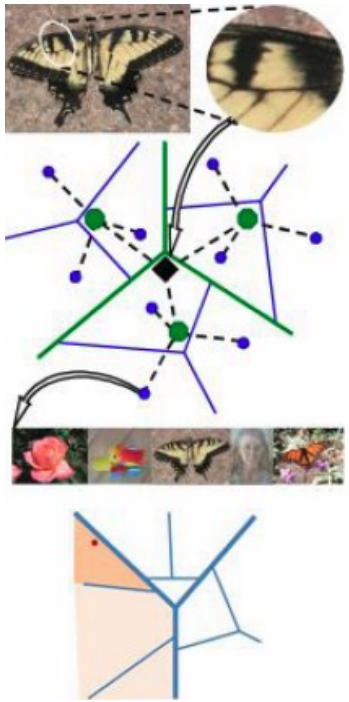
- Повышение размера словаря увеличивает точность сопоставления изображений.

Алгоритм поиска изобажений

- **Требования к алгоритму:**
  - Быстро строить словарь;
  - Быстро квантовать особенности;
  - Уменьшить ошибки дискретизации;
  - Минимизировать размер индекса.
- **Основные подходы к построению словаря и решению проблемы квантования**
  - **Hierarchical k-means (HKM);**
  - **Approximate k-means (AKM);**
  - **Hamming embedding;**
  - Soft assignment;
  - Fine vocabulary.

Hierarchical k-means (HKM)

- «Словарное дерево»;
- Иерархическое разбиение:
  - Кластеризуем всё на K кластеров ($k = 10$);
  - Затем данные в каждом кластере снова на k кластеров.
- Пример:
  - Глубина 6 даёт 1M листьев.
- Для снижение эффекта квантования дескриптор «мягко» присваивается всем родителям по ветви.

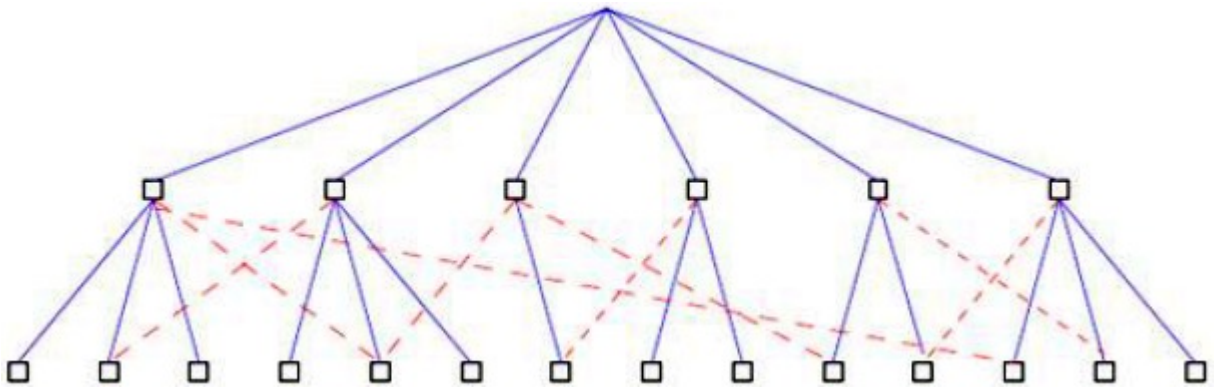


Approximate k-means (AKM)

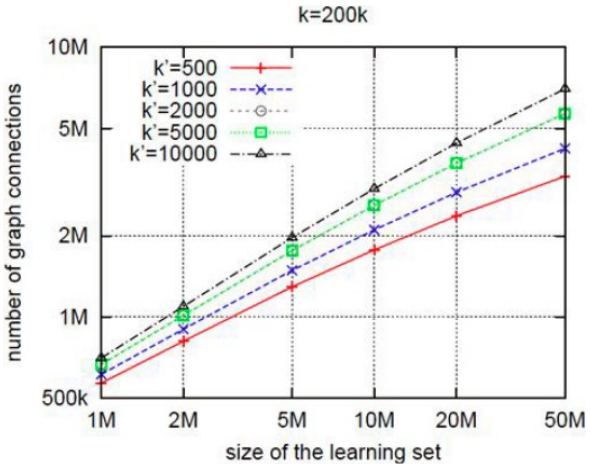
- Алгоритм:
  - Лес из 8 рандомизированных k-d деревьев;
  - Параметр (координата) разбиения выбирается случайно из набора с наибольшим разбросом;
  - Порог разбиения выбирается случайно недалеко от медианы.
- Такое разбиение позволяет уменьшить эффекты квантования.
- Сложность каждого этапа k-средних падает с NK до Nlog(K).

Ускорение квантования

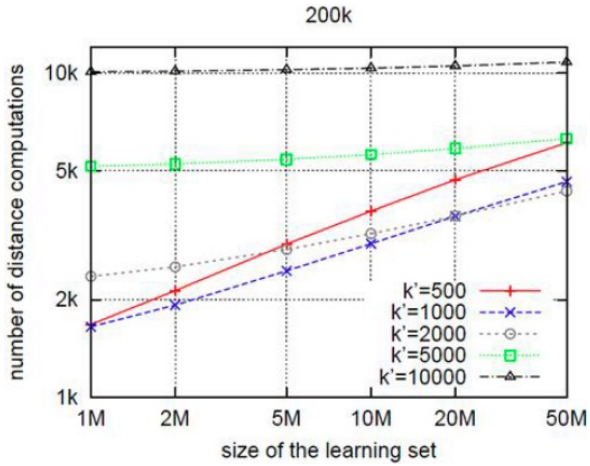
- Алгоритм:
  - Прямое сравнение дескриптора со всем словарём очень медленное
  - Построим иерархическую структуру:
    - С помощью k-средних построим первый уровень из k-слов;
    - Повторяем алгоритм k-средних над кластерами;
    - Обучим дополнительные связи между уровням на обучающей выборке.



Анализ алгоритма



Размер связей в графе в зависимости от размера обучающей выборки



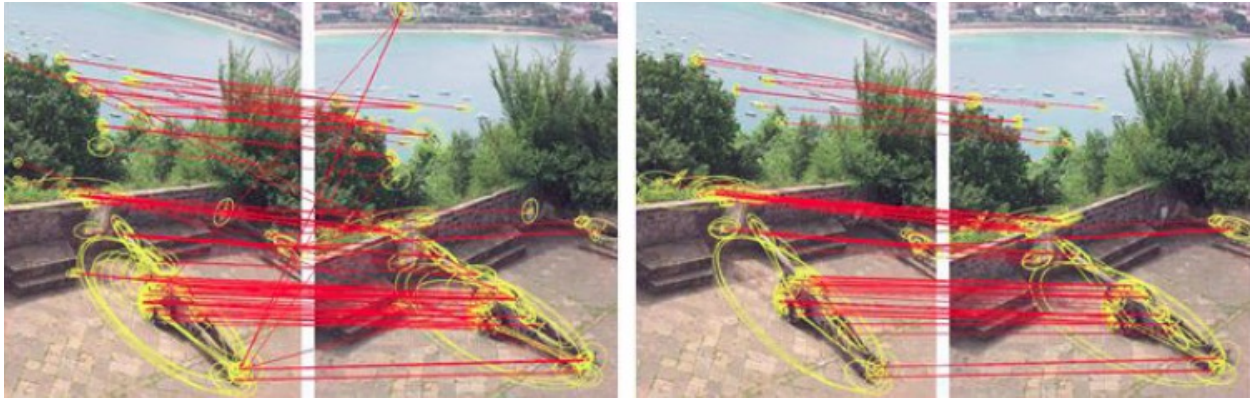
Количество расчетов расстояний при поиске ближайших

Поиск по мешку слов

- **Алгоритм:**
  - Дескриптор «мешок слов» большой размерности (1M);
  - АКМ для построения словаря по большой коллекции (5к);
  - Инвертированный индекс для хранения.
- **Тестирование:**
  - 5к+100к изображений, 1M слов, 1GB индекс, поиск в нём 0,1с.
  - 5к+100к+1M изображений, 1M слов, 4GB+ индекс, хранение файла на диске, поиск 10-35с.

Hamming embedding

- **HE, Вложение Хэмминга**
- Проблема выбора размера словаря:

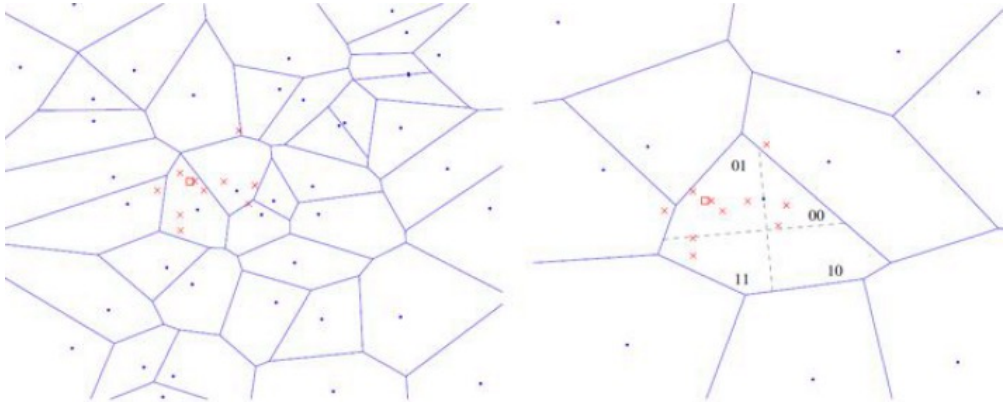


20 000 словарь 200 000 словарь

- Кластеризация недостаточно точно приближает функцию сравнения дескрипторов.
- Малый словарь – много ложных соответствий, большой – много пропущенных.

Hamming embedding

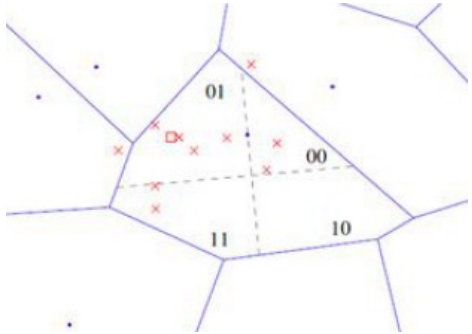
- Проблема выбора размера словаря:



- Маленький словарь – большие ячейки:
  - Слишком грубый порог на сравнение.
- Большой словарь – маленькие ячейки:
  - Слишком точный порог на сравнение.

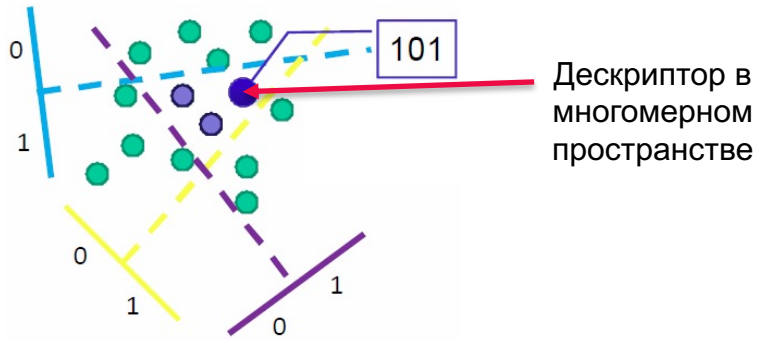
Hamming embedding

- Хотим записать не только номер слова для особенности из изображения, но и описать положение внутри ячейки (доп. код).
- Будем сравнивать тогда не только по номеру, но и по доп. коду.
- Код должен быть маленьким и сравнение быстрое:
  - Построим бинарный код;
  - Сравнивать будем по расстоянию Хэмминга.



Locality Sensitive Hashing

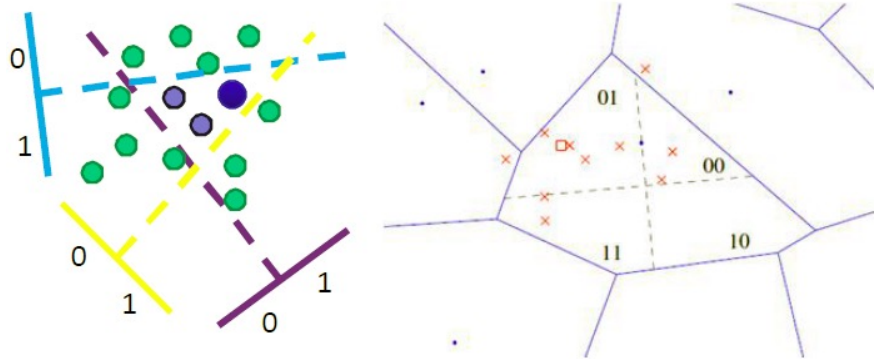
- LSH, Хеширование с учетом местоположения;
- Возьмем случайную проекцию данных на прямую;
- Случайно выберем порог, пометив проекции 0 или 1 (1 бит подписи);
- С увеличением числа бит подпись приближает L_2 – метрику в исходных дескрипторах.



- Недостатки:
  - Приближение L_2 – асимптотическое.
  - При реализации может потребоваться слишком много бит для подписи.

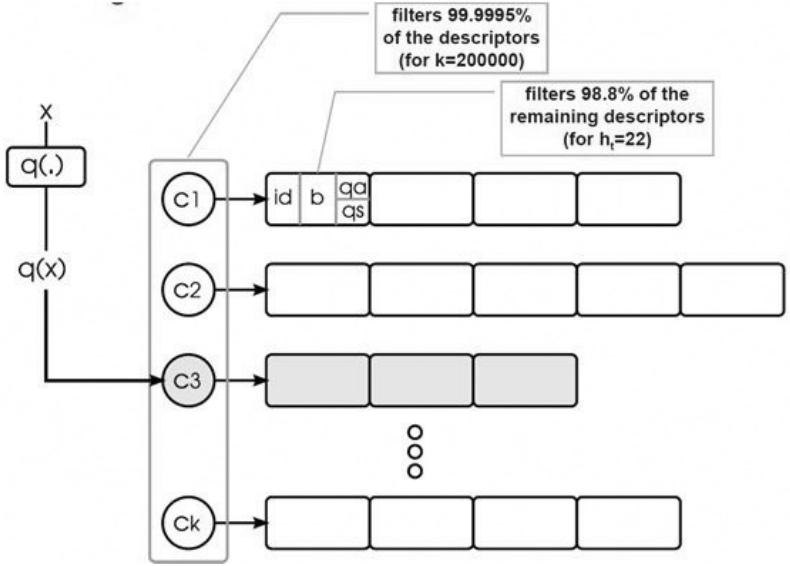
Hamming Embedding

- Возьмем все дескрипторы, попавшие в одну ячейку.
- Сгенерируем n случайных прямых (направлений проецирования).
- Спроецируем все дескрипторы на прямую.
- Выберем точку на прямой (порог) таким образом, чтобы справа и слева было поровну точек.
  - Такой код будет оптимальным.



Модификация индекса

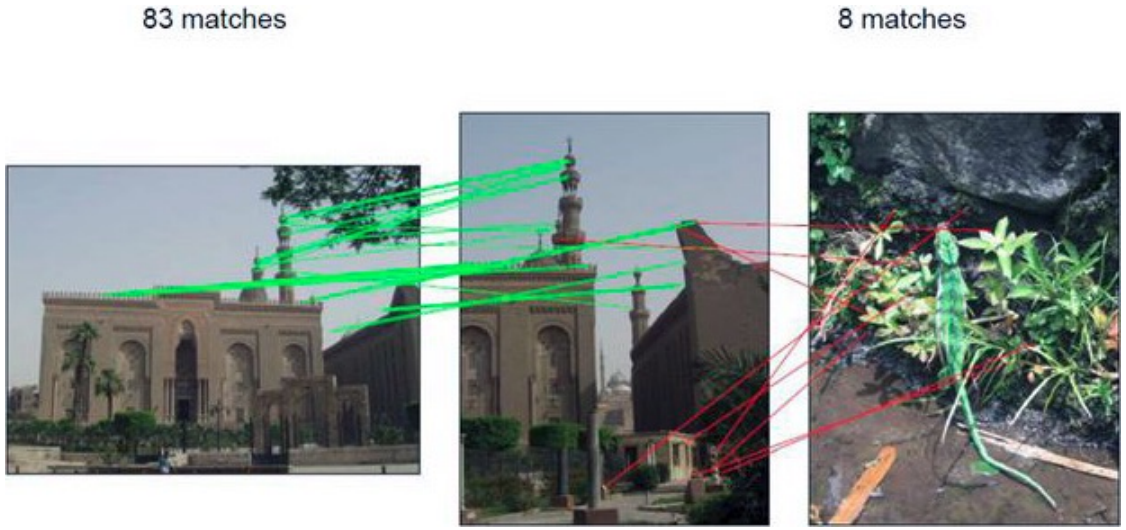
- Для каждой особенности – своя запись в индексе (до этого объединяли их в одну и писали просто количество).



Алгоритм

- Для каждого дескриптора:
  - Квантуем по словарю (номеру слова);
  - Вычисление бинарного кода.
- Считаем точки сопоставленными, только если выполняются оба условия:
  - номера слов совпадают;
  - коды по расстоянию Хэмминга отличаются не более чем на z.

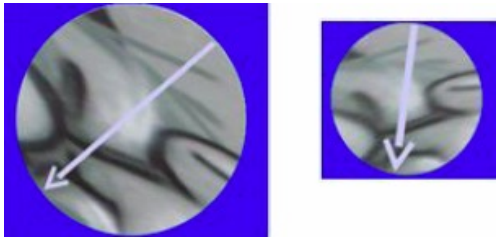
Hamming Embedding



«Слабая» геометрия (WGS)

- Каждая характерная точка определяется в т.ч. масштабом (характерным размером) и ориентацией.
- **Пример:**

20 градусов
разницы
по ориентации;
масштаб в 1,5 раза.

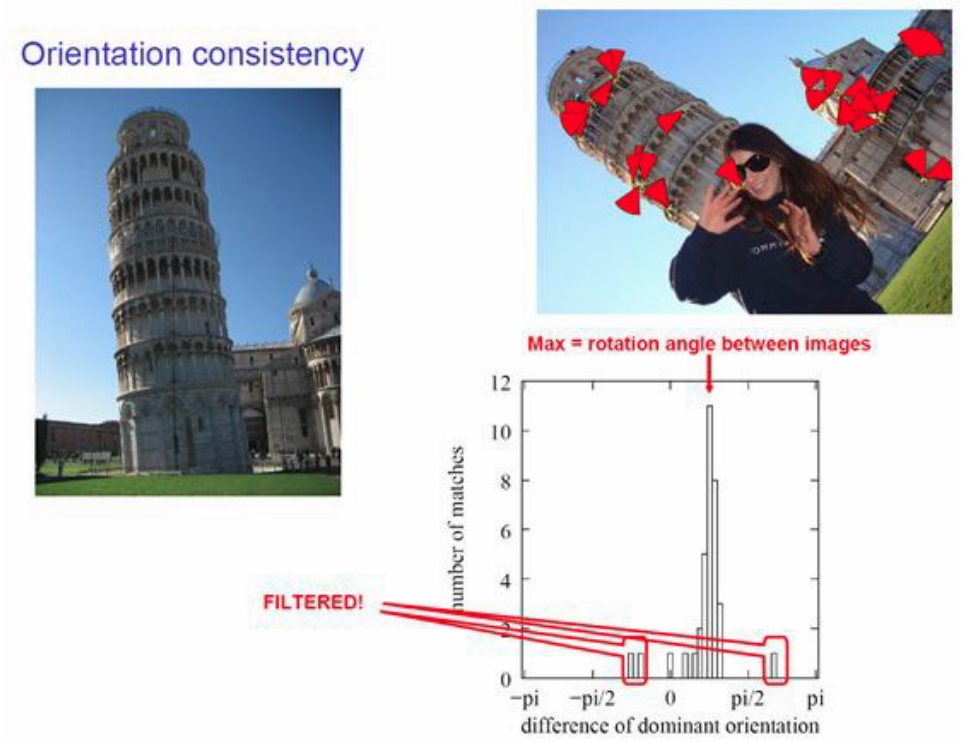


- Каждое соответствие точек задаёт разницу по углу и масштабу.
- Для изображения изменения должны быть согласованы.
- Каждая пара соответствующих точек будет голосовать за определенную комбинацию разницы в ориентации и масштабе.

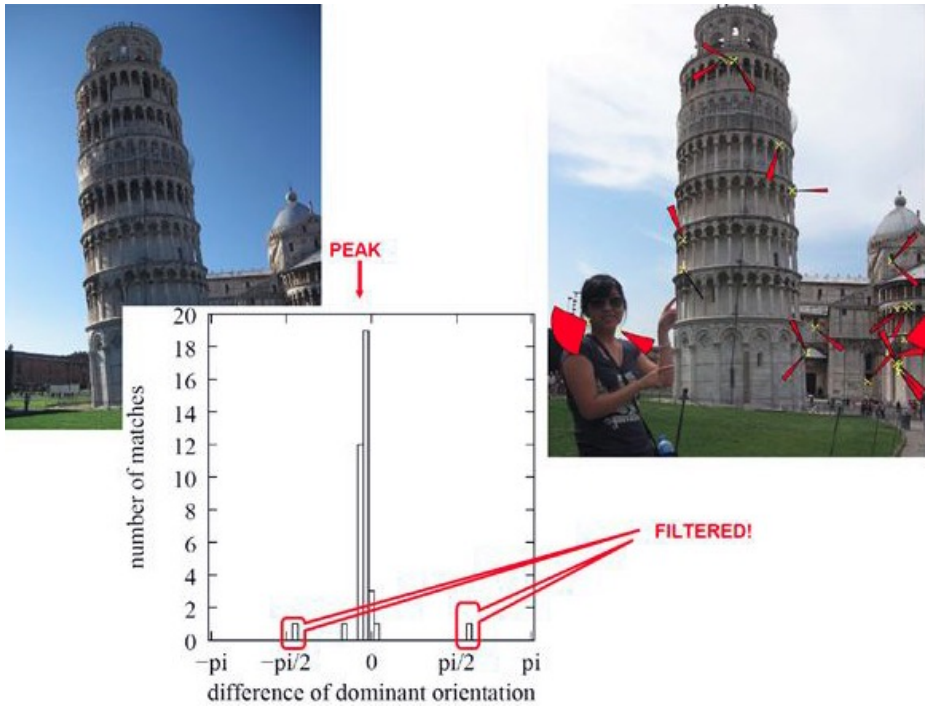
Пример


Pisa tower: Let analyze the dominent orientation difference of matching descriptors

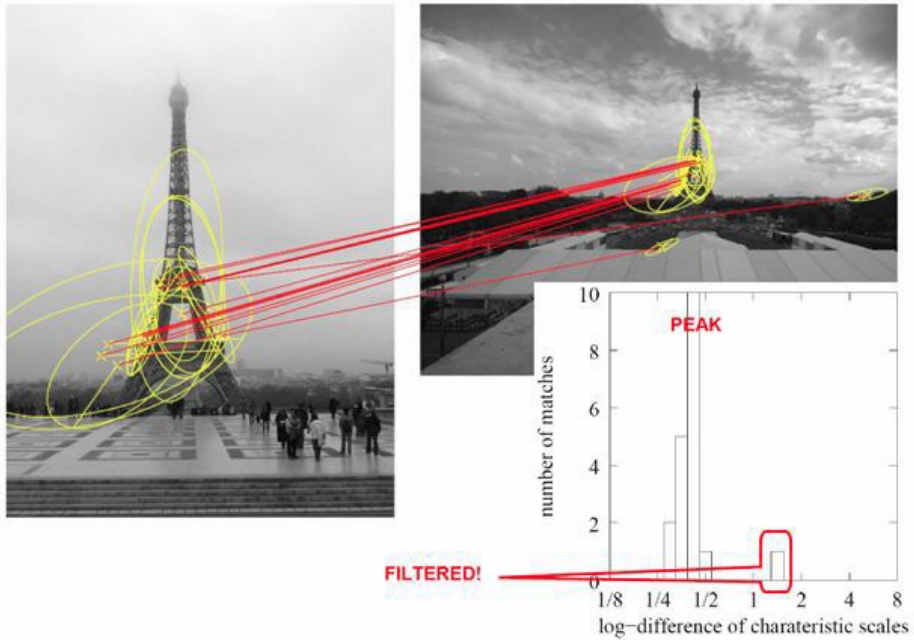
Пример


Orientation consistency
Max = rotation angle between images
FILTERED!
number of matches
12
10
8
6
4
2
0
−pi
−pi/2
0
pi/2
pi
difference of dominant orientation

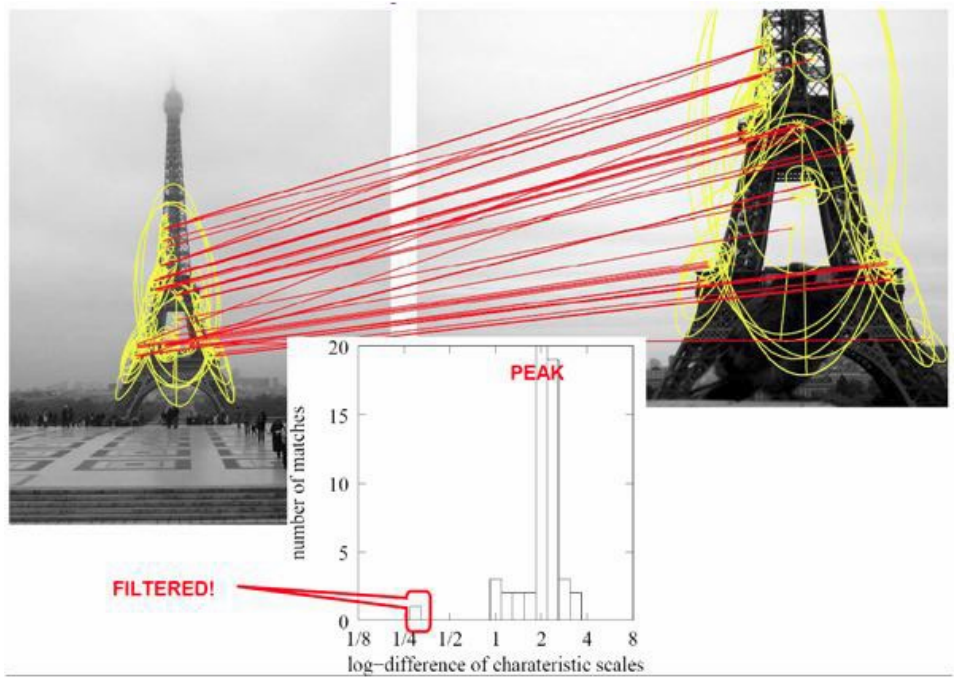
Пример


PEAK
FILTERED!
number of matches
20
18
16
14
12
10
8
6
4
2
0
−pi
−pi/2
0
pi/2
pi
difference of dominant orientation

Пример


PEAK
FILTERED!
number of matches
10
8
6
4
2
0
1/8
1/4
1/2
1
2
4
8
log−difference of charateristic scales

Пример

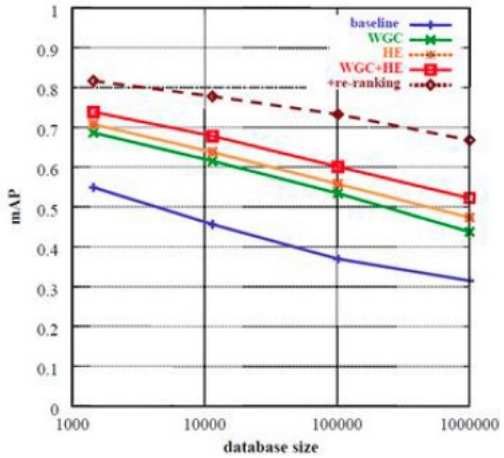

PEAK
FILTERED!
number of matches
20
15
10
5
0
1/8
1/4
1/2
1
2
4
8
log−difference of charateristic scales

Метод

- Масштаб и ориентация «примерно» не зависят друг от друга.
- Голосование с учётом дискретного масштаба и поворота:
  - Отдельный вес для каждой комбинации (угол/поворот):
    - Фактически, гистограмма.
  - Берем максимумы по углу / масштабу.
  - Берём из них минимум.
- Только соответствия, согласованные по изменению масштаба и ориентации вносят вклад в финальную оценку.

Анализ

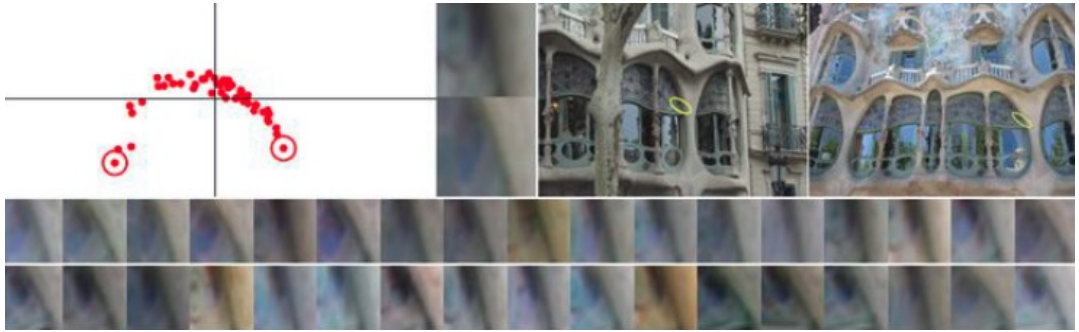
- Каждый элемент – weakly geometry, hamming embedding, ранжирование по геометрии существенно повышает точность.
- При этом совместно использование WGC и HE позволяет достичь скорости, сравнимой с базовым методом.



Average query time (4 CPU cores)	
Compute descriptors	880 ms
Quantization	600 ms
Search – baseline	**620** ms
Search – WGC	**2110** ms
Search – HE	**200** ms
Search – HE+WGC	**650** ms

Альтернативные слова

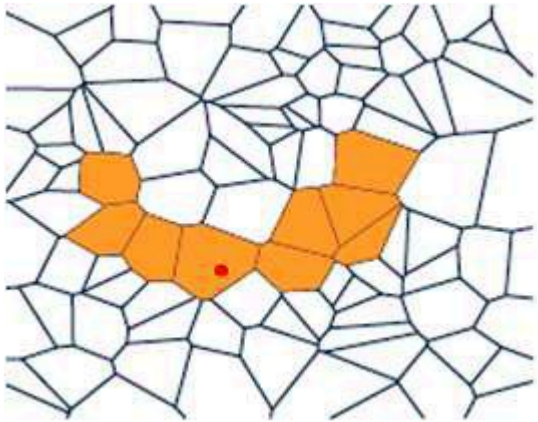
- Дескриптор SIFT не всегда достаточно инвариантен.
- Сильные перспективные искажения приводят к резкому увеличению расстояния.
- На рисунке – отображения дескрипторов одной и той же точки сцены на серии изображений.
- Одна и та же точка может попасть в разные кластеры.



Альтернативные слова

- На этапе обучения запомним «альтернативные слова», т.к. в каких ячейках могут оказаться «правильные» соответствия.
- На этапе поиска будем голосовать не только за то же слово, но и за альтернативные.
- Это увеличит размер индекса на:

$$\text{число альтернативных слов} * \text{размер словаря}$$



Обработка результата запроса

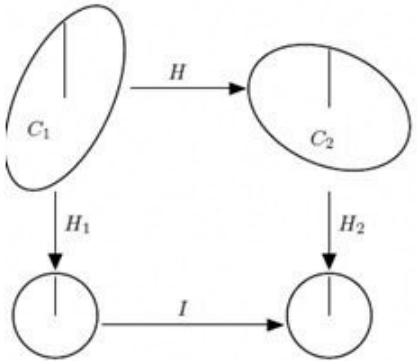
- Ранжирование списка результатов
- Раскрытие запросов
  - Если решаем задачу поиска объектов, то найденные изображения можно хорошо сопоставить с запросом.
  - Стандартная схема (локальные особенности + робастное вычисление преобразования) слишком медленное для полного перебора.
  - Можем использовать для постобработки – ранжирования найденных изображений.

Схема ранжирования

- Сопоставим особенности между «запросом» и отфильтрованными поиском изображением.
- Выбросы отфильтруем с помощью LO-RANSAC:
  - Сначала построим простую модель;
  - Затем по инлаерам более сложную модель.
- Уточним хорошую модель по найденным инлаерам:
  - Аффинная модель.
- Отсортируем изображения:
  - Для тех, которые сопоставились:
    - В начало списка;
    - Порядок по количеству инлаеров (чем больше – тем выше в списке).
  - Для тех, которые не сопоставились:
    - В конец списка;
    - Без изменения порядка.

Оценка модели по 1 паре

- Достаточно одной пары соответствующих точек для генерации гипотезы.
- Можно оценить до 5 параметров:
  - Сдвиг (2);
  - Масштаб (1);
  - Поворот (1);
  - Пропорции (эллипсоид).



Результаты ранжирования

- Доля нужных изображений в верхней части ранжированного списка после геометрического сопоставления:

- Например, для поиска изображений архитектуры ранжирование показывает существенный прирост в точности.

Раскрытие запросов

ИТМО

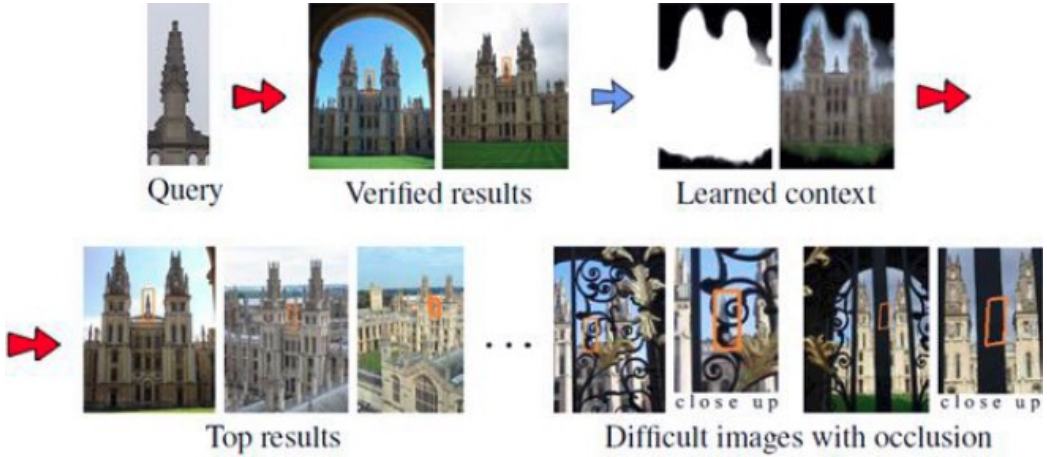
1. Transitive closure expansion (TCE)
   - Строим дерево запросов;
   - Вершина – исходный запрос;
   - Потомки – наиболее хорошо сопоставленные изображения из ответа на запрос.
2. Additive query expansion (AQE)
   - Отображаем интересные точки с найденных изображений на исходное;
   - Используем модифицированное изображение для поиска и дополнения результатов.
3. Average query expansion
   - Усредняем дескрипторы всех найденных изображений и используем для поиска.

Смущающие особенности

- В повторяющихся хаотических текстурах (вода) бывает много особенностей, которые не относятся к объекту, и они снижают качество поиска.

- **Идея:** в таких случаях изображения очень плохо геометрически сопоставляются.
- Обнаружим такие ситуации, выучим модель «*смущающих особенностей*» и удалим их из запроса.

Улучшение модели



- ***Модель*** – набор особенностей из запроса.
- **Идея:** Если мы нашли хорошо сопоставленное изображение, то стоит добавить особенности из него в модель.
- Обновленная модель позволит найти больше похожих изображений.

Incremental Spatial Re-Ranking (iSP)

- **Идея:**
  - Если мы нашли хорошо сопоставленное изображение, то стоит добавить особенности из него в модель.
- **Схема метода:**
  - Модель M – особенности из запроса X;
  - Пробегаем по найденному списку S;
  - Если между M и $S[i]$ сопоставилось больше $T = 15$ особенностей, тогда добавим их из $S[i]$ в M.

Вопросы?

ITMO more than a UNIVERSITY

s.shavetov@itmo.ru