Задача

- Content-based image retrieval
- Поиск изображений с определенным содержанием в базе изображений
- Задача похожа на выделение объектов и классификацию изображений, но фокусируется в основном на масштабировании

R. Datta, D. Joshi, J. Li, and J. Z.Wang. Image retrieval: Ideas, influences, and trends of the new age. *ACM Computing Surveys, 2008.*

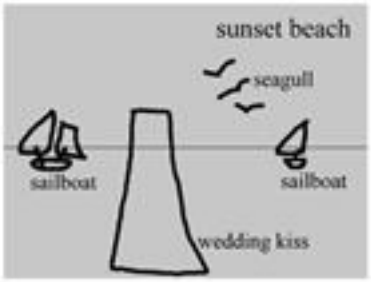
Что запрашивает пользователь?

- Запрос в виде атрибутов/текстового описания изображения
  - «Московская фотобиеннале-2012, фото» - аннотация (атрибуты) изображения
  - Нужна категоризация/аннотация изображений
- Запрос в виде изображения-примера («найди то же самое»)

- Запрос в виде некоторой характеристики содержимого

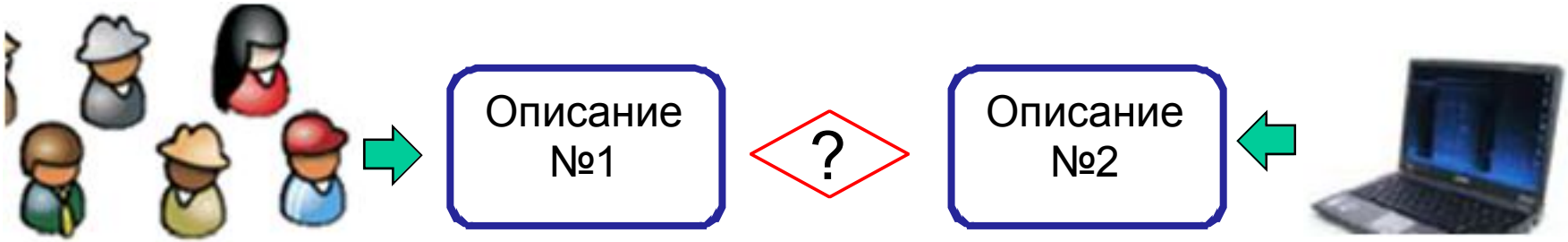
- Гистограмма цветов

- Запрос в виде рисунка-наброска



Semantic Gap

- Запрос в виде изображения-примера («найди то же самое», «найди похожее изображение»)
- Что имел пользователь в виду?
- Что значит «похожее изображение»?
- «Семантический разрыв» – несовпадение информации, которую можно извлечь из визуальных данных, и интерпретацией тех же самых данных со стороны пользователя



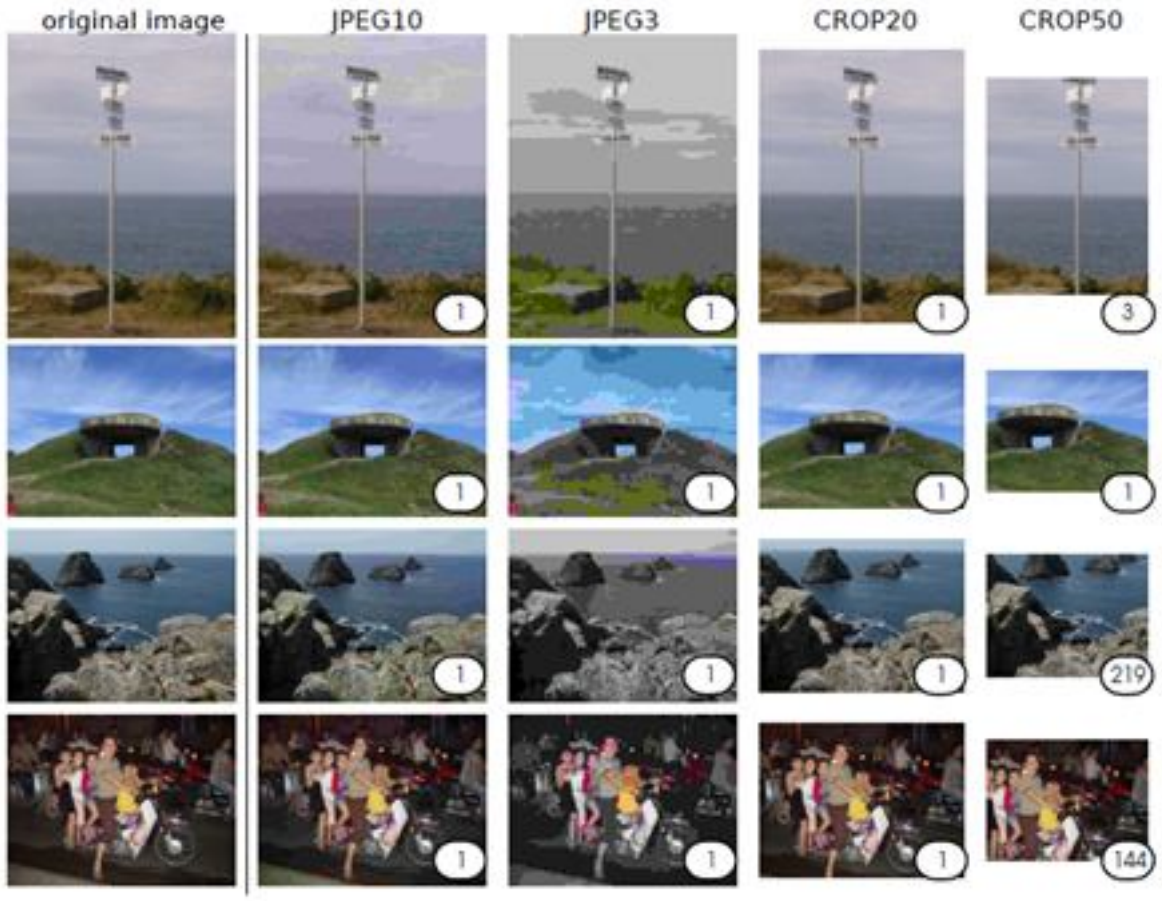
Что значит похожее?

1. Похожее по каким-то характеристикам, например, по цвету

Запрос

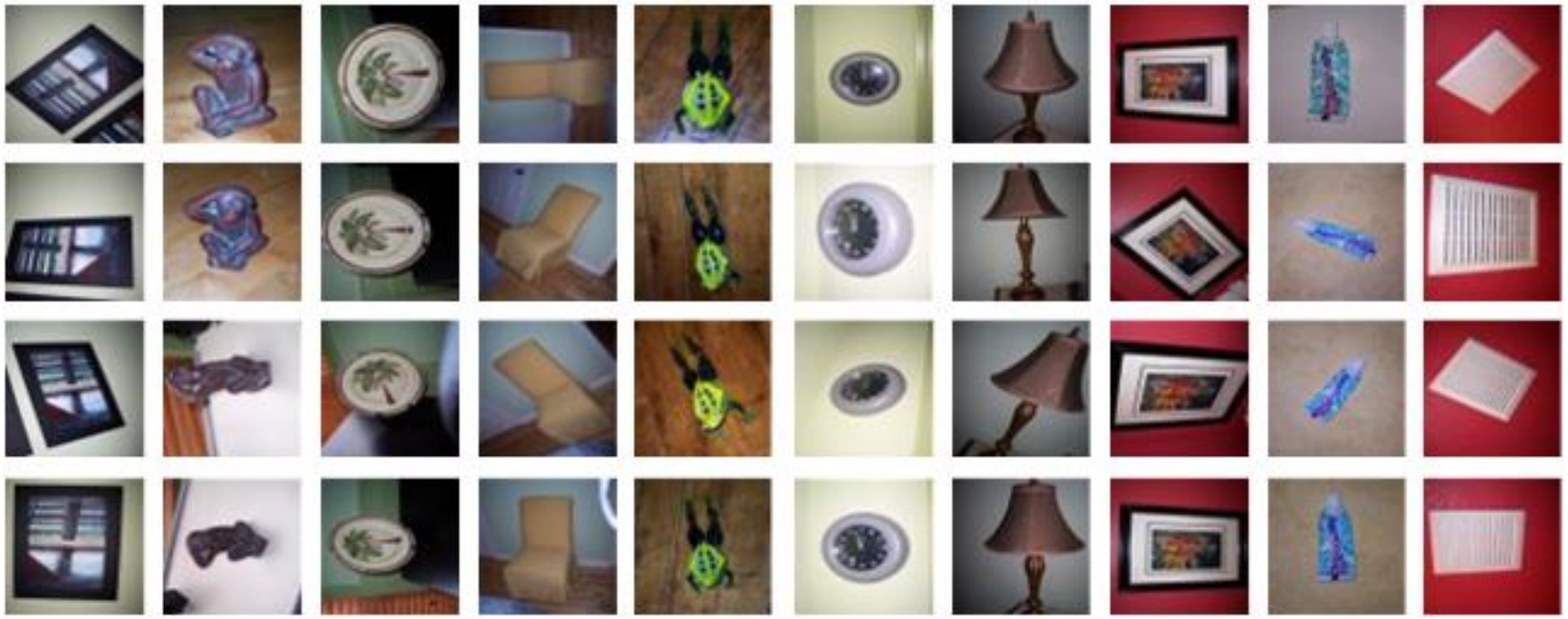
Что значит похожее?

2. Полудубликаты (Near-duplicates) – слегка измененная версия изображения (ракурс, цвета)



Что значит похожее?

3. Тот же самый объект или сцена («Object retrieval»)
   - Большие вариации ракурсов, фона, и т.д., чем при поиске полудубликатов



Что значит похожее?

4. Похожие визуально по геометрии сцены с учетом ракурса (могут быть разные по назначению)

Кухня

Приемная

Бар

Автобус

Самолет

Зал

Что значит похожее?

5. «Category-level classification» - изображения одного класса сцен или объектов



Пример – банкетный зал.

Например, 256 классов из базы Caltech 256

Анализ постановок задач

- Какие постановки задач внутренне схожи, а какие существенно отличаются?

Визуальное подобие

Семантическое подобие

- В последнем случае нам нужно выполнить автоматическую аннотацию изображения, определить класс сцены, присутствующие объекты, их характеристики
- Затем мы будем сравнивать метки запроса с метками изображений в базе
- Мы рассмотрим именно визуальное подобие

Общая схема поиска изображений

Построение индекса

Коллекция изображений

Признаки изображения → Подпись (Signature) → Индекс коллекции

Дескриптор на основе извлеченных признаков

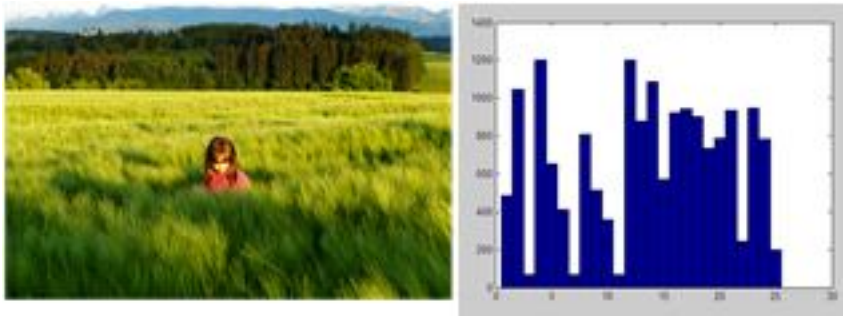
Тестирование

Тестовое изображение

Признаки изображения → Подпись (Signature) → Поиск по индексу → Список похожих

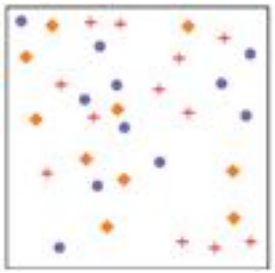
Ищем ближайших соседей по выбранной метрике

Дескрипторы изображений



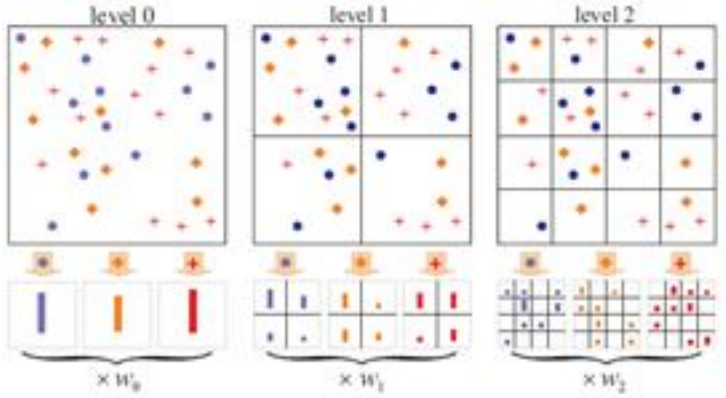
Гистограммы цветов (1995)

Мешок слов (2003)



Отдельные особенности (2003)

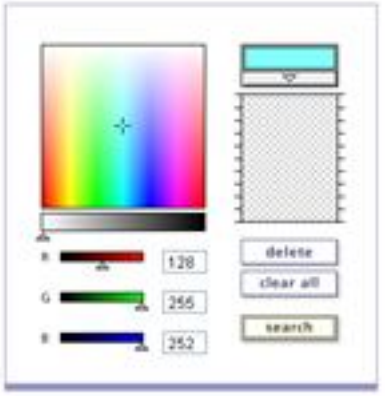
Гистограммы градиентов (2005)

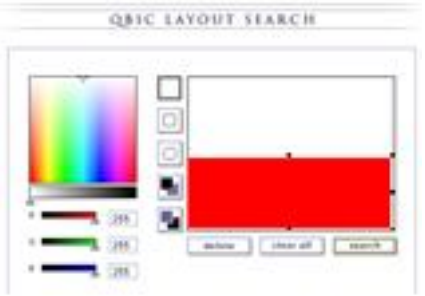


Мешок слов и пирамида (2006)

QBIC (1995)

- «Query By Image Content»
- Самая первая система CBIR
- Изображения сравниваются по набору признаков
  - Цветовая гистограмма
  - Выделенные вручную объекты и признаки их формы (размер, площадь, количество)
- ~10000 изображений в базе

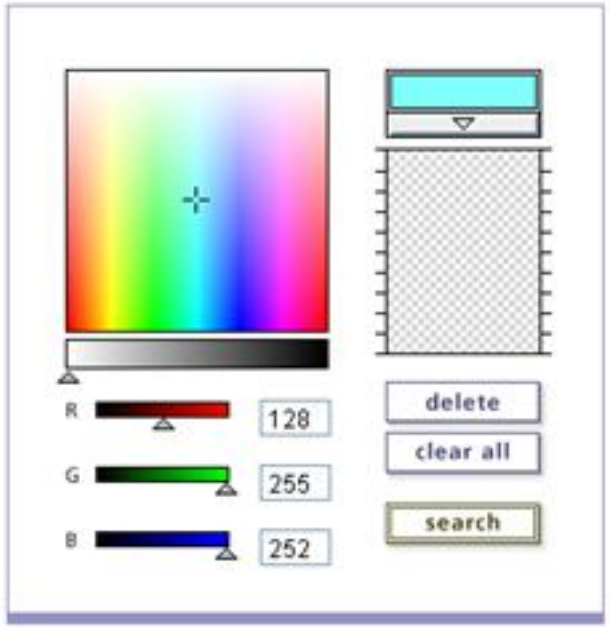




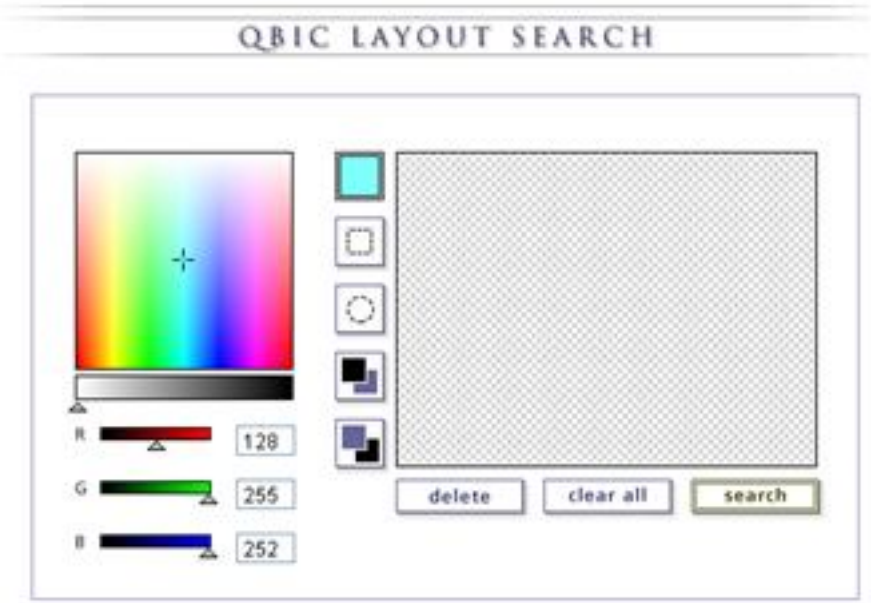
M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker. Query by image and video content: the QBIC system. *IEEE Computer, 28(9):23–32, 1995.*

QBIC: Интерфейс

Реализация на сайте Эрмитажа



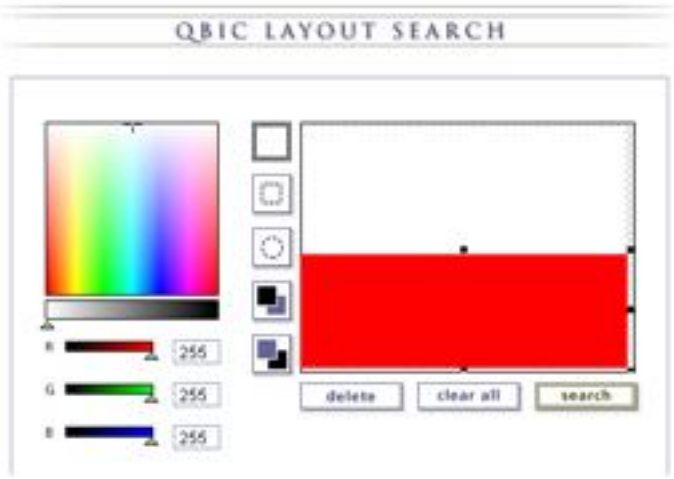
Гистограмма



Пространственное распределение цветов

http://www.hermitagemuseum.org/fcgi-bin/db2www/qbicSearch.mac/qbic?selLang=Russian

QBIC: Пример использования



1) **Vase of Flowers**

Huijsum, Jan van Early 18th century

2) **Seascape with Venice in the Distance**

Cottet, Charles Circa 1896

3) **Boats on a Sea Shore**

Goyen, Jan Jozefsz van 1641

4) **Avenue in a Park**

Watteau, Antoine Circa 1715

5) **Bird Perching on a Rose Twig**

UNKNOWN 18th century

6) **Old Woman with a Spindle**

Watteau, Antoine 1710s

7) **Interiors of the New Hermitage. The Room of Russian Sculpture**

Premazzi, Luigi 1854

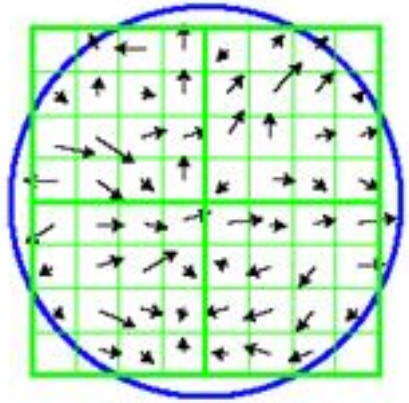
8) **Allegory of George I, King of England**

Vanloo, Carle (Charles-Andre) 1736

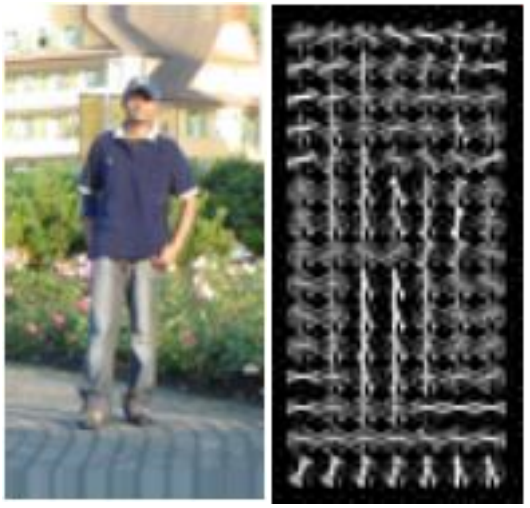
Качество работы системы понятно. Но первая!

Гистограммы градиентов

- Что общего у SIFT и у HOG?
- Насколько они подходят для сравнения визуального подобия произвольных изображений?
- Как в этих методах учитывается масштаб/размер объекта?

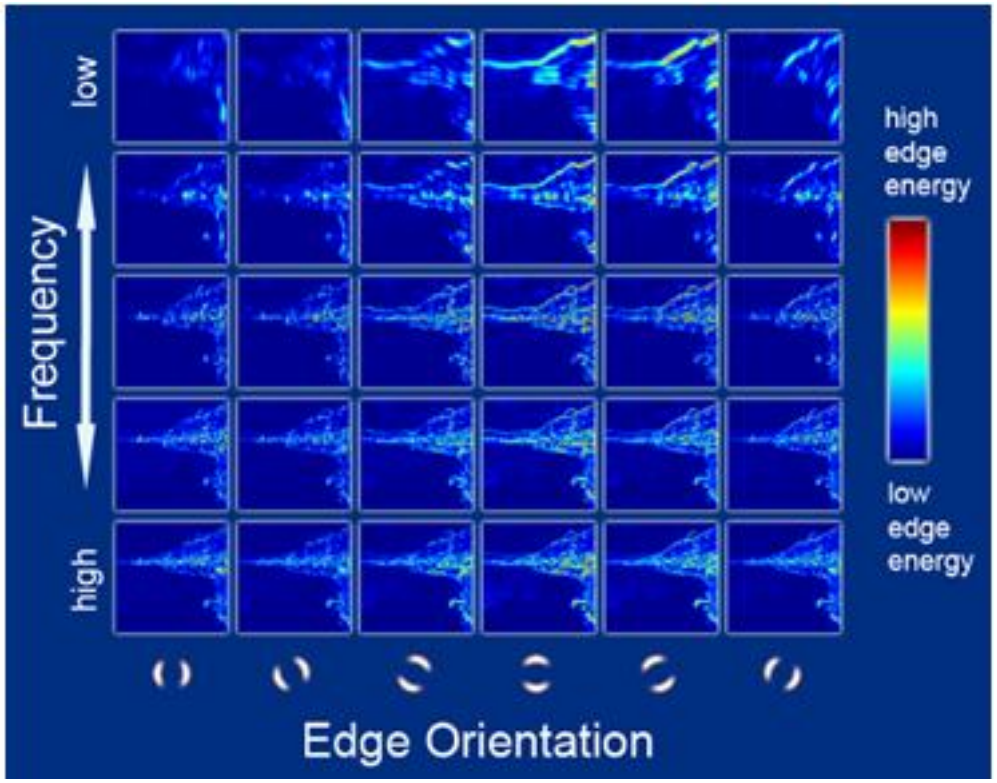


SIFT (2003)



HOG (2003)

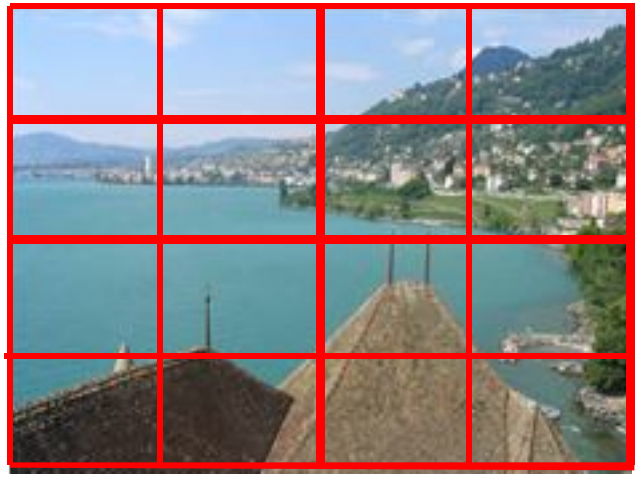
Дескриптор изображения GIST

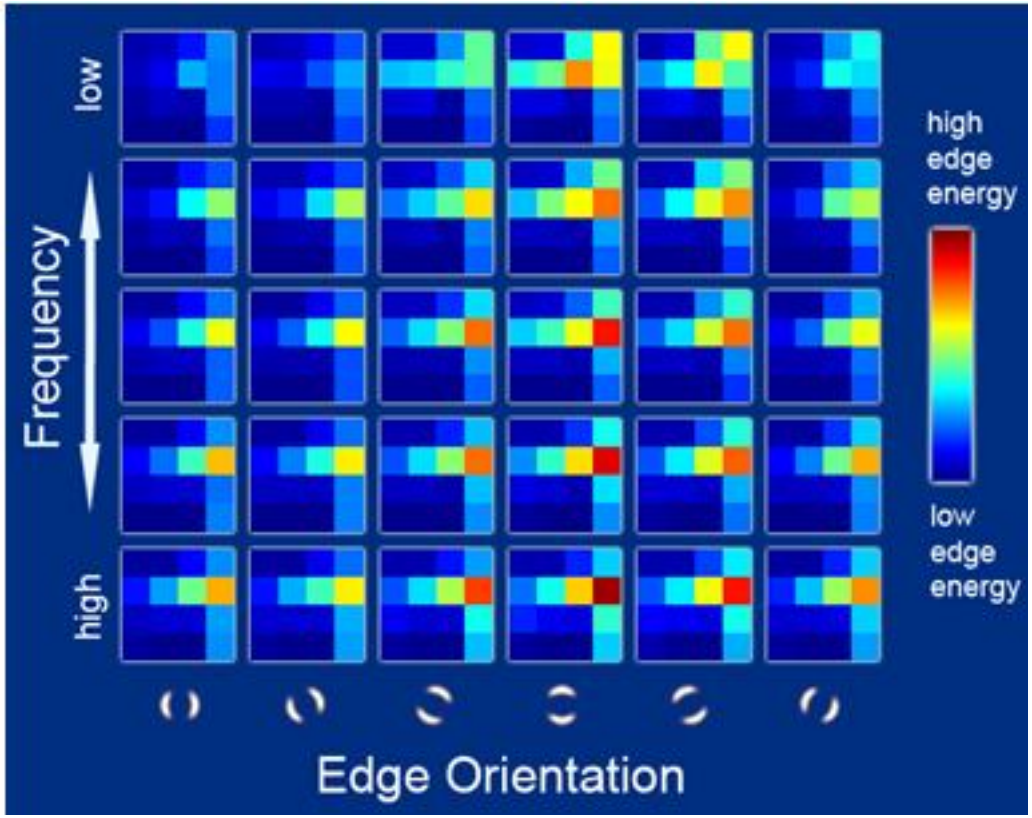


- Воспользуемся методом, похожим на SIFT.
- Посчитаем отклики детекторов краёв на 5 разных масштабах и 6 ориентациях края
- Получим 33 «канала» - RGB-цвет и 30 откликов фильтров края

TORRALBA, A., MURPHY, K. P., FREEMAN, W. T., AND RUBIN. Context-based vision system for place and object recognition. In ICCV 2003

Дескриптор изображения GIST





- Разобьём изображение сеткой 4x4 на 16 ячеек
- В каждой ячейке усредним значения всех каналов
- Получим дескриптор GIST

Применение GIST

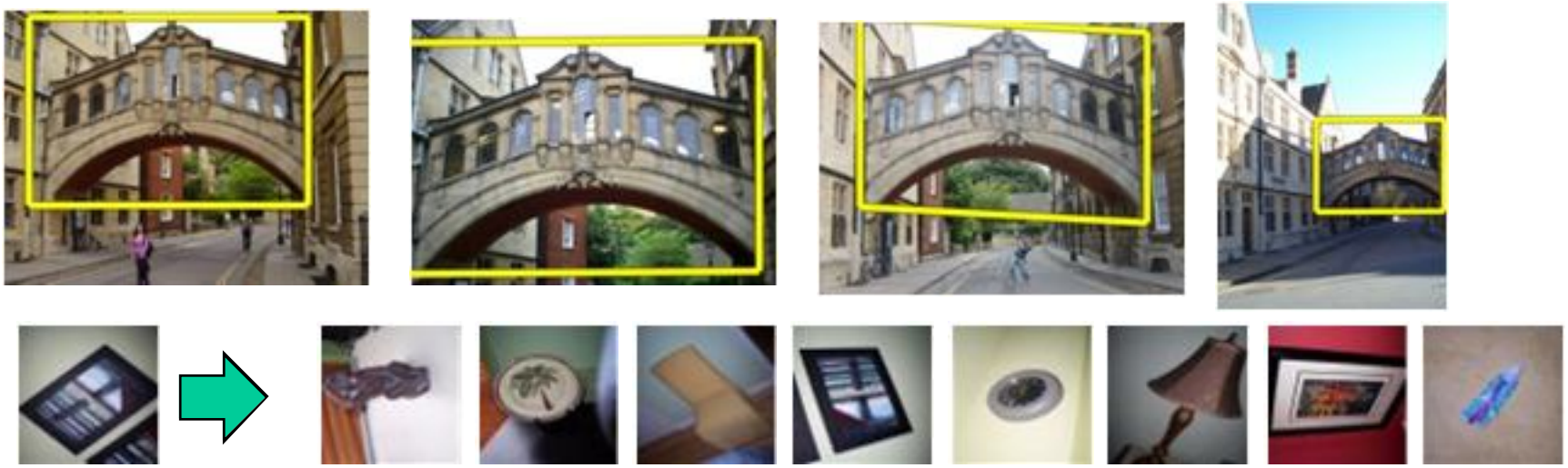
Запрос

Похожие по GIST + цвету

- GIST хорошо себя показал для оценки визуального подобия изображений
- Отлично подходит для поиска полудубликатов

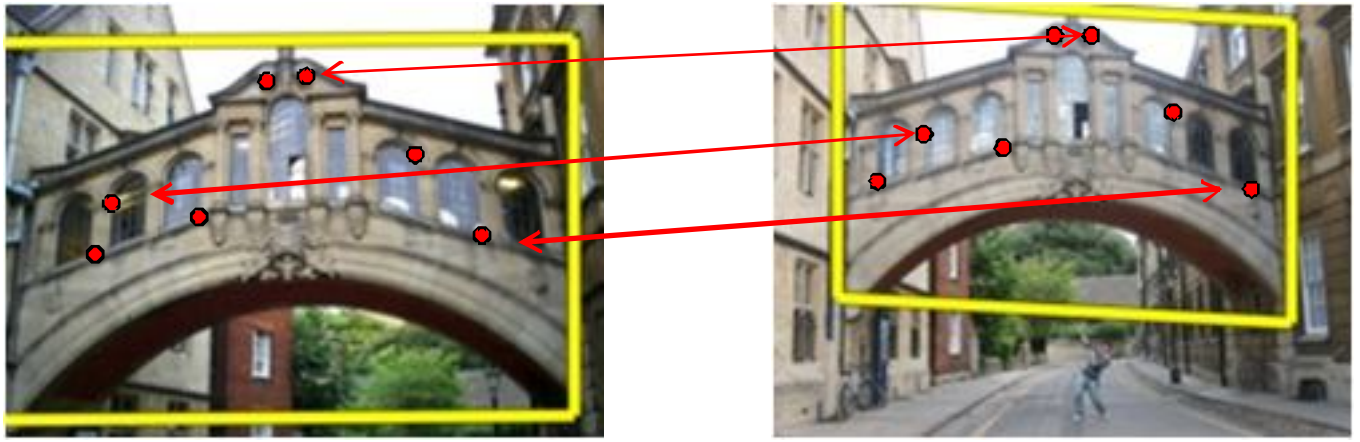
Ограничения GIST и цветов

- Ищем изображения одного и того же объекта



- Как с этой задачей справится GIST?
- Не очень хорошо, т.к. размеры и ориентация объектов могут значительно меняться
- Может помочь сопоставление изображений по ключевым точкам

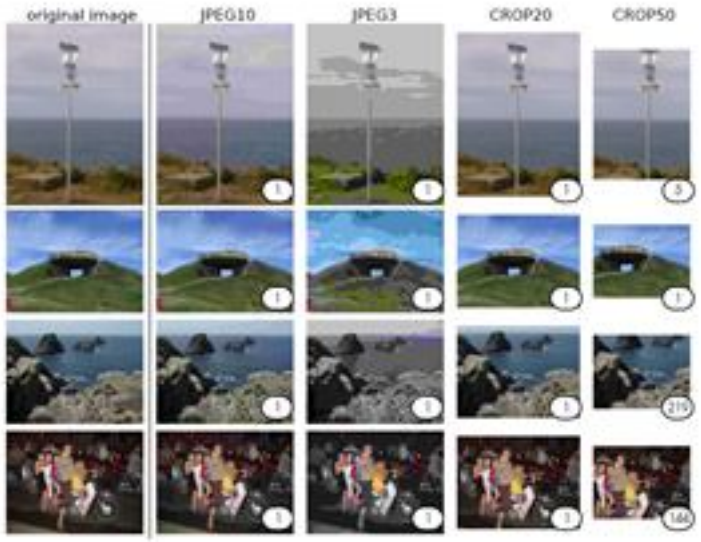
Геометрическое сопоставление



- Построим «дескриптор изображения» как набор найденных точек SIFT с дескрипторами
- Визуальное подобие будем оценивать через сопоставление двух изображений по набору SIFT
  - Найдём соответствующие пары точек по дескрипторам
  - Чем больше соответствий, тем более похожи два изображения
  - Можем оценить геометрическое преобразование с помощью RANSAC и отфильтровать ложные

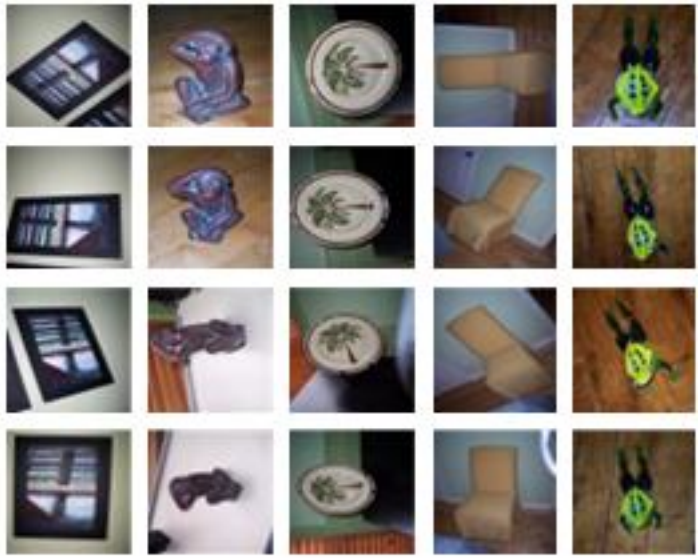
Выбор дескрипторов

Полудубликаты



GIST

Похожие объекты



Набор точек SIFT

- Теперь проанализируем эффективность по вычислениям и по памяти

Затраты по памяти

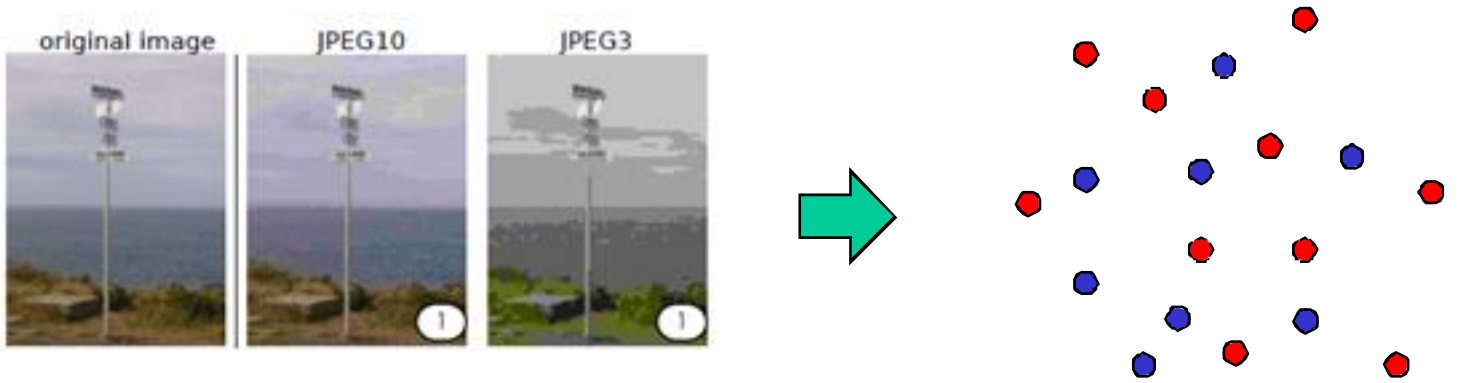
- GIST
  - решетка 4x4 * 8 ориентаций * 4 масштаба = 512 параметров
  - при 4 байтах на параметр – 2048 байт (16384 бита)

- Точки и дескрипторы SIFT
  - SIFT = решетка 4x4 * 8 ориентаций = 128 параметров
  - 1000 точек = 128000 параметров
  - Итого дескриптор – от 128 до 512 кбайт

- 1M изображений
  - 2Гб для GIST
  - 512Гб для точек SIFT

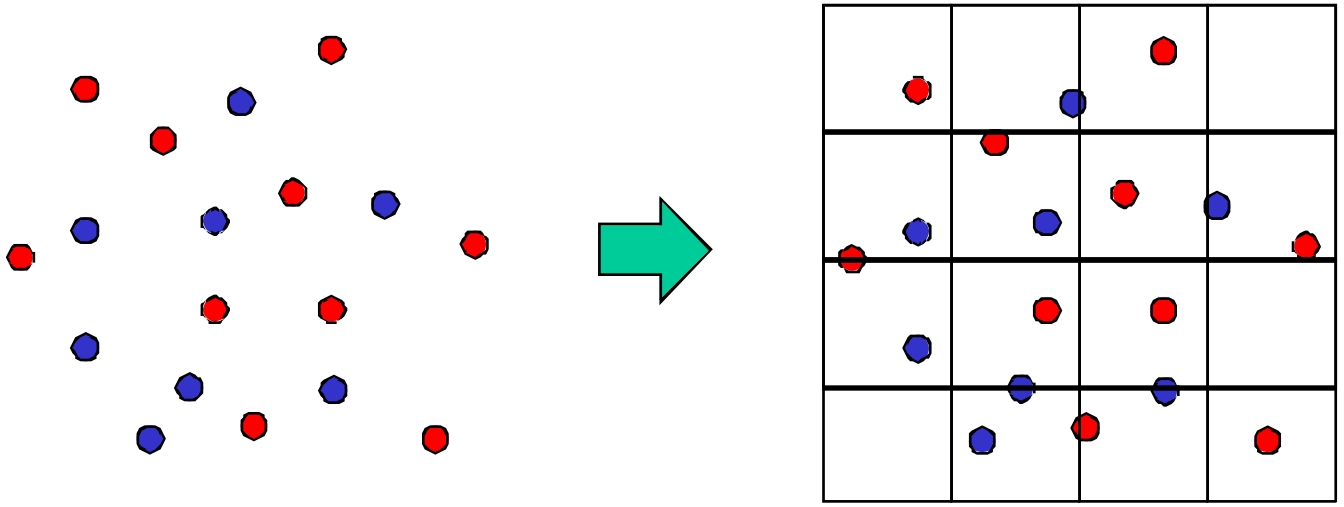
Уже тяжело или невозможно всё в RAM хранить!

Скорость работы



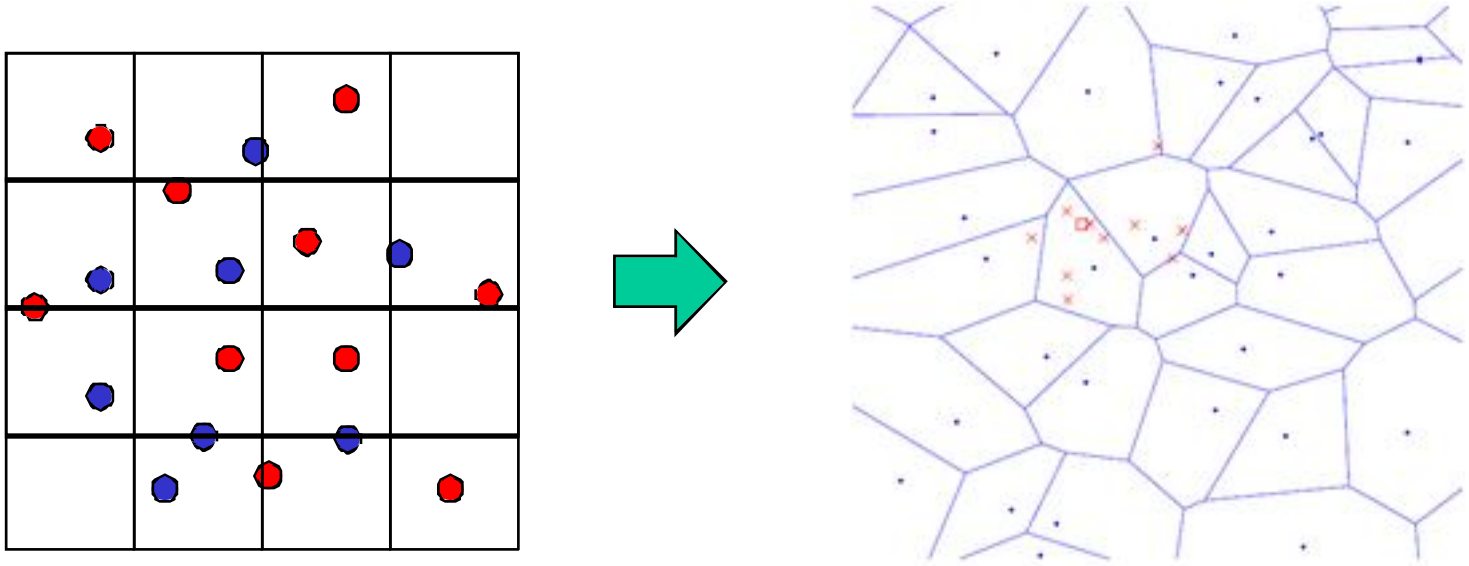
- Задача CBIR сводится к поиску ближайших соседей по дескриптору во всей коллекции (Nearest Neighbor)
- Простейшее и точное решение – линейный поиск
- Размер коллекции – миллионы и миллиарды векторов
- Нужны быстрые методы, пусть и с потерей точности

Квантование векторов



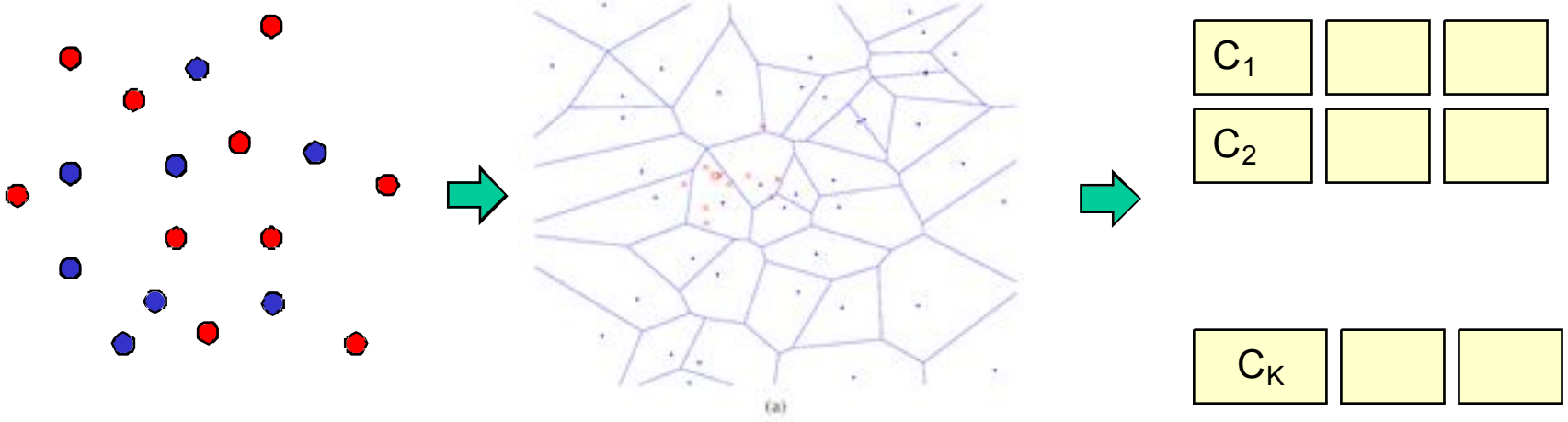
- «Vector quantization» – способ ускорения поиска за счёт уменьшения точности
  - Отображение x на q(x), где q(x) in $\{c_i\}$, c_i – центроиды, i от 1 до K
- Индекс q(x) можно записать в виде битового кода длиной $\log_2(K)$
- Простейший способ квантования – разбиение пространства дескрипторов на ячейки вдоль осей координат

Адаптивное квантование



- Регулярное разбиение не учитывает структуру данных, их распределение в пространстве дескрипторов
- Адаптивно разбить пространство можем с помощью кластеризации K-средними

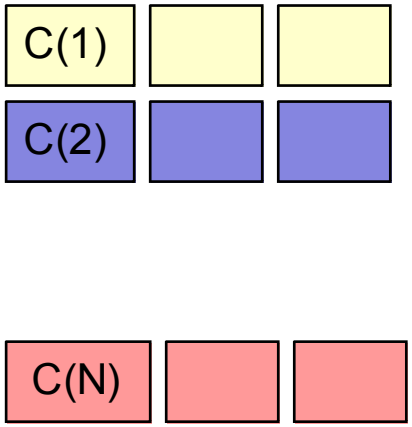
Инвертированный индекс

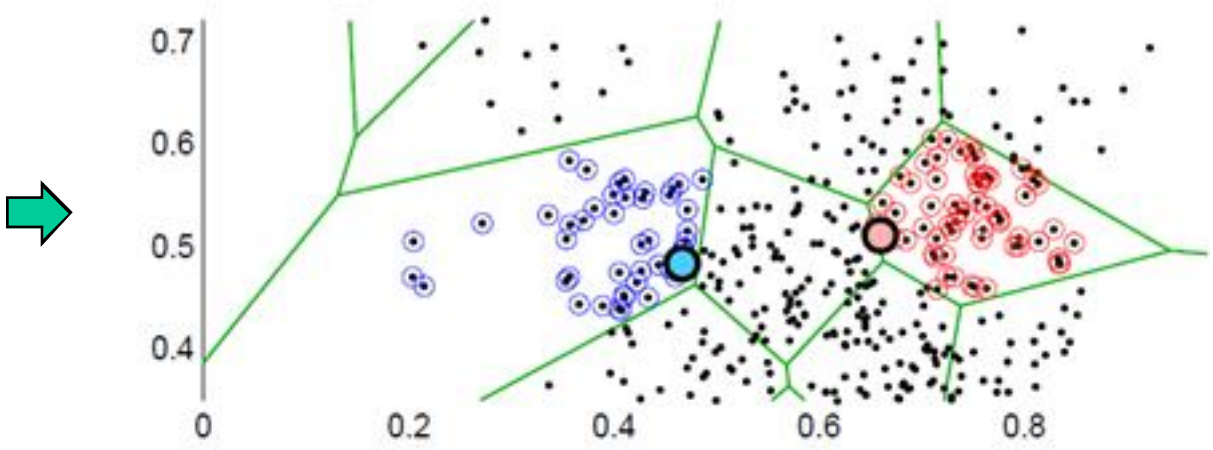


Квантованные вектора удобно записывать в форме «инвертированного индекса»

- K списков по числу центроидов (кодовых слов)
- В каждом списке храним индексы всех векторов, квантованных до C_i

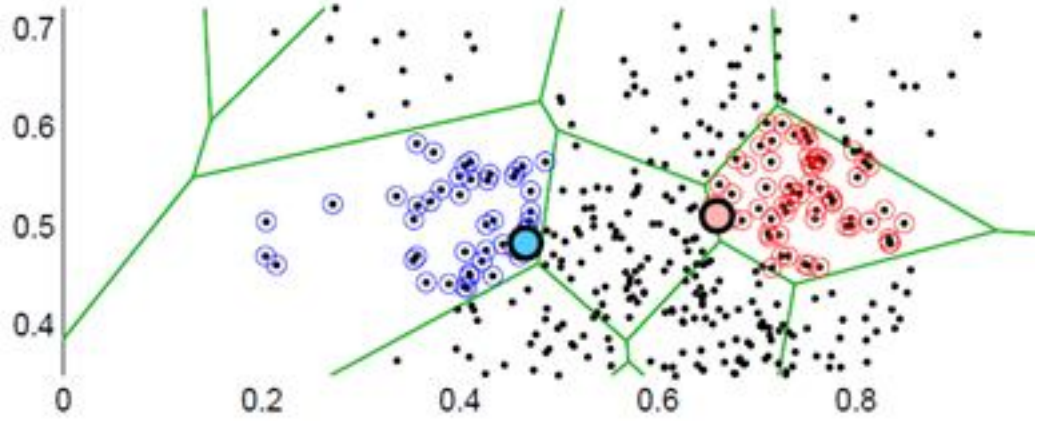
Инвертированный индекс





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

Ранжирование результатов



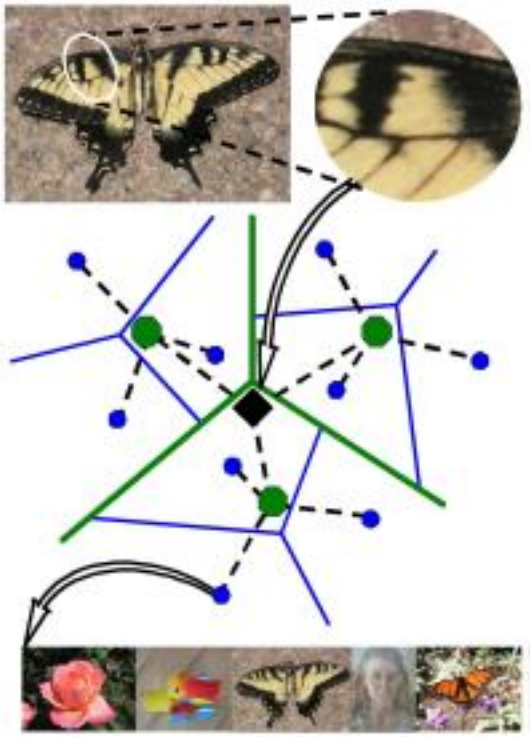
- Зачем выдавать все элементы в списке из одного кластера неупорядоченно?
- Упорядочим результаты (Re-ranking)
  - Рассчитаем расстояния от вектора-запроса до каждого элемента списка по полному дескриптору
  - Упорядочим результаты по близости (первые – ближайшие)
- Есть и другие способы ранжирования!

Проблемы квантования К-средними

- Для повышения точности требуется существенно увеличивать число кластеров K
- Для векторов SIFT квантование до кодов 64 бита (0.5 бита на параметр) требует подсчёта и хранения 2^{64} центроидов, что невозможно
- Сложность одного этапа O(NK)
  - Медленно на больших выборках (большие N)
  - Медленно при больших K
- Поэтому нужны другие, более быстрые приближенные методы.

Hierarchical k-means (HKM)

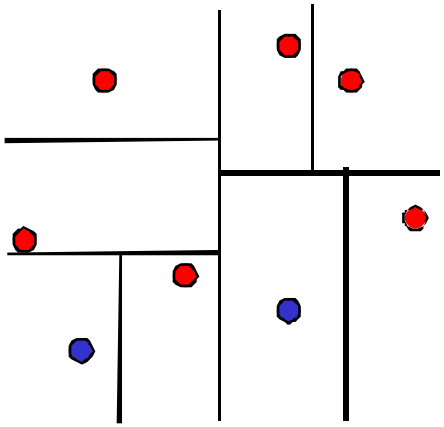
- Иерархическое разбиение
  - Кластеризуем всё на K кластеров (K=10) с помощью K-средних
  - Затем данные в каждом кластере снова разбиваем на K кластеров
  - Повторяем до достижения нужной глубины
- Пример:
  - Глубина 6 даёт 1M листьев
- По точности проигрывает существенно K-средним

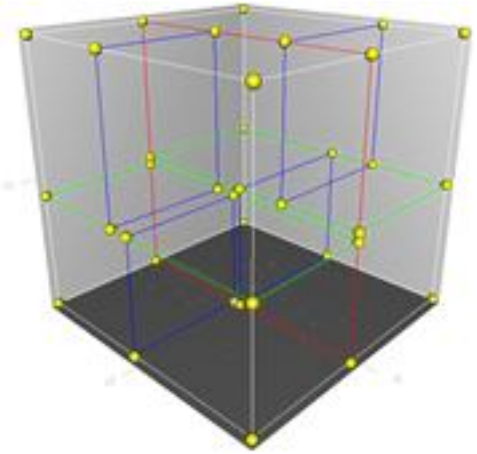


D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In Proc. CVPR, 2006.

KD-дерево

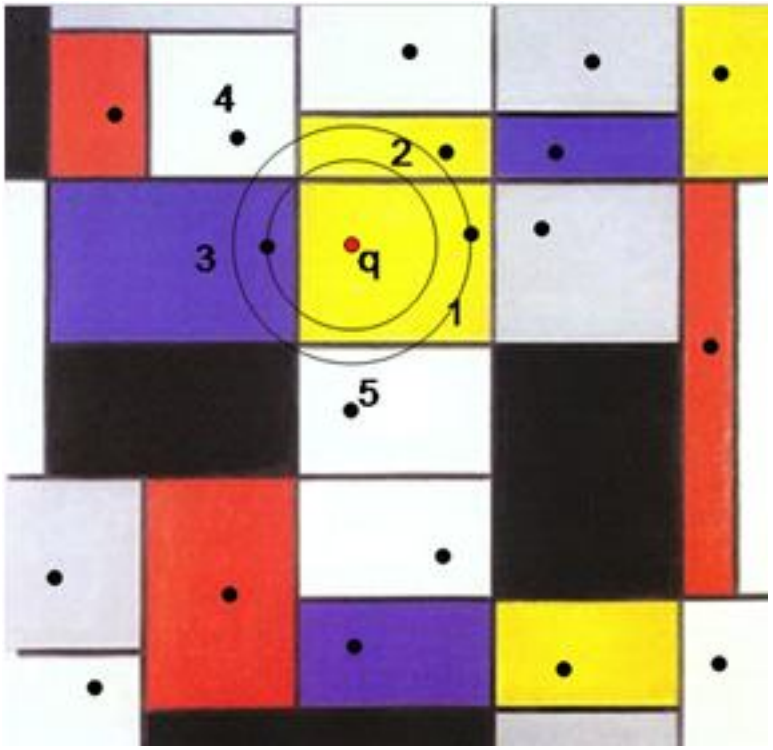
- Метод быстрого поиска ближайшего соседа
- Алгоритм:
  - Выбираем параметр (ось), по которому разброс точек наибольший
  - Вычисляем медиану по этому параметру
  - Запоминаем параметр и медиану в вершине
  - Разбиваем пространство на 2 полупространства, запускаем процедуру построения вершины для всех элементов из каждого полупространства
- В каждом листе 1 элемент
- Высота дерева $\log_2(N)$





J. H. Freidman et. Al. An algorithm for finding best matches in logarithmic expected time. ACM *transactions on mathematical software*, 1977

ANN



- ANN – Approximate Nearest Neighbour
- Priority search
  - Спускаемся по дереву в ячейку
  - Составляем список ячеек, сортируя его по расстоянию от запроса до границы
  - Просматриваем элементы в ячейках по списку

S. Arya et. Al.. An optimal algorithm for approximate nearest neighbor searching in fixed dimensions. Journal of the ACM, 1998

Проблемы высокой размерности

- Эффективность kd-деревьев падает с увеличением размерности
  - «Curse of dimensionality»
- Количество векторов N в выборке должно быть больше 2^M, где M – количество параметров
  - Иначе мы просто не сможем учесть все параметры при построении дерева
- Для 512-размерных GIST такое количество точек невозможно обеспечить

Рандомизированные kd-деревья

- Не можем построить дерево нужно глубины из-за нехватки данных
- Построим несколько рандомизированных kd деревьев
  - При выборе разбиения будем брать не параметр с наибольшим разбросом, а выбирать случайно параметр из набора параметров с наибольшим разбросом (например, из 5)
  - Порог разбиения тоже выбираем случайно недалеко от медианы
  - Для поиска ближайшего будем строить одну очередь (priority search) из листьев всех деревьев

C. Silpa-Anan and R. Hartley, “Optimized kd-trees for fast image descriptor matching,” CVPR, 2008

Approximate k-means (AKM)

- Раз иерархические K-средние сильно проигрывают обычным K-средним, построим алгоритм «приближенных K-средних»
  - Лес из 8 рандомизированных k-d деревьев с общим списком обхода
  - Будем использовать этот лес для поиска ближайших на одном из этапов K-средних
  - Сложность каждого этапа k-средних падает с O(NK) до O(Nlog(K))

J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, “Object retrieval with large vocabularies and fast spatial matching,” CVPR, 2007.

Сравнение K-means и AKM

Clustering parameters		mAP	
# of descr.	Voc. size	k-means	AKM
800K	10K	0.355	0.358
1M	20K	0.384	0.385
5M	50K	0.464	0.453
16.7M	1M		0.618

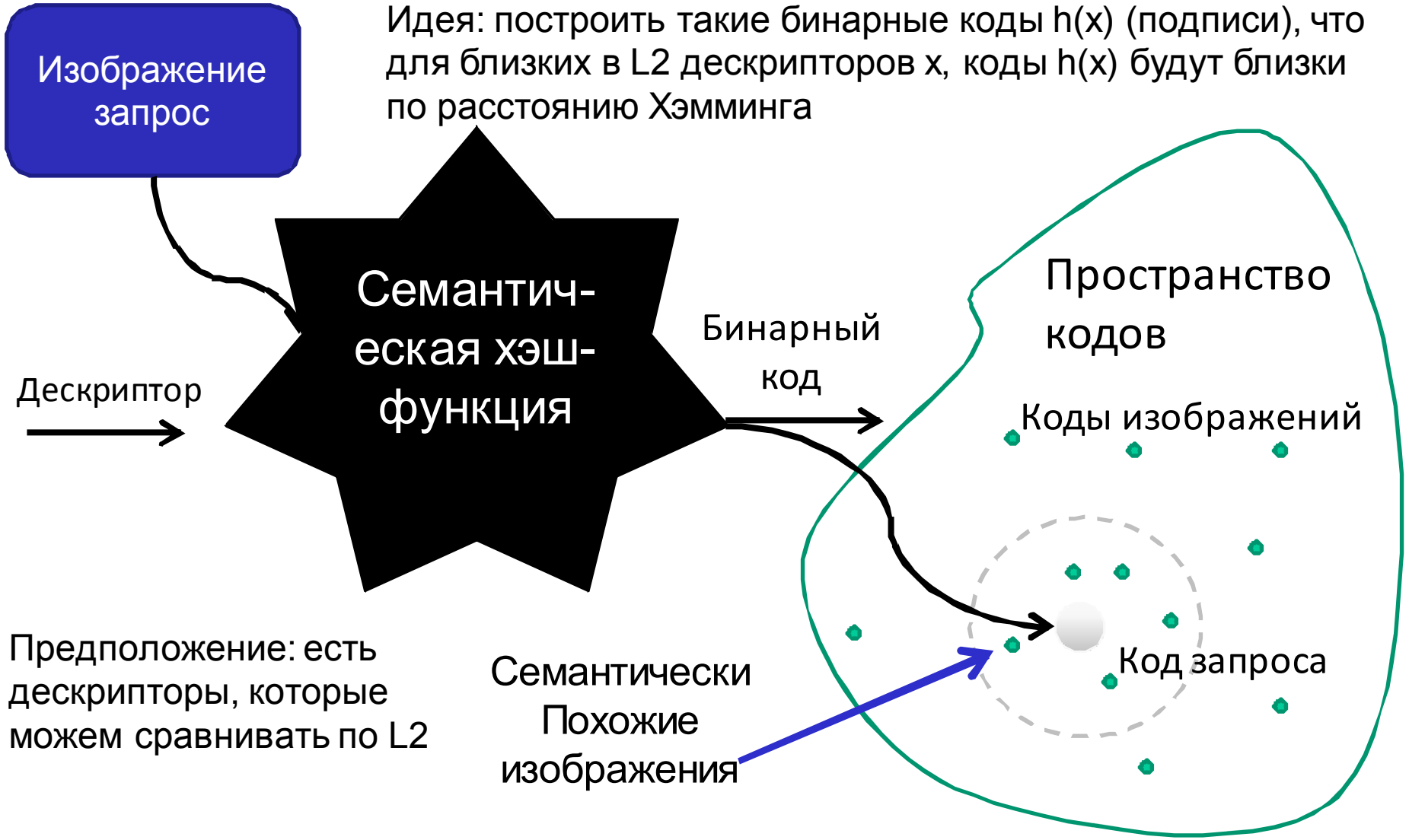
- Сравнение AKM с обычным K-means показывает небольшое падение точности (1%)
- На больших размерах выборки обычным K-means слишком долго считать

Сравнение AKM и HKM

Method	Dataset	mAP
		Bag-of-words
(a) HKM-1	5K	0.439
(b) HKM-2	5K	0.418
(c) HKM-3	5K	0.372
(d) HKM-4	5K	0.353
(e) AKM	5K	0.618
(f) AKM	5K+100K	0.490
(g) AKM	5K+100K+1M	0.393

- Выводы:
  - AKM превосходит по точности HKM
  - С ростом размера выборки точность сильно падает

Семантическое хеширование



R. R. Salakhutdinov and G. E. Hinton. Semantic hashing. In SIGIR workshop on Information Retrieval and applications of Graphical Models, 2007.

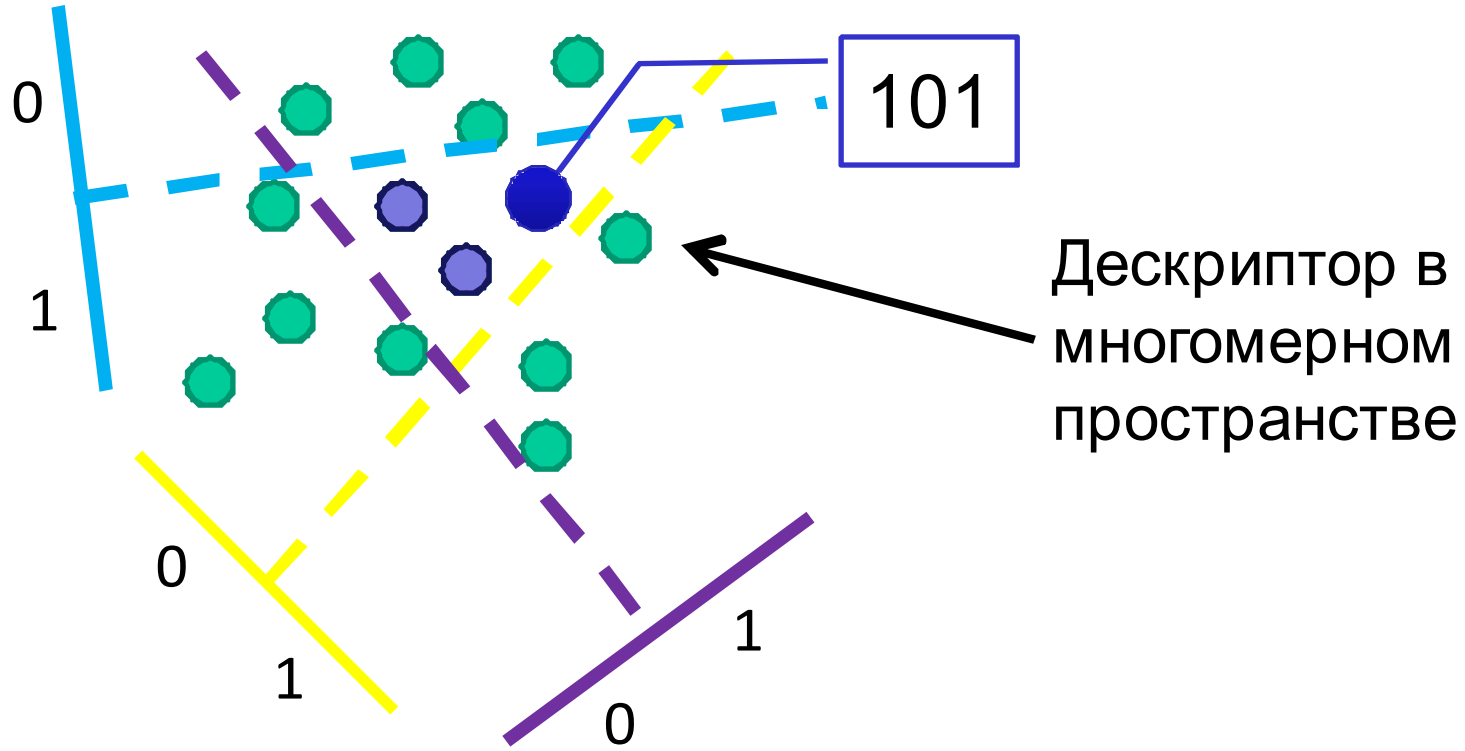
Формализация и методы

- Смысл:
  - Пусть x,y – вектор-признаки (дескрипторы)
  - $N_{100}(x)$ – ближайшие 100 векторов к *x* в исходном пространстве по L2
  - Хотим найти h(x) – бинарную подпись, такую что
    - $N_{h100}(x) = N_{h100}(y)$
    - Где $N_{h100}(y)$ – расстояние Хэмминга

- Методы:
  - LSH – Locality Sensitive Hashing
  - Обучение кодов (BoostCC, RBM)
  - Спектральное хэширование
  - Обучение метрики с помощью LSH

Locality Sensitive Hashing (LSH)

- Возьмем случайную проекцию данных на прямую
- Выберем порог близко к медиане проекций на прямую
- Пометим проекции 0 или 1 (1 бит подписи)
- С увеличением числа бит подпись приближает L2-метрику в исходных дескрипторах



A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In FOCS, 2006

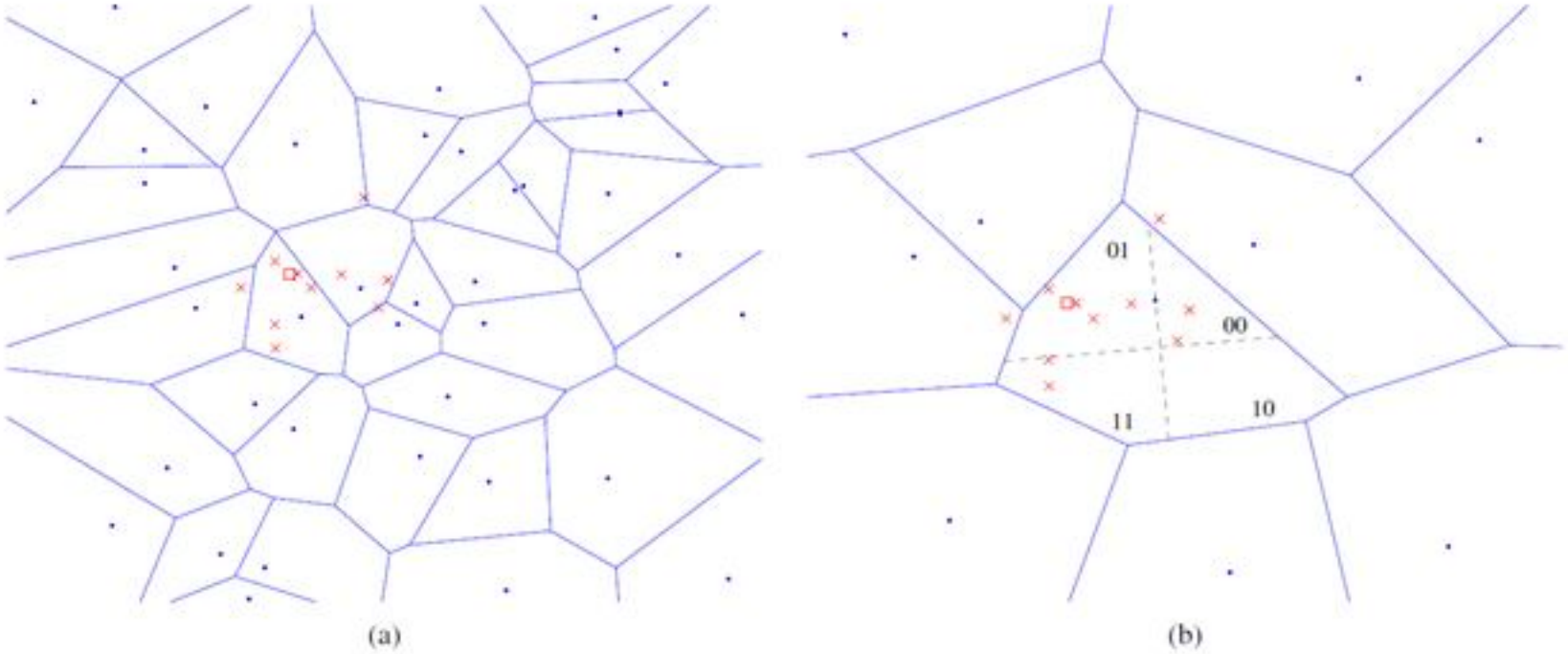
Locality Sensitive Hashing (LSH)

- Плюсы
  - Приближенный способ вычисления ближайшего соседа
  - Быстрое квантование (быстро вычисляем бинарный код)

- Недостатки:
  - Приближение L2 лишь асимптотическое
  - На практике может потребоваться слишком много бит для подписи

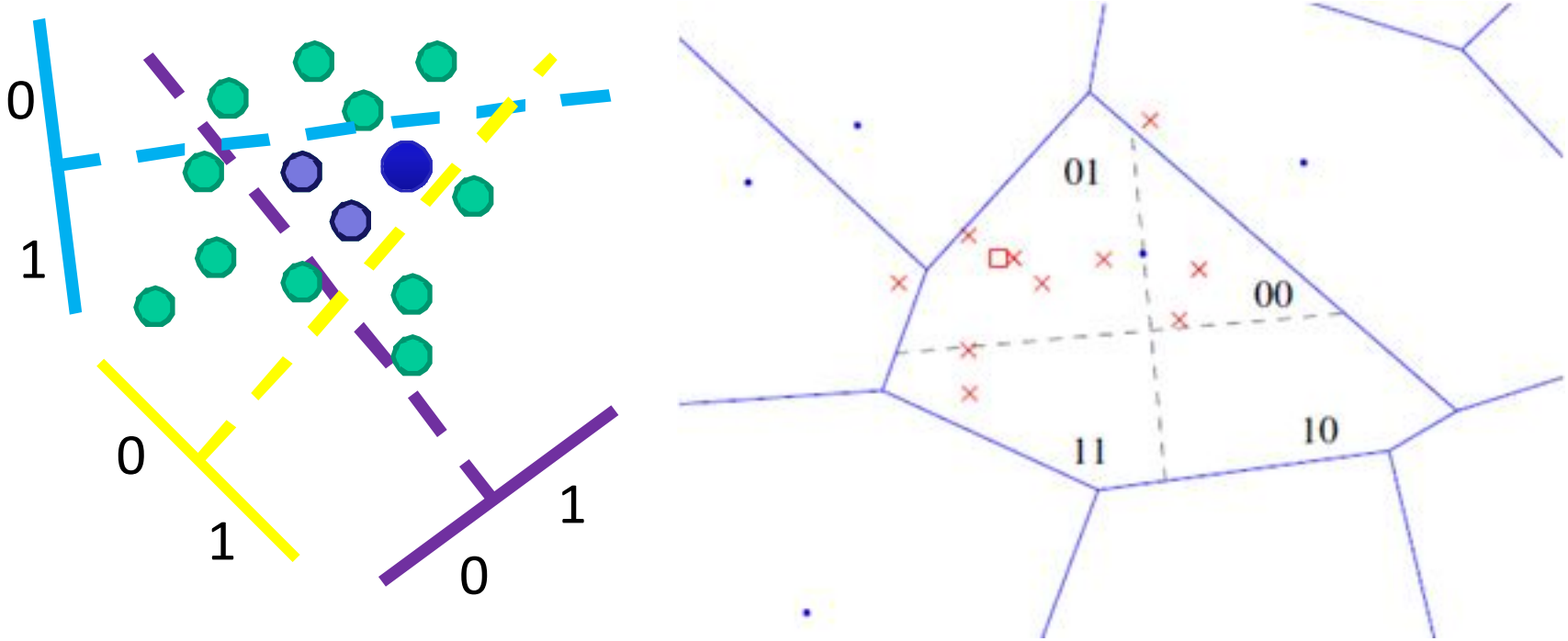
- Вывод:
  - Использовать как замену K-средних нельзя
  - Можно использовать в дополнение

Проблемы K-средних



- Маленькое K – большие ячейки
  - Слишком грубый порог на сравнение!
- Большое K – маленькие ячейки
  - Медленнее кластеризация и квантование
  - Ячейка может оказаться слишком маленькой, не все соседи попадут

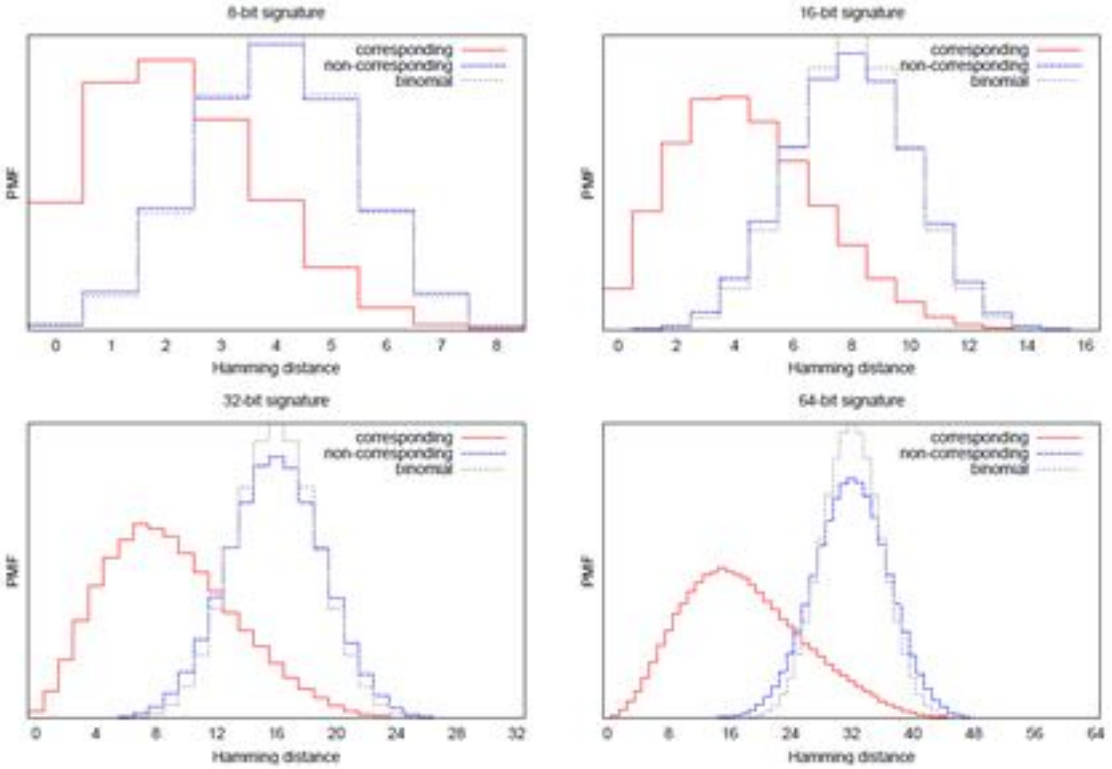
Hamming Embedding



Идея:

- Припишем каждому дескриптору бинарный код, задающий положение дескриптора внутри ячейки диаграммы Вороного
- Будем использовать эти коды для сравнения дескрипторов, попавших в одну ячейку
- Коды построим с помощью LSH для каждой ячейки независимо

Влияние кодов



- Посмотрим распределение расстояний Хэмминга между правильными и неправильными парами
- Сравнение по кодам показывает, что при длине кода > 32 бит наблюдается заметная разница между правильными и неправильными сопоставлениями

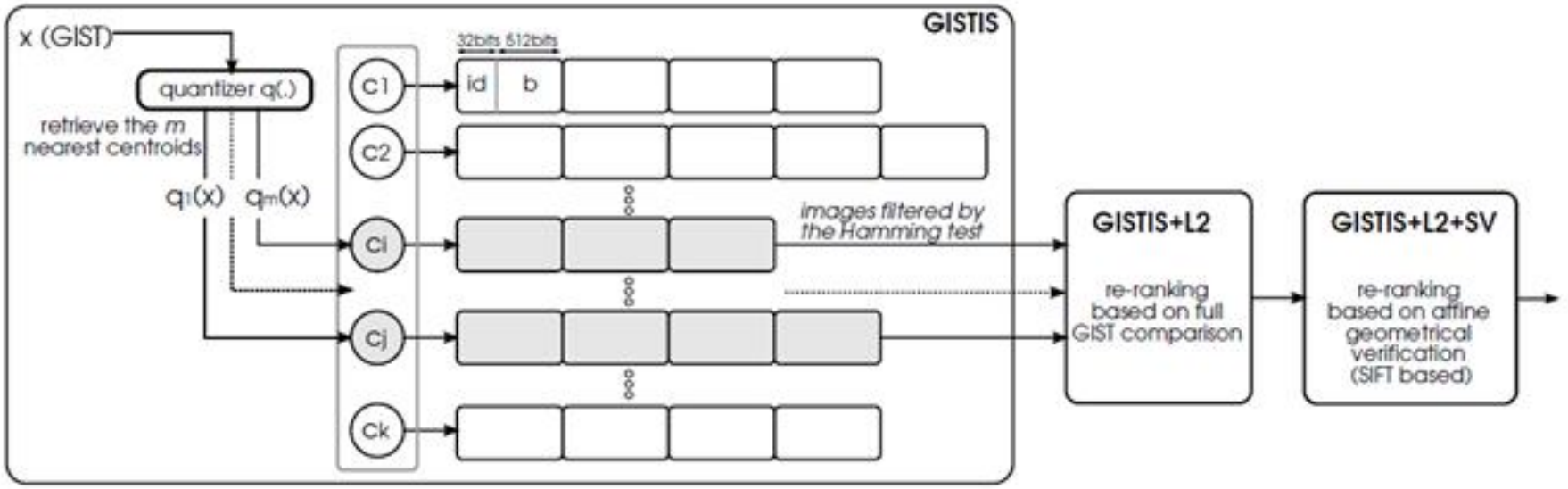
Алгоритм GISTIS

- GIST Indexing Structure
  - Добавим в наш простой алгоритм для поиска полудубликатов hamming embedding

- Схема метода:
  - Строим GIST для каждого изображения
  - Кластеризуем все дескрипторы с помощью k-means на k=200 слов
  - Применяем LSH к каждому кластеру и для всех векторов в кластере считаем бинарную подпись
  - Идентификатор картинки и бинарная подпись хранится в индексе в RAM

M.Douze et.al., Evaluation of gist descriptors for web-scale image search. In International Conference on Image and Video Retrieval. ACM,2009.

Схема метода



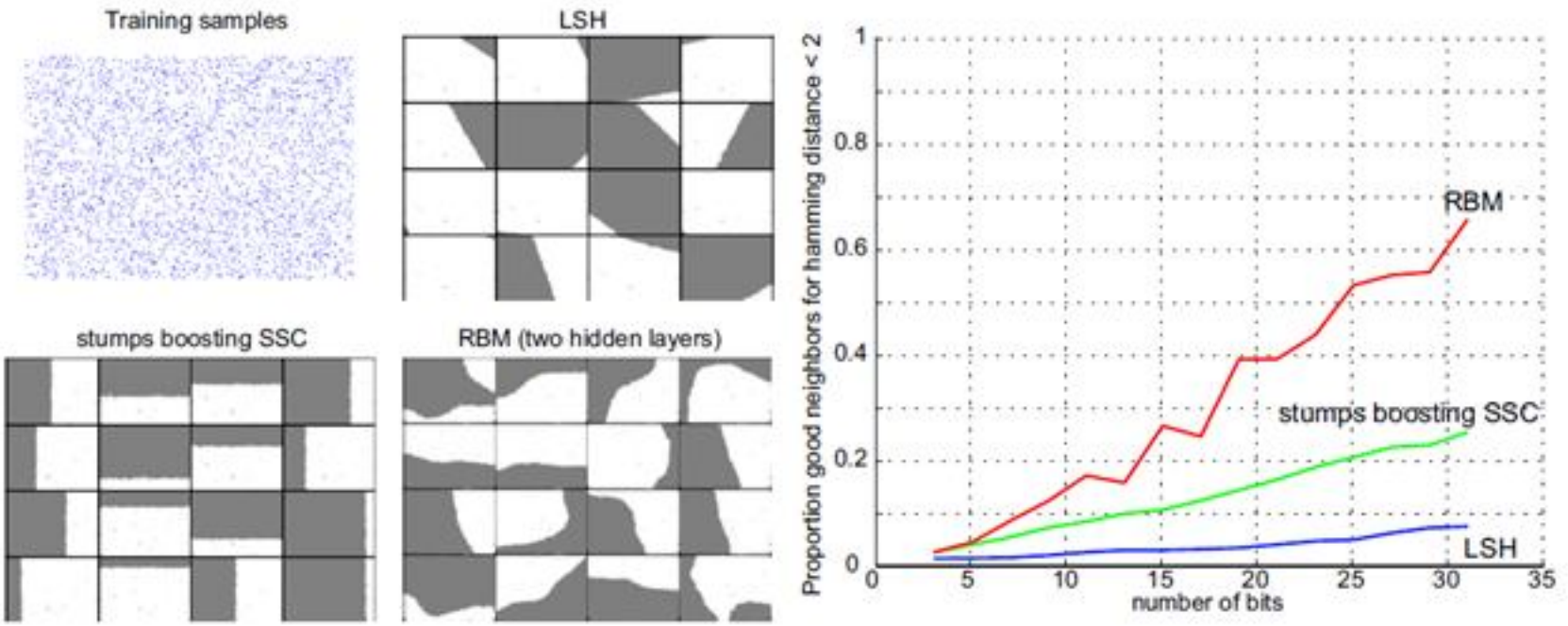
- В индексе в RAM хранится только бинарная подпись изображения (512 бит) и идентификатор
- Сам GIST хранится на жестком диске
- Можем проводить сортировку несколько раз:
  - Вначале по бинарным подписям
  - Затем по GIST с жёсткого диска

Результаты

	Байт на изображение в RAM	Время на построение дескриптора	Время поиска в базе из 110M изображений
GIST	3840	35мс	1.26с
GISTIS	68	36мс	2мс
GISTIS + L2	68	36мс	6/192мс

M.Douze et.al., Evaluation of gist descriptors for web-scale image search. In International Conference on Image and Video Retrieval. ACM,2009.

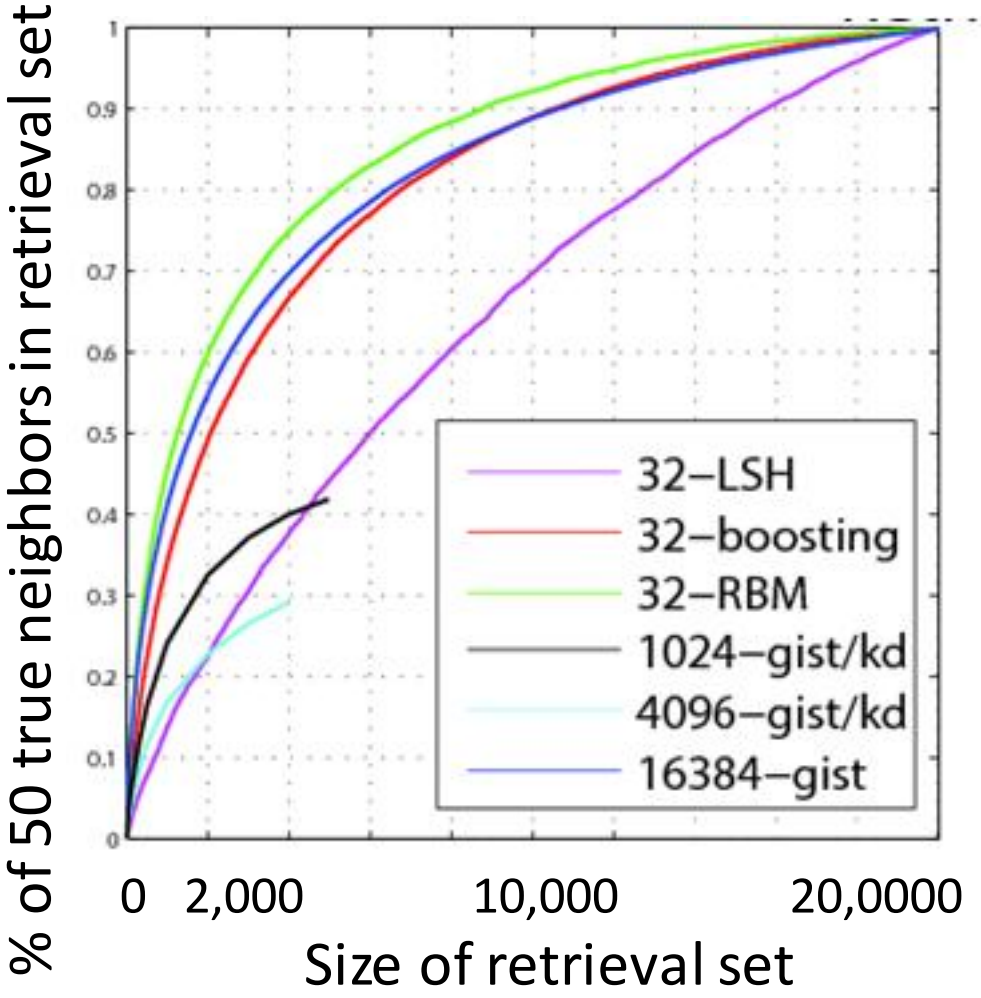
Обучение кодов



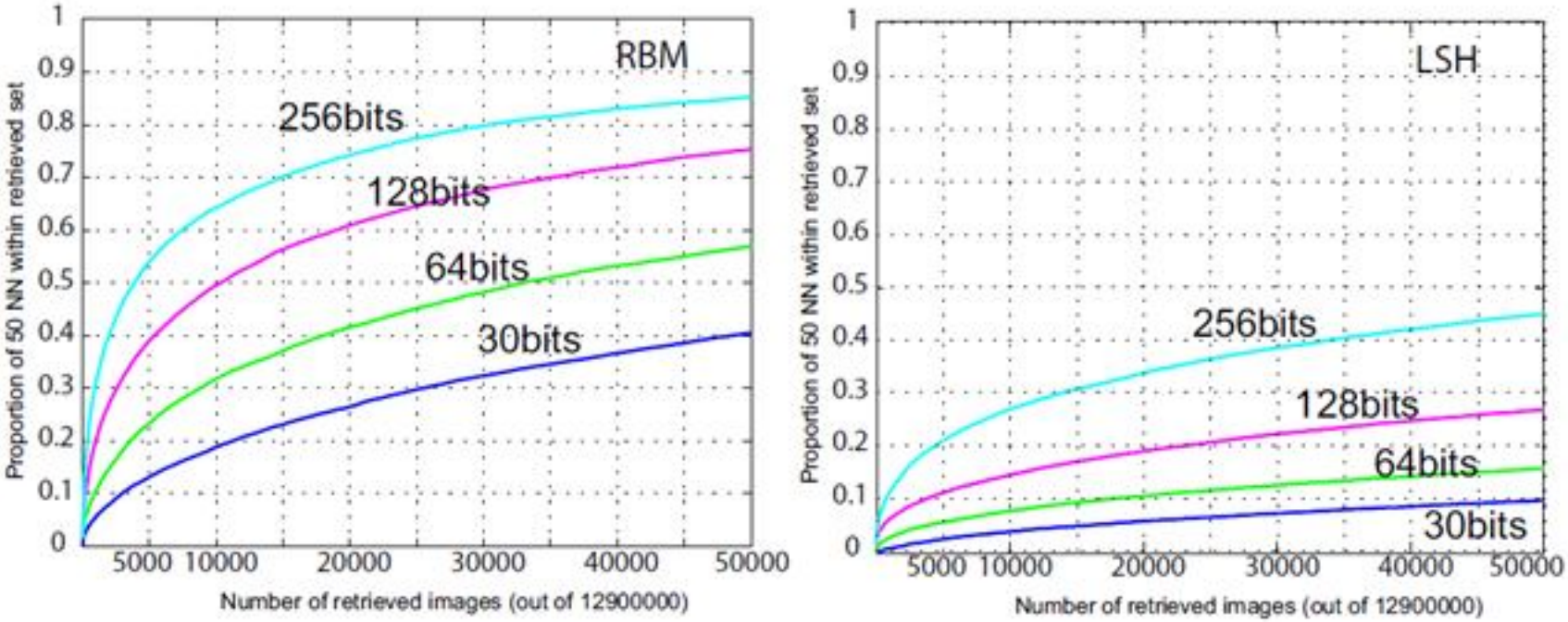
- Рассмотрим синтетический пример – точки в прямоугольнике
- BoostingCC:
  - Разбиение по порогу по одной из координат («простые признаки»)
  - Поиск оптимальной комбинации таких простых разбиений
- RBM – Restricted Bolzman machines
  - Вариант стохастических «нейронных» сетей
  - По сути, простая «Марковская сеть» (Вторая часть курса)

Сравнение на LabelMe

- 32-битные коды работают так же хорошо, как исходный дескриптор в 512 вещественных чисел
- Методы на основе обучения обгоняют LSH



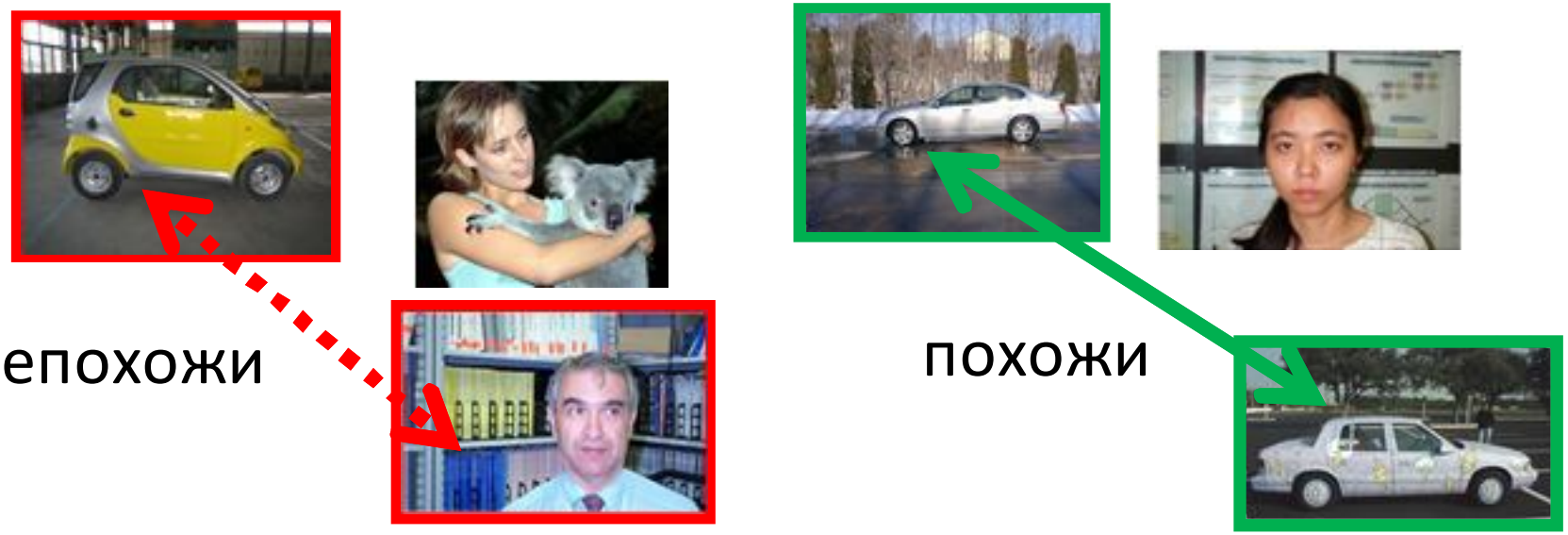
Сравнение на web



- Сравнение на 12.9M изображений

Пример работы

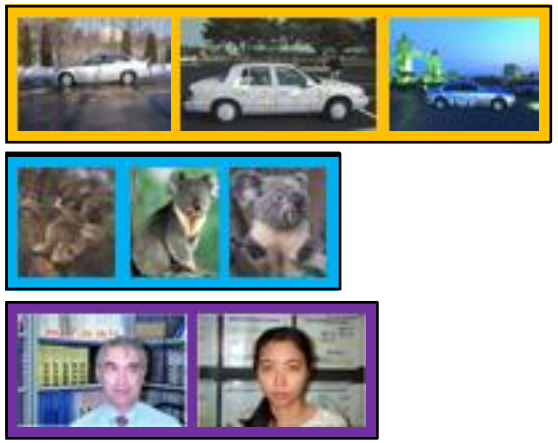
Обучаемые метрики



- Что если для наших дескрипторов Евклидово расстояние плохо подходит для описание близости изображений?
- Модифицируем процедуру построения бинарных кодов так, чтобы они учитывали априорно известную информацию, какие изображения похожи, а какие нет

Когда применимо?

Частично-размеченная база изображений

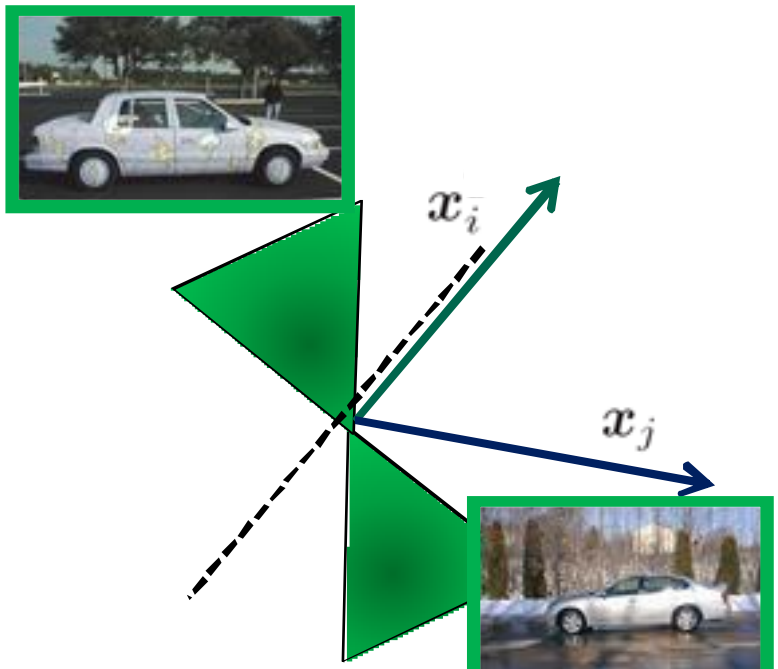


Полностью размеченная база

Специфическая проблемная область

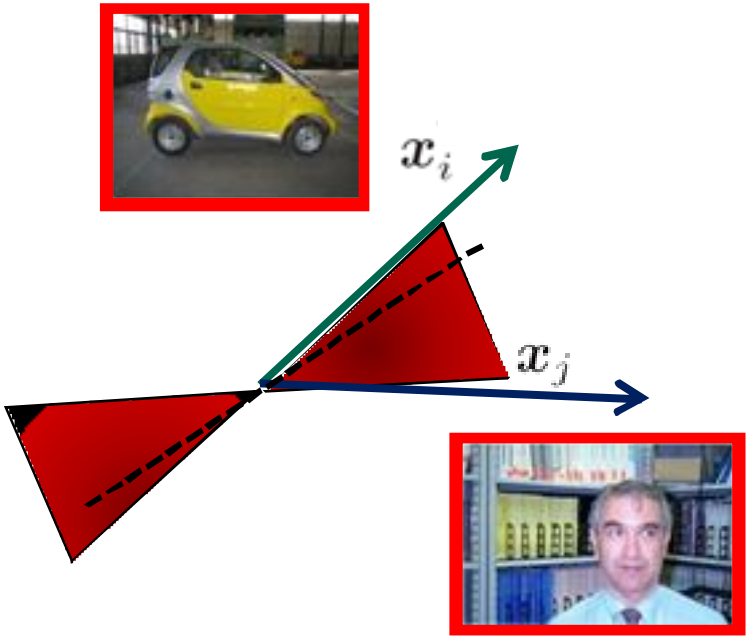
Обучение расстояния с LSH

Будем выбирать линии проекции «не случайно»



$$h(x_i) = h(x_j)$$

С меньшей вероятностью разбиваем подобные пары с ограничением сходства



$$h(x_i) \neq h(x_j)$$

С большей вероятностью разобьем пары с ограничением несходства

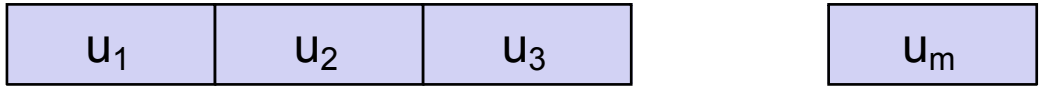
Результаты

- Сравнение на tiny images (80M)
- Обученная метрика позволяет найти то же количество правильных соседей, что и обычных LSH, просмотрев меньше 1% базы
- Скорость – 0.5с вместо 45с

B. Kulis and K. Grauman. Kernelized locality-sensitive hashing for scalable image search. In *ICCV, 2009.*

Product quantization

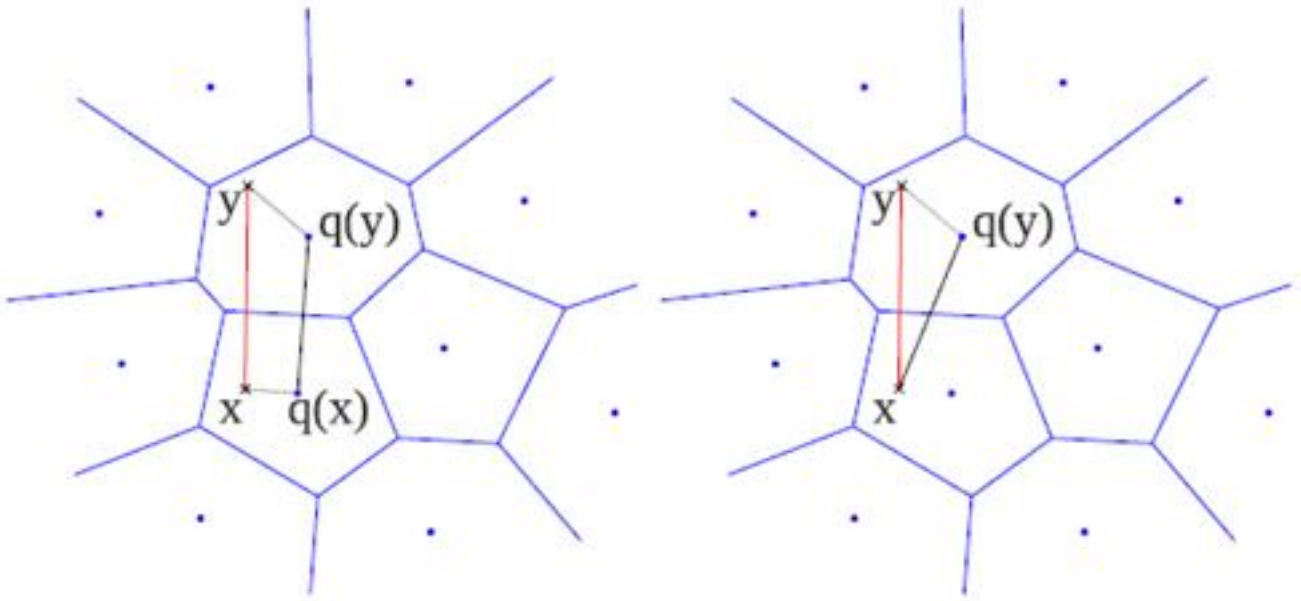
- Вспомним:
  - Для векторов SIFT квантование до кодов 64 бита (0.5 бита на параметр) требует подсчёта и хранения 2^{64} центроидов, что невозможно
- Простая идея:
  - Разобьём вектор x длины D на m частей
  - Квантуем каждый подвектор u_j независимо от остальных



  - Пусть каждый подвектор квантуем на k* центроидов, тогда всего центроидов $(k^*)^m$
  - Длина кода $l = m \log_2 k^*$
- Сравнение с K-средними по памяти
  - Память: kD (K-средние) и $mk^*(D/m)=k^{1/m}D$

H.Jégou et. al. Product quantization for nearest neighbor search, PAMI 2011

Сравнение векторов



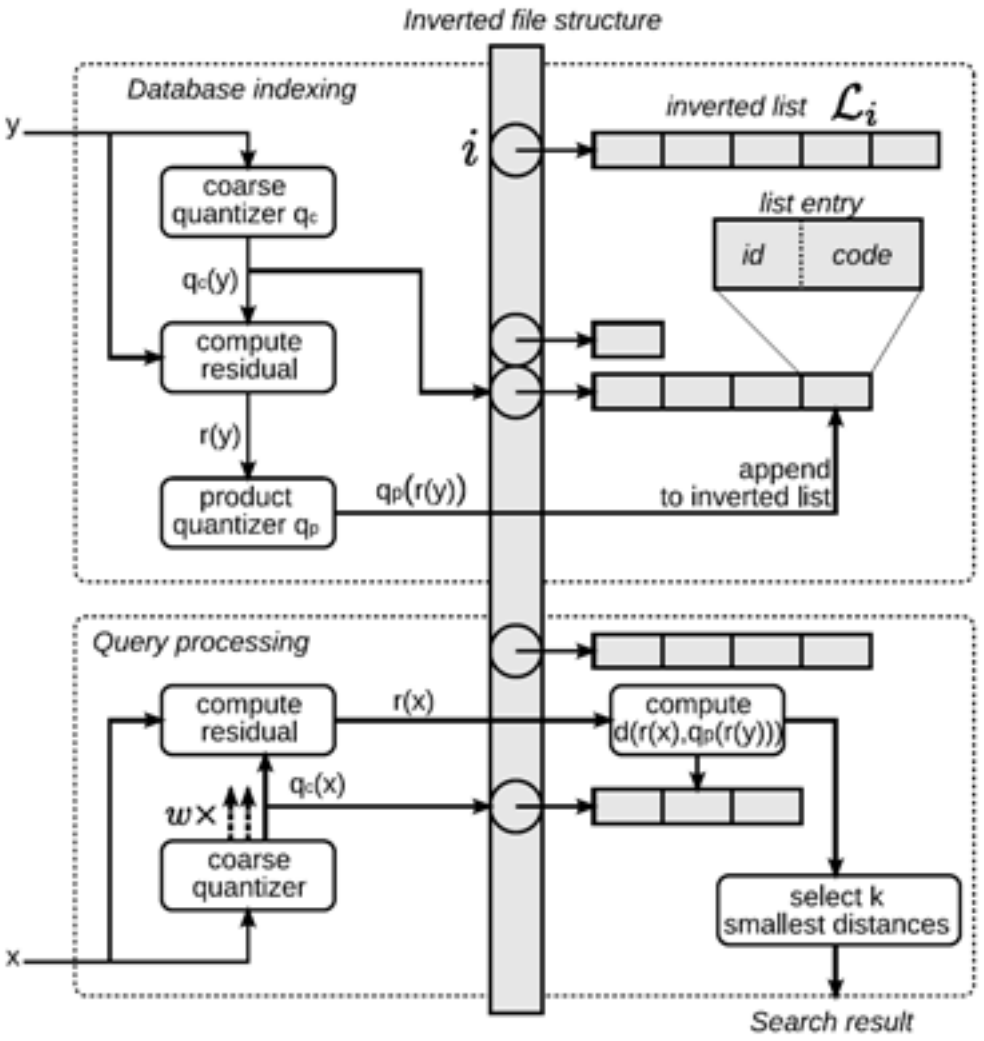
symmetric case asymmetric case

- Можем применять квантование произведения и к записям в базе, и к запросу (симметричное сравнение)
- Можем сравнивать вектор-запрос с реконструированной версий квантованных векторов (ассиметричное сравнение)
- Ассиметричный вариант точнее

Инвертированный индекс

Воспользуемся схемой «hamming embedding»:

- Сделаем «грубое» квантование
- Построим по нему инвертированный индекс
- Для каждого вектора y вычислим ошибку r = y – q(y)
- Применим «квантование произведения» к ошибке
- Допишем к записи каждого вектора код
- Используем эти коды для уточнения поиска в списке (ранжирование)



Резюме поиска ближайшего

- Для векторов большой размерности (SIFT, GIST) обычные быстрые методы поиска соседей (kd-деревья) работаю неточно и долго
- Есть целый ряд хороших подходов для приближенного вычисления на основе квантования:
  - Иерархические K-средние
  - Рандомизированные kd-деревья и приближенное k-среднее
  - Семантическое хеширование и обучение бинарных кодов
  - Квантование произведения