ITMO

Images Search

Computer Vision

Images Search. Outline

- Search tasks
- Indexing
- Near-duplicates
- Search for objects
- Results expansion

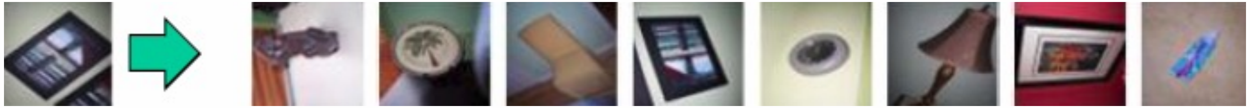
Images Search. Task Statement

- Content-based image retrieval
- Search for images with specific content in the image database
- The task is similar to image recognition, but it focuses mainly on scaling and interaction with the end user

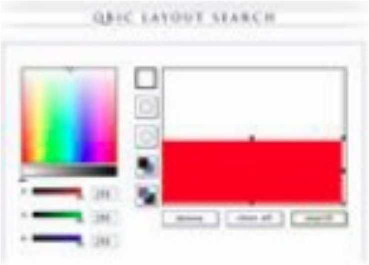
Datta, Ritendra, et al. "Image retrieval: Ideas, influences, and trends of the new age." *ACM Computing Surveys (Csur)* 40.2 (2008): 1-60.

Images Search. A user request

1. Text request
   - Example: “The Great Wall, photos” – image annotation
   - database categorization is required
2. Sample image (find the same or similar one)

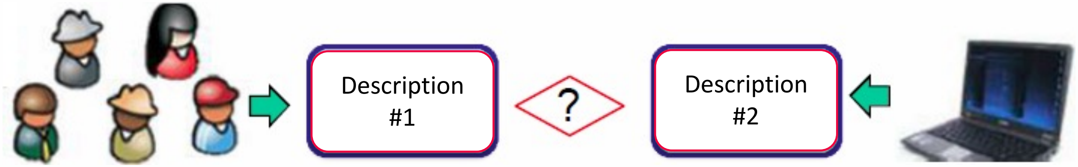


3. Query as content features
   - Example: color histogram



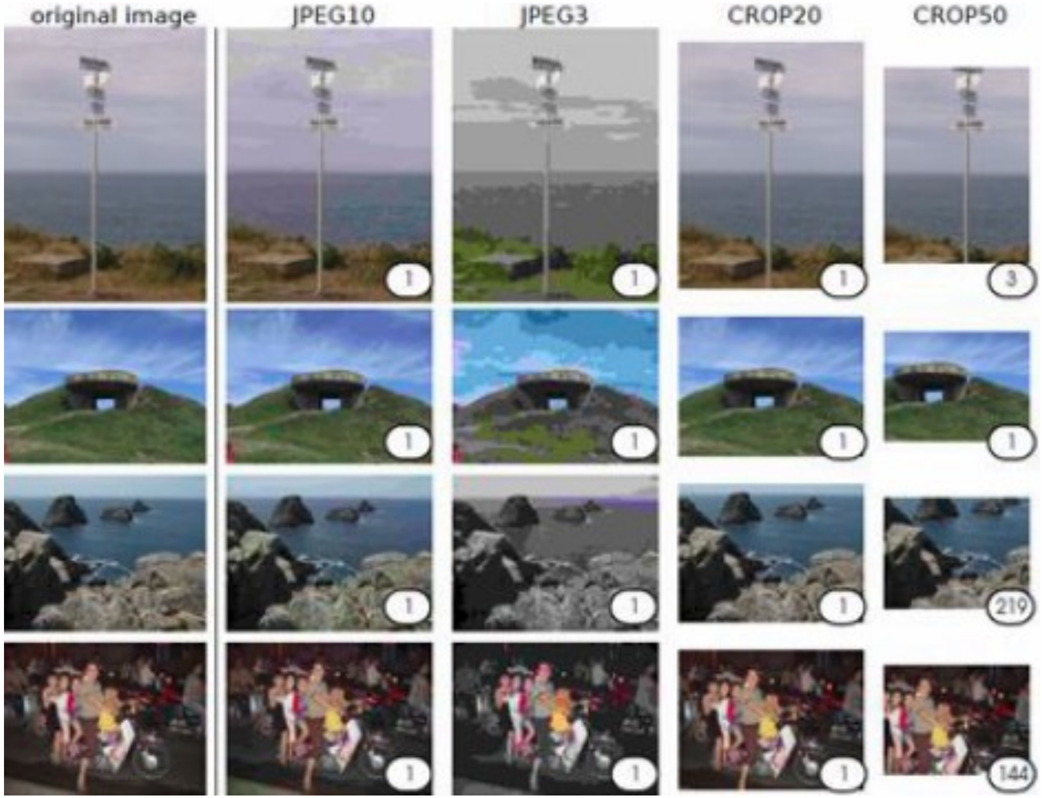
Images Search

- What is a **“Similar image”**?
- **Semantic gap** – the mismatch between the information that can be extracted from visual data and the interpretation of this data by the user



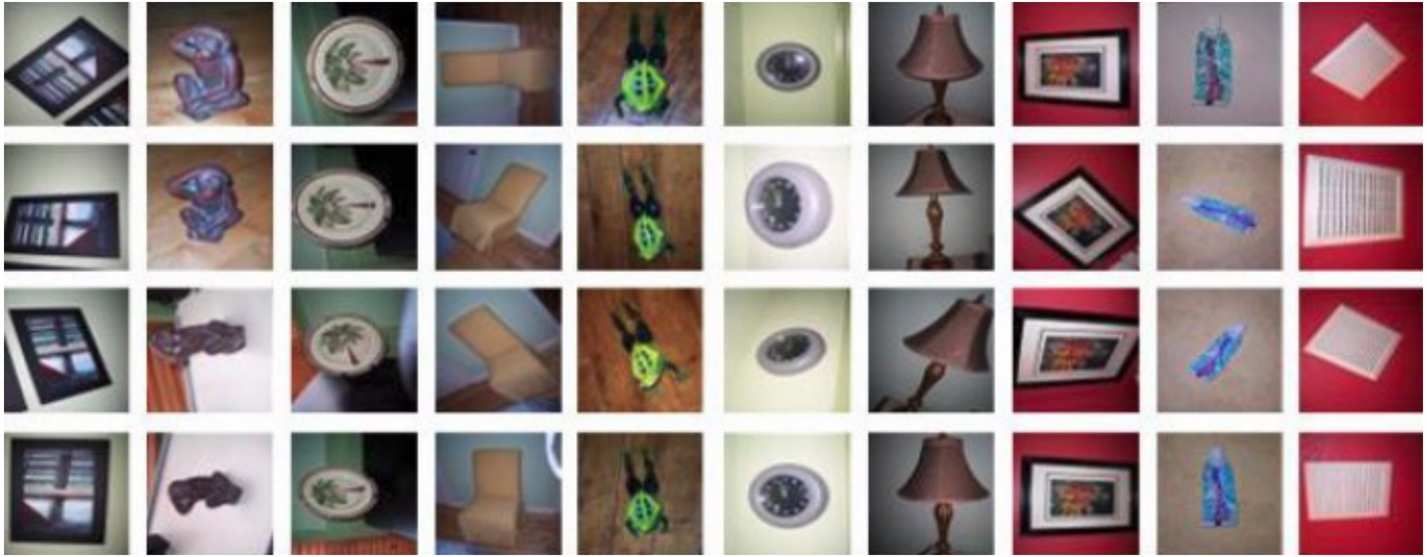
Images Search. Similar Image Type 1

- **Near-duplicates**
  - slightly modified version of the same image (different angle, colors, size, etc.).



Images Search. Similar Image Type 2

- **The same object or scene (object retrieval)**
  - strong variations in angles, backgrounds, and other changes compared to near-duplicates



Images Search. Similar Image Type 3

- **Scenes with similar configuration**
  - may have the different purpose

Kitchen

Bathroom

Bar

Bus

Airplane

Conference room

Images Search. Similar Image Type 4

- **Images of the same category (category-level classification)**
  - scenes or objects

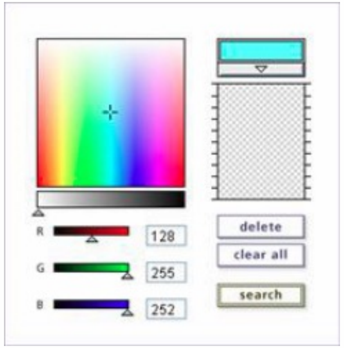
Banqueting hall

Images Search. Task Statement

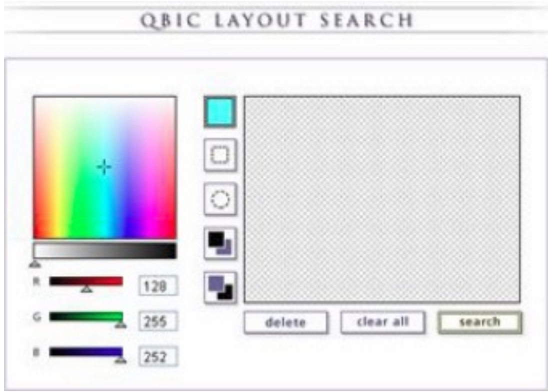
- All 4 tasks have different task statements
- They require different algorithms to solve them.
- **QBIC «Query By Image Content»** (1995) – the very first image content search engine:
  - Calculates the following feature sets of objects:
    - Feature colour histograms
    - Area, perimeter, etc.
  - Binary mask is used to describe objects
  - Segmentation of objects is carried out manually or automatically:
    - Highlighting contrasting objects against the uniform background (museum exhibits)
    - Flood-fill and “snakes” methods
  - Database contains about 10 000 images

Flickner, Myron, et al. "Query by image and video content: The QBIC system." *computer* 28.9 (1995): 23-32.

QBIC Example



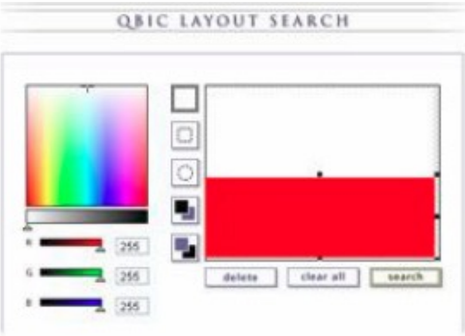
Histogram



Spatial color distribution

Flickner, Myron, et al. "Query by image and video content: The QBIC system." *computer* 28.9 (1995): 23-32.

QBIC Example



Flickner, Myron, et al. "Query by image and video content: The QBIC system." *computer* 28.9 (1995): 23-32.

Image search. General Workflow

iTMO

1. Building image index

Images database → Feature extraction → Signature → Images database index

Descriptor based on extracted features

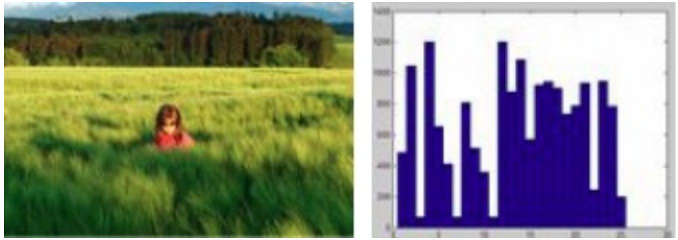
2. Search

Image

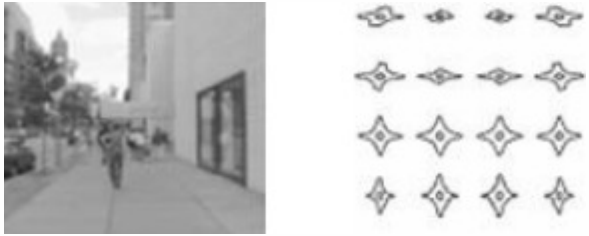
Image → Feature extraction → Signature → Index search → List of similar images

Image Descriptors

ITMO

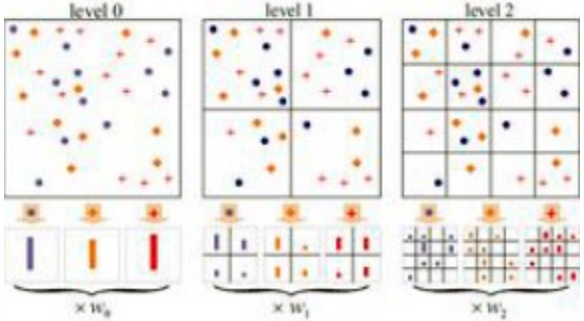


Color histograms



Gradient histograms (GIST / HOG)

Bag of words
(Bag of features)



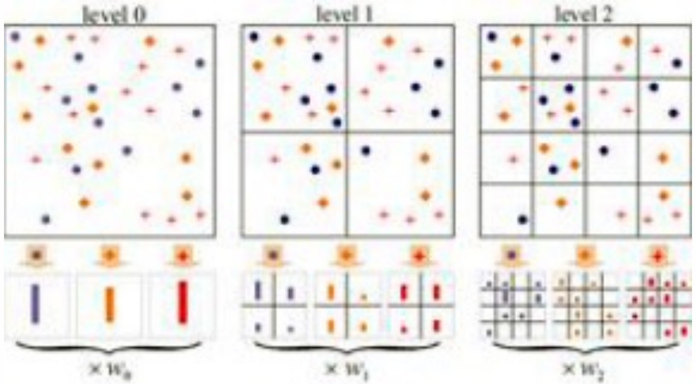
A spatial bag of words
and individual features

Required Computation Resources

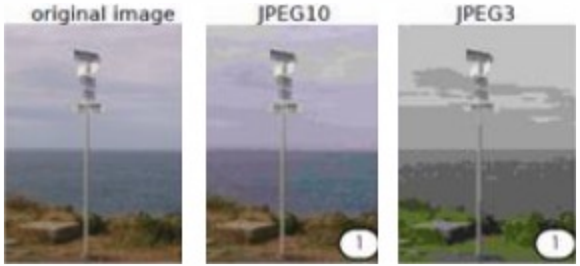
- Desired number of images in the database: billions
- Example: GIST descriptor:
  - 4 x 4 grid with 8 orientations and 4 scales: 4 * 4 * 8 * 4 = 512 parameters
  - If single parameter is 4 bytes: 2048 bytes for image = 2 KB
  - With a collection of 1 million images: descriptors would occupy 2 GB
- Conclusion: a simple descriptor usage requires a lot of memory

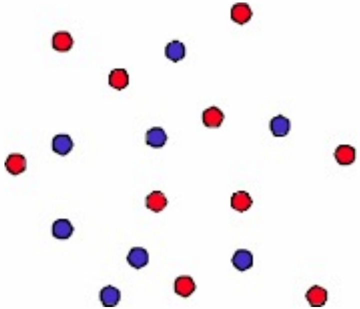
Required Computation Resources

- Example: Bag Of Words (Bag Of Features):
  - Dictionary size is from 200 to 1 million words
  - Up to 1 million parameters in single histogram
  - Up to 4 MB per image
- Example: Pyramidal Bag Of Words:
  - Pyramid of three levels - 21 histograms
  - BOW * 21 = up to 80 MB per image
- Example: object filters:
  - 200 main classes;
  - Pyramid of three levels - 21 histograms
  - 21 * 200 * 4 bytes = 16 KB;
  - 1 million images would require 16GB for descriptors



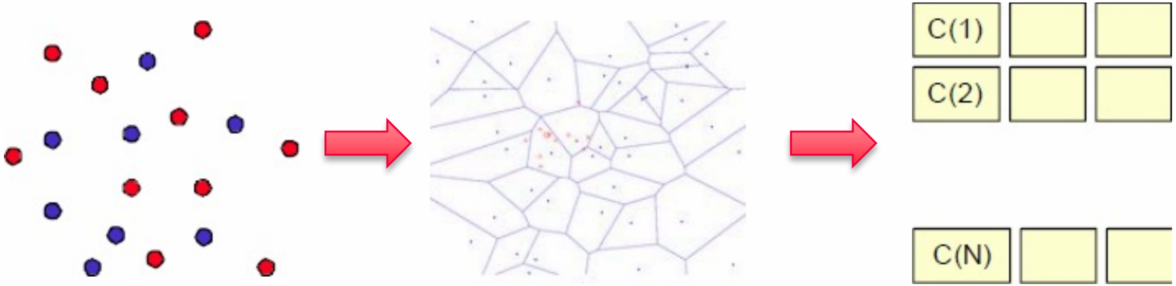
Search for the Nearest Neighbour





- It is necessary to find the Nearest Neighbors by descriptor in the entire collection
- The simplest index is a linear list of all descriptors
- Full search – comparison of the test vector with each example from the collection
  - C = 1M images, GIST = 512 parameters, C * GIST = 512M operations
- Approximate methods for finding neighbours are required
  - Approximate nearest neighbor

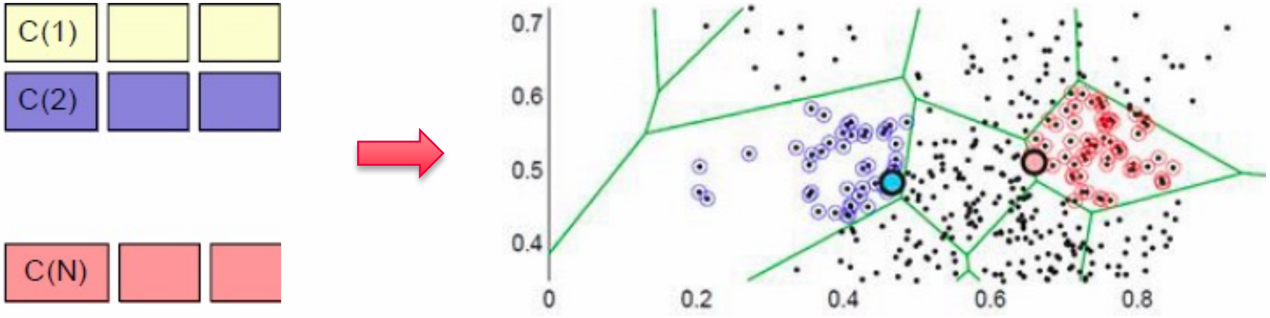
Inverted Index



- Execute clustering (quantize) of all descriptors to get K-clusters
- Split the entire collection into clusters
- Build the «inverted index»:
  - List of clusters (with corresponding GIST)
  - Each cluster stores the indices of images that belong to the cluster

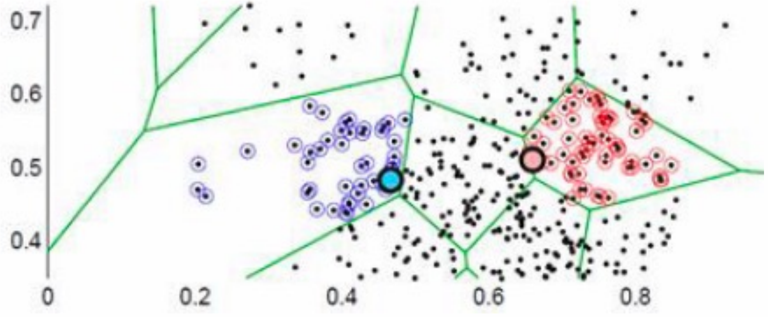
Inverted Index

ITMO



- Search by index:
    - Check the inverted index to find the nearest cluster
    - All elements in the nearest cluster list are the nearest vectors (approximately)

Ranking of Results



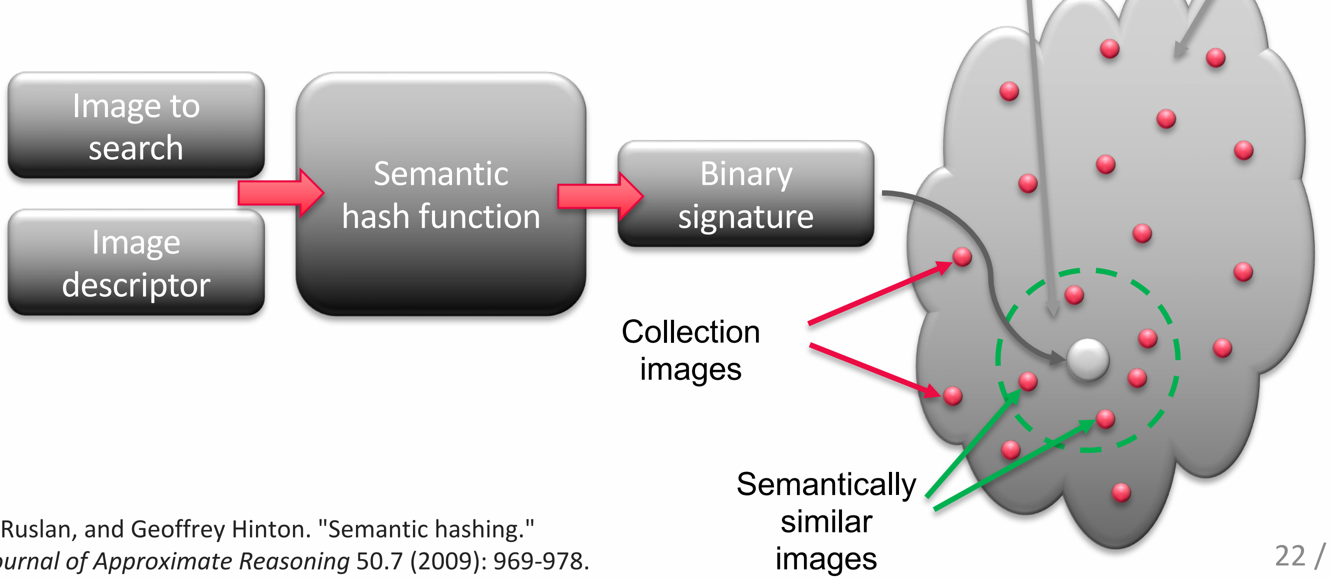
- To improve the accuracy, we will can reorder results from the cluster:
  - Calculate the distances to each element of the list by their full descriptors
  - Re-rank results basing on the proximity (the nearest ones first)

Simple Approach

- **Indexing:**
  - Calculate GIST for each image
  - Quantize of the descriptors to get 200 clusters
  - Building an inverted index of clusters
  - We store the entire GIST in memory
- **Search:**
  - Calculate GIST for image
  - Find the nearest cluster in the inverted index by comparing against GIST
  - Re-rank the list from the index by GIST

Semantic Hashing

- Allows you to speed up a simple algorithm and increase the size of the collection
- Idea: build such short binary signatures for images that if images close in L_2 they will have close signatures



We assume that there is a description of the images that can be compared by L_2.

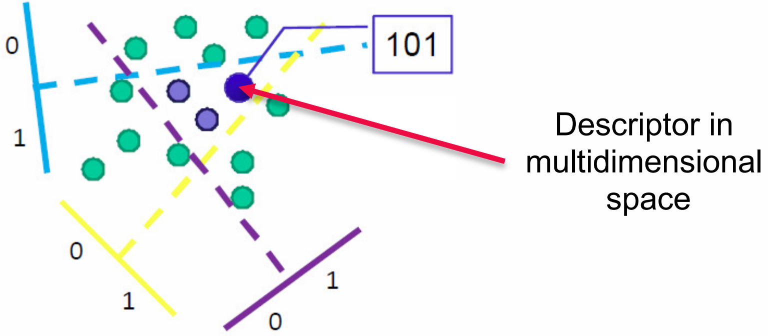
Salakhutdinov, Ruslan, and Geoffrey Hinton. "Semantic hashing." *International Journal of Approximate Reasoning* 50.7 (2009): 969-978.

Formalize

- We have vector descriptors x, y
- Need to get the binary signature $h(x)$ for image search:
  - If $\mathrm{x} \approx y$ then $h(x) \approx h(y)$
  - $h(x)$ – semantic hash function (*binary signature*)

Locality Sensitive Hashing (LSH)

- Let's take a random projection of data onto a straight line
- Randomly choose a threshold and mark the projections by 0 or 1
  - 1 signature bit
- With an increase in the number of bits, the signature approximates L_2,
  the metric in the original descriptors
- Disadvantages:
  - L_2 Approximation is asymptotic
  - The implementation may require too many bits for the signature



Andoni, Alexandr, and Piotr Indyk. "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions." *2006 47th annual IEEE symposium on foundations of computer science (FOCS'06)*. IEEE, 2006.

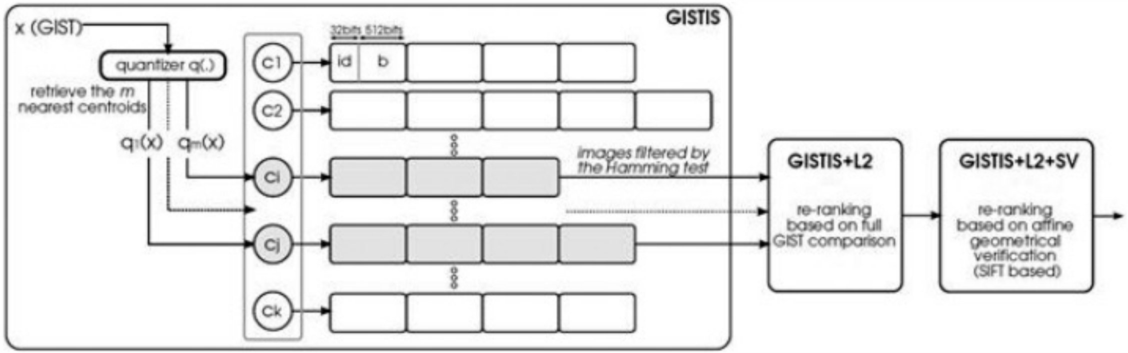
GIST Indexing Structure (GISTIS)

- Build a GIST for each image
- Quantize all descriptors using the k-means method for k=200 words
- For each cluster, calculate the binary signature using LSH
- Image indices and binary signatures (512 bits) are stored in an index in random access memory (RAM);
- GISTs are stored on the hard drive
- Can execute sorting multiple times
    - At first by binary signatures
    - Then by GIST from the hard drive

Douze, Matthijs, et al. "Evaluation of gist descriptors for web-scale image search." *Proceedings of the ACM International Conference on Image and Video Retrieval*. 2009.

Image Search

Workflow



Results

110 millions images

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

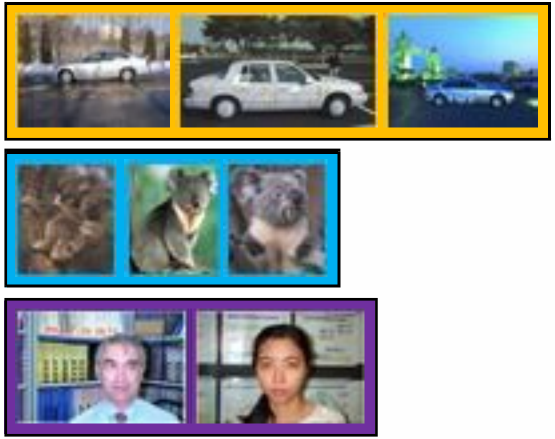
Trainable Metrics

- In case if the Euclidean metric L_2 is not suitable and it is difficult to choose the correct metric, then it can be trained

Trainable Metrics

- Training set

Partially tagged image database

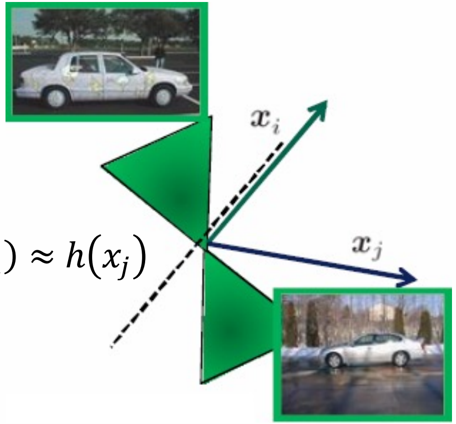


Tagged image database

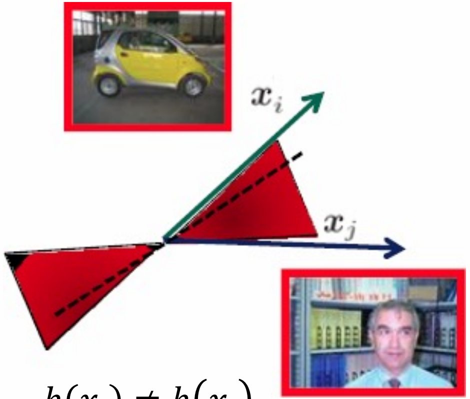
Trainable Metrics

- Distance learning via LSH

$h(x_i) \approx h(x_j)$



Split pairs of similar images
with less probability

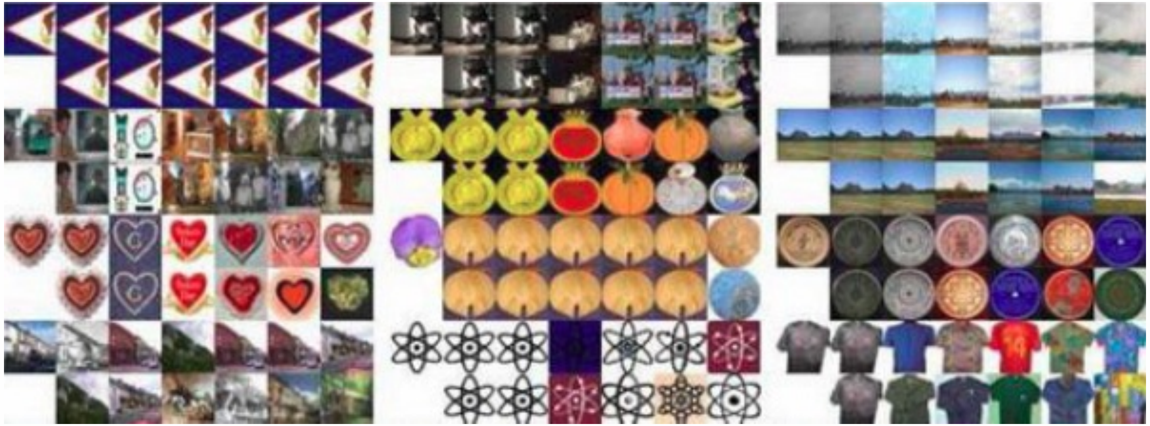


$h(x_i) \neq h(x_j)$

Split pairs of not similar images
with higher probability

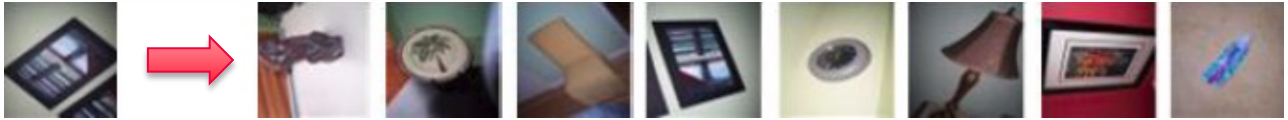
Results

- Comparison is performed on 80 million small images
- The trained metric allows you to find the same results, but with accessing less than 1% of the database
- Execution time is 0.5 second instead of 45 seconds



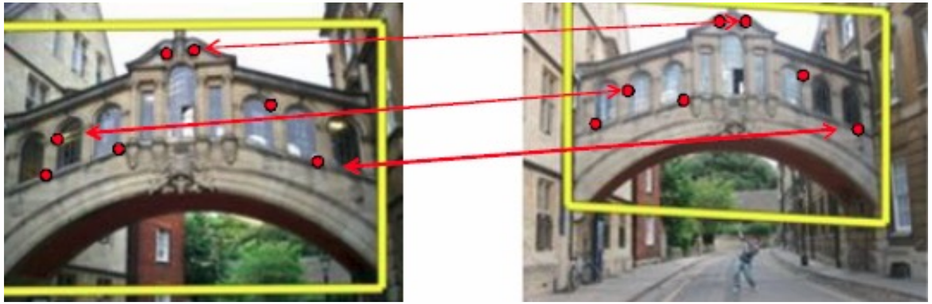
Kulis, Brian, and Kristen Grauman. "Kernelized locality-sensitive hashing for scalable image search." *2009 IEEE 12th international conference on computer vision*. IEEE, 2009.

Search for Same Objects or Scenes



Search for Same Objects or Scenes

- Sime straightforward approach:
  - Find image feature points (Harris, SIFT, SURF)
  - Calculate descriptors for found feature points (SIFT)
  - Match feature points by descriptors
  - Use the Random Samples Consensus method(RANSAC) to find the image transformation, reject the false matches;
  - If more than K matches were found then images are considered similar



Search for Same Objects or Scenes

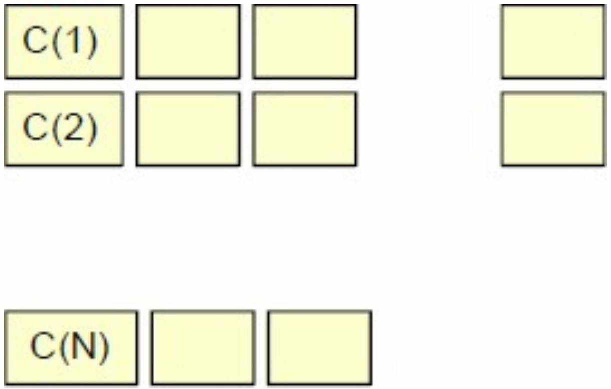
- Advantages:
  - Quality
- Disadvantages:
  - If we had N points, then each of them has 2 $(x, y)\ +\ 128$ (SIFT) parameters – **requires too much RAM**
  - Matching descriptors of all points by SIFT with some metric (e.g., using L_2 metric) – **will take a very long time**

Speed Up Feature Matching and Reduce Index Size

- Decrease the size of the descriptor to describe the local feature
- Dictionary quantization:
  - Build a dictionary of feature descriptors
  - Quantize the features by replacing the descriptor with an index in the dictionary
  - Modify the metric to compare descriptors:
    - Features are similar (distance is 0) if index is the same
    - Features are not similar (infinite distance) if index is not the same
- Matching can be even more simplified:
  - Let use describe an image with Bag Of Words
  - The image matching can be calculated as the intersection of histograms (Bags Of Words)

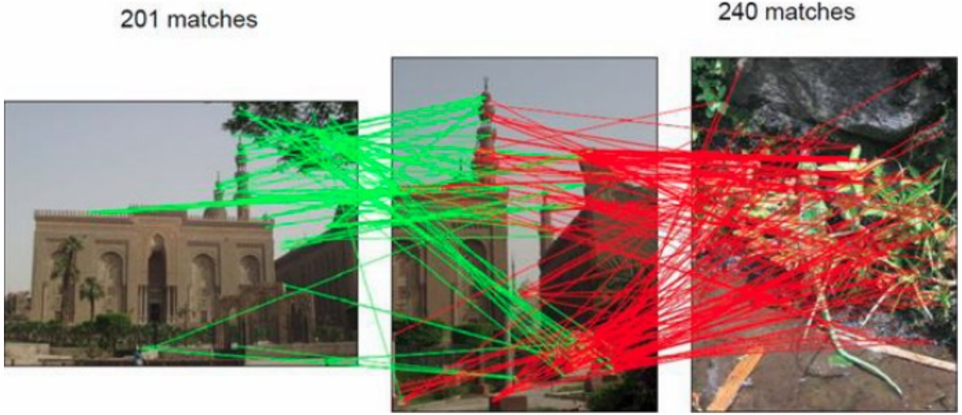
Inverted Index

- The vector of words in the descriptor is very sparse
  - For example, it may have 1K non-zero elements of 1M dictionary
- It's convenient to store it in an inverted index
  - Table (words) x (images)
  - List of words in dictionary (features)
  - For each word, we store a list of images in which the “word” occurs
- Search acceleration:
  - The most frequent words are at the top of the list



Side Effect of Quantization

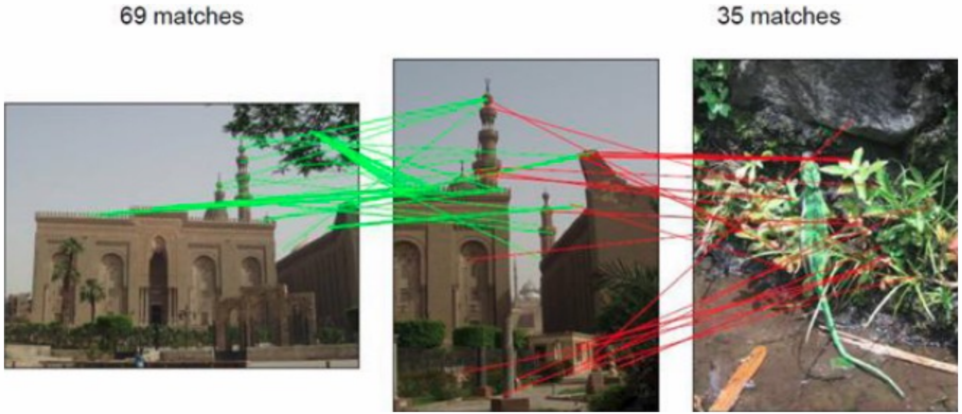
- 20 000 words dictionary



- Let's compare the features in two pairs of images according to the dictionary, and see the effect of the size of the dictionary
- The more words in the dictionary, the more accurate the representation of descriptors

Side Effect of Quantization

- 200 000 words dictionary



- Increasing the dictionary size increases the accuracy of image matching

Algorithm Requirements

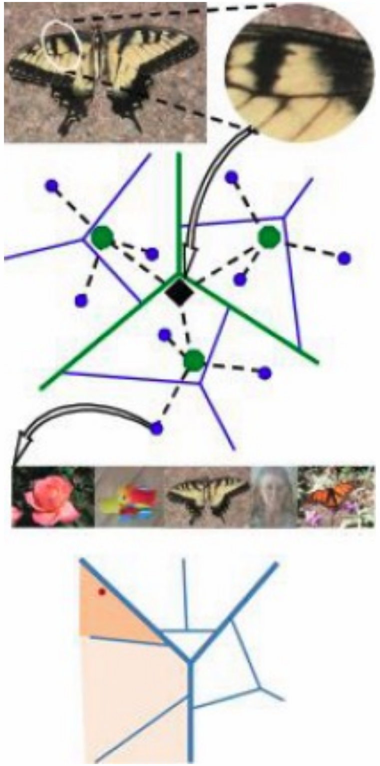
- Quickly build a dictionary
- Quickly quantize features
- Reduce discretization errors
- Minimize index size

Basic Approaches

- To build a dictionary and solve the quantization problem:
  - **Hierarchical k-means (HKM)**
  - **Approximate k-means (AKM)**
  - **Hamming embedding**
  - Soft assignment
  - Fine vocabulary

Hierarchical k-means (HKM)

- «Words tree»
- Hierarchical subdivision
  - Quantize all data by K clusters ($k = 10$)
  - Then quantize each cluster by k clusters
- For example:
  - Tree depth 6 results in 1M leaves
- To reduce the effect of quantization, the descriptor is "softly" assigned to all parents along the corresponding tree branch



Nister, David, and Henrik Stewenius. "Scalable recognition with a vocabulary tree." *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. Vol. 2. Ieee, 2006.

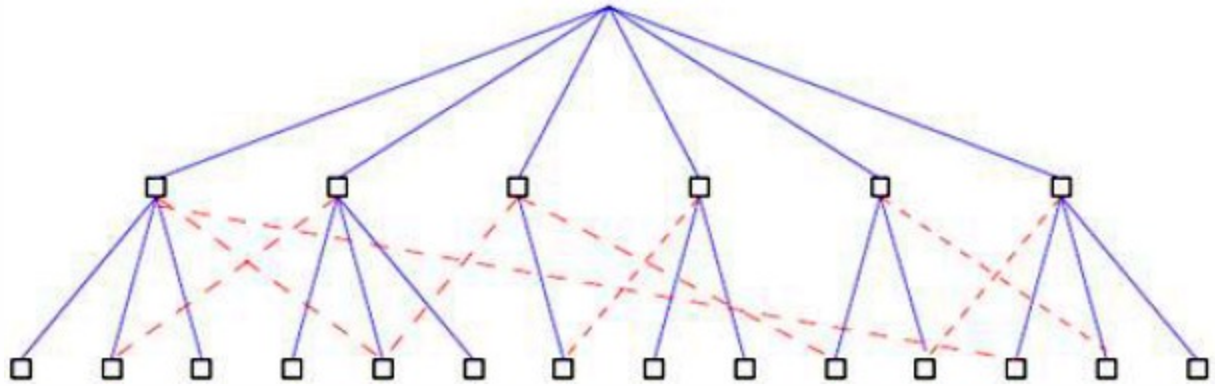
Approximate k-means (AKM)

- Algorithm
  - Forest of 8 randomized *k-d* trees
  - The partition parameter (coordinate) is chosen randomly from the set – the one with the largest spread
  - The partition threshold is chosen randomly close to the median
- Such a partition allows one to reduce the side effects of quantization
- The complexity of each k-means phase is reduced from *O(NK)* to *O(Nlog(K))*

Philbin, James, et al. "Object retrieval with large vocabularies and fast spatial matching." *2007 IEEE conference on computer vision and pattern recognition*. IEEE, 2007.

Quantization

- Direct comparison of a descriptor with the entire dictionary is very slow
- Build a hierarchical structure:
  - Build the first level of k-words with k-means algorithm
  - Repeat the k-means algorithm on clusters
  - Train additional connections between levels with the learning set



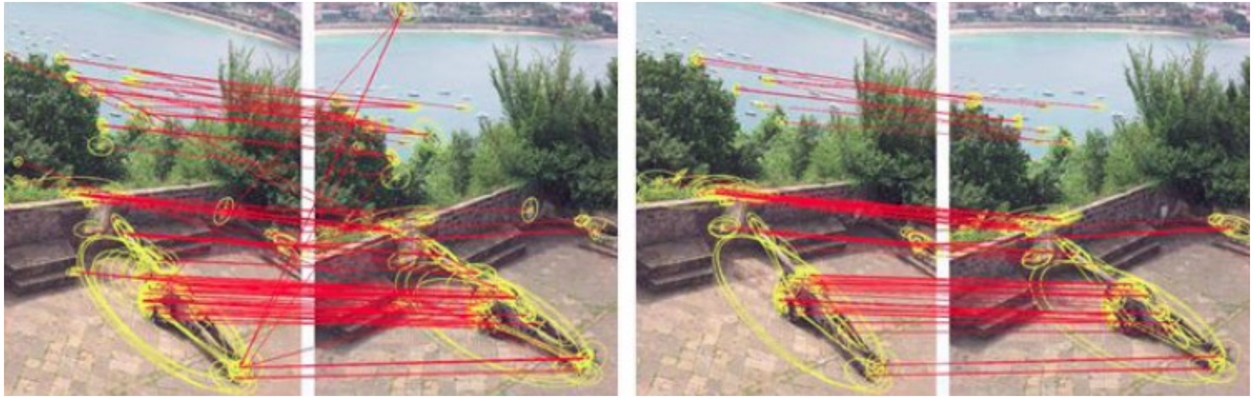
Bag Of Words Summary

- **Algorithm**
  - “Bag Of Words” descriptor of high dimension (1M)
  - Approximate k-means to build a dictionary for a large collection (5k)
  - Inverted index for storage
- **Testing**
  - 5k+100k images, 1M words, 1GB index, 0.1sec. search time
  - 5k+100k+1M images, 1M слов, 4GB+ index, stored on HDD, 10-35 sec. search time

Philbin, James, et al. "Object retrieval with large vocabularies and fast spatial matching." *2007 IEEE conference on computer vision and pattern recognition*. IEEE, 2007.

Hamming Embedding (HE)

- Dictionary size selection problem:



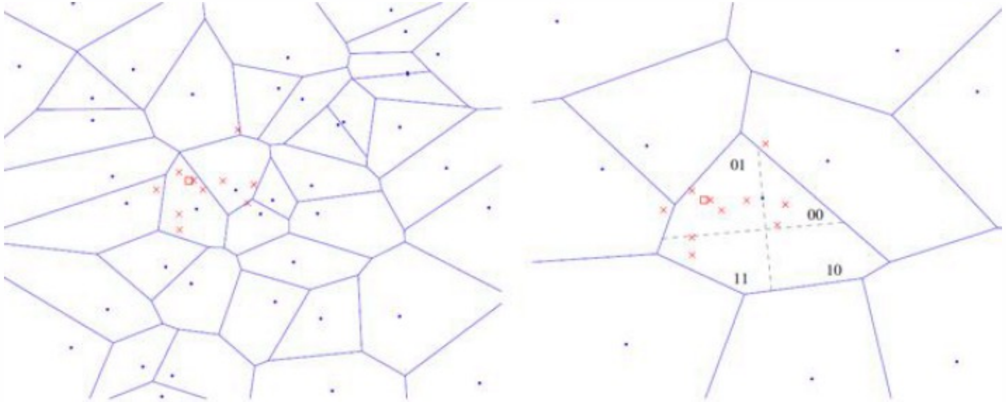
20 000 words 200 000 words

- Clustering does not approximate the descriptor comparison function accurately enough
- Small dictionary – lot of false matches, large dictionary – lot of misses

Jegou, Herve, Matthijs Douze, and Cordelia Schmid. "Hamming embedding and weak geometric consistency for large scale image search." *European conference on computer vision*. Springer, Berlin, Heidelberg, 2008.

Hamming Embedding (HE)

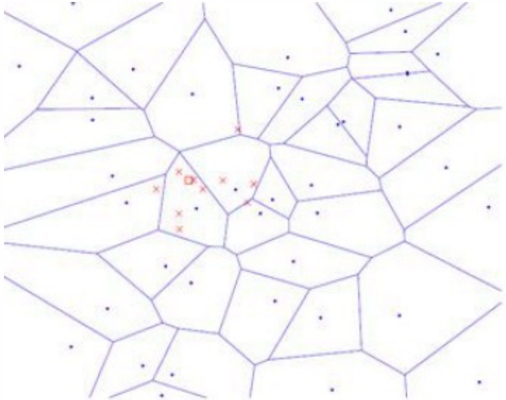
- Dictionary size selection problem:

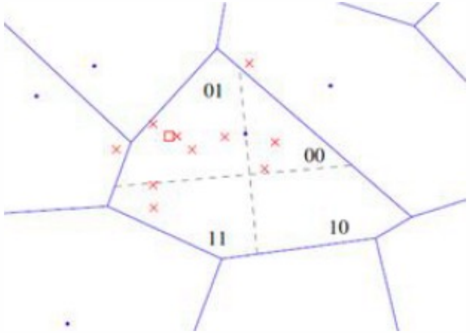


- Small dictionary – big clusters
  - Too rough comparison threshold
- Big dictionary – small clusters
  - Too precise comparison threshold

Hamming Embedding (HE)

- We want to store not only the index of the word for the feature from the image, but also describe its position inside the cluster (additional code)
- Then compare features not only by index, but also by additional code
- The code should be small and the comparison is fast:
    - We need a binary code
    - We will compare codes by the Hamming distance
        - the number of positions at which the word symbols are different

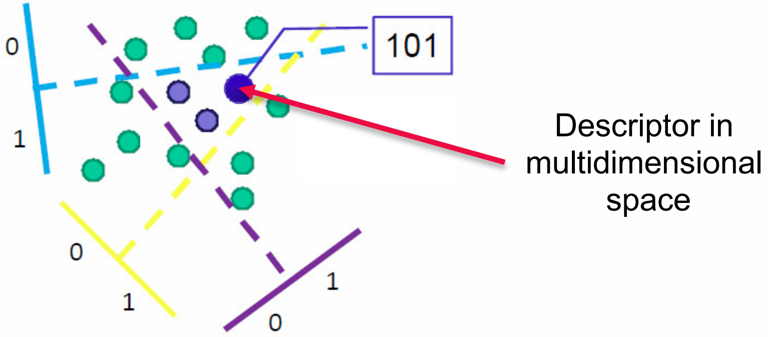




Locality Sensitive Hashing (LSH)

ITMO

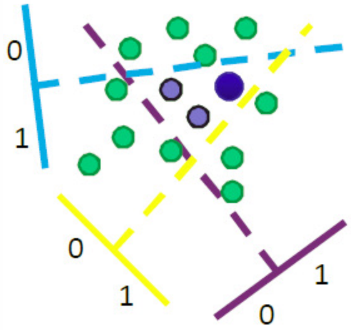
- Let's take a random projection of data onto a straight line
- Randomly choose a threshold and mark the projections by 0 or 1
  - 1 signature bit
- With an increase in the number of bits, the signature approximates L_2,
  the metric in the original descriptors
- Disadvantages:
  - L_2 Approximation is asymptotic
  - The implementation may require too many bits for the signature

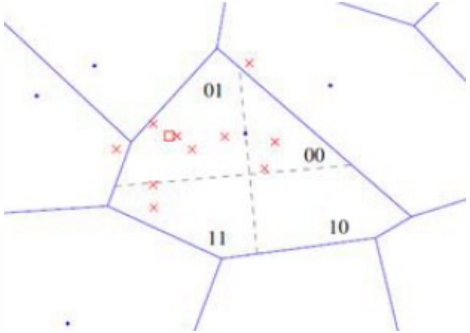


Andoni, Alexandr, and Piotr Indyk. "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions." *2006 47th annual IEEE symposium on foundations of computer science (FOCS'06)*. IEEE, 2006.

Hamming Embedding (HE)

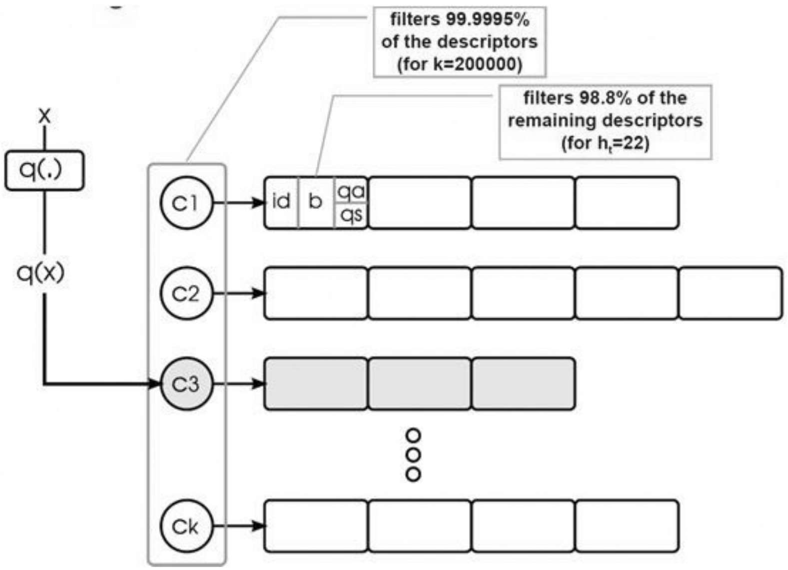
- Take all the descriptors that are quantized into one cluster
- Generate n random straight lines (projection directions)
- Project all descriptors onto these lines
- Choose a point on each line (threshold) in such a way that there is equal number of points on both sides of it
- Build a binary code basing on feature position related to a selected lines and thresholds
  - Such a code would be optimal





Collection Index Modification

- Each feature has its own entry in the index (before that, they were combined into one with their number)

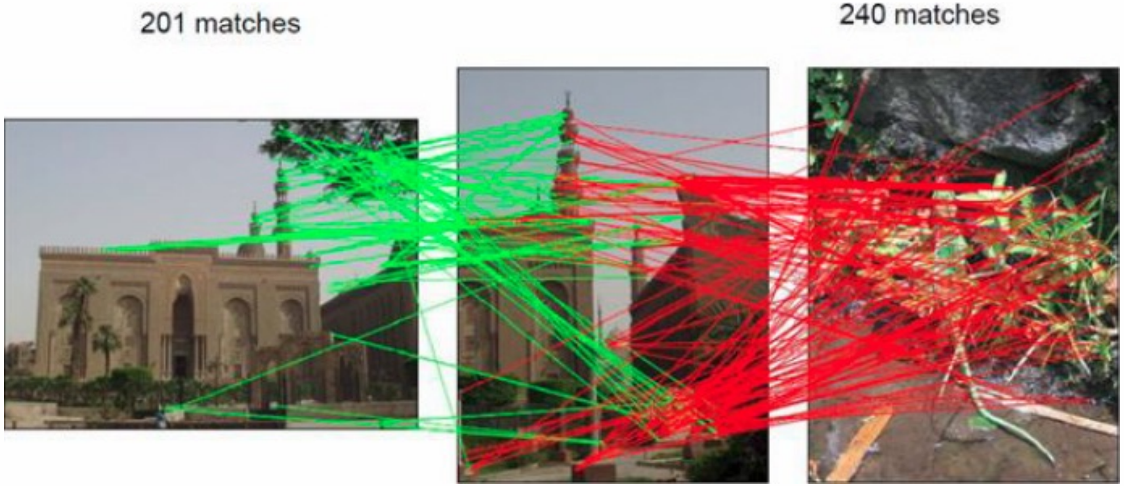


General Algorithm

- For each descriptor:
  - Quantize it by dictionary (word index)
  - Compute binary code
- We consider points to be matched only if both conditions are met:
  - Word indices are the same
  - Binary codes differ by no more than z computed with Hamming distance

Hamming Embedding (HE)

- Example
- 20k words

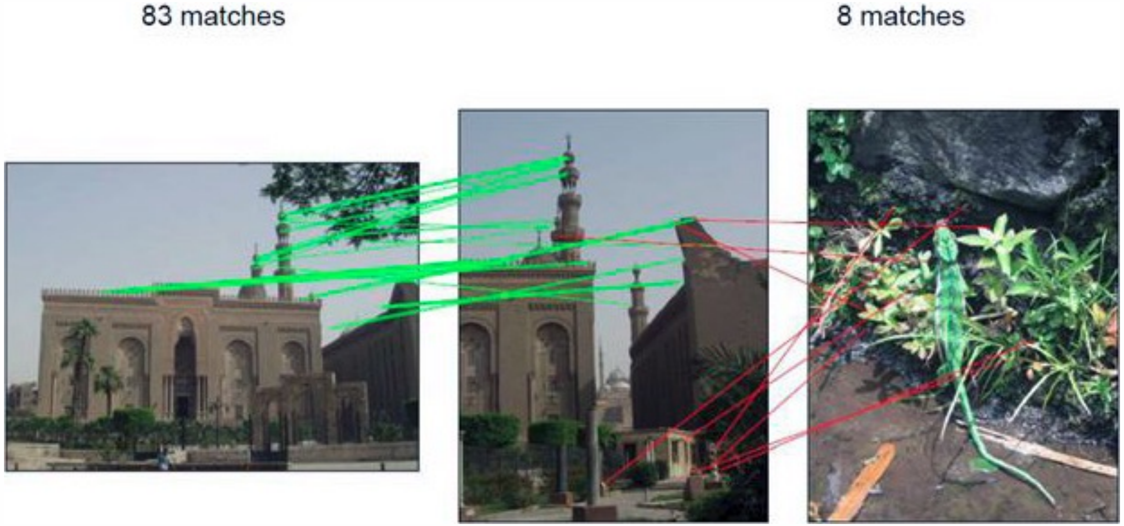


Hamming Embedding (HE)

- Example
- 200k words

Hamming Embedding (HE)

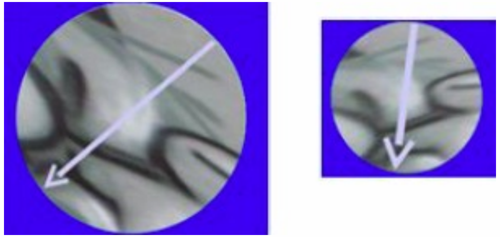
- Example
- 200k words + HE



Weak Geometric Consistency (WGC)

- Each feature point is also described by its scale (characteristic size) and orientation
- **Example:**

difference in
orientation is 20°
scaled by 1.5 times

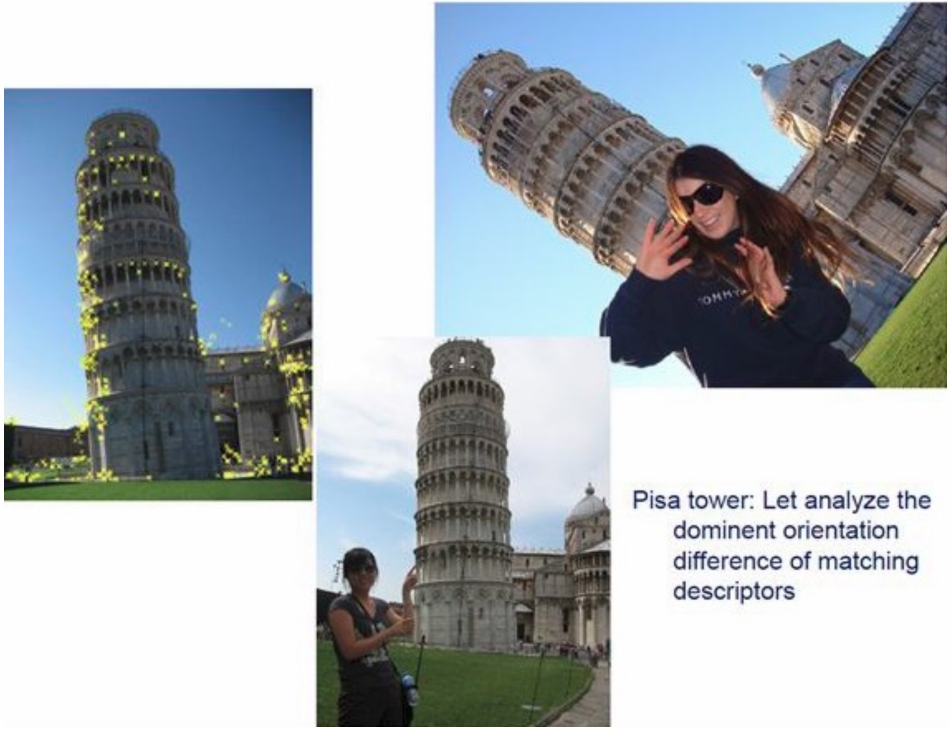


- Each pair of matched points sets the difference in angle and scale
- For the entire image, these angles and scales should be consistent
- Each pair of matching points will vote for a certain combination of difference in orientation and scale.

Jegou, Herve, Matthijs Douze, and Cordelia Schmid. "Hamming embedding and weak geometric consistency for large scale image search." *European conference on computer vision*. Springer, Berlin, Heidelberg, 2008.

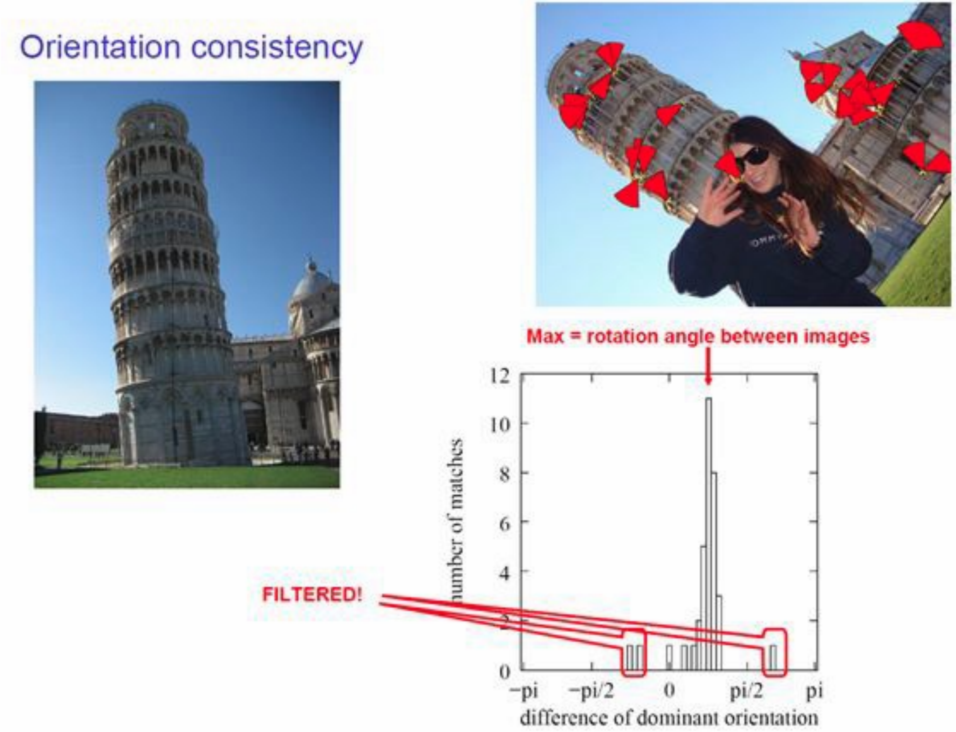
Weak Geometric Consistency (WGC)

- Orientation consistency
- Example



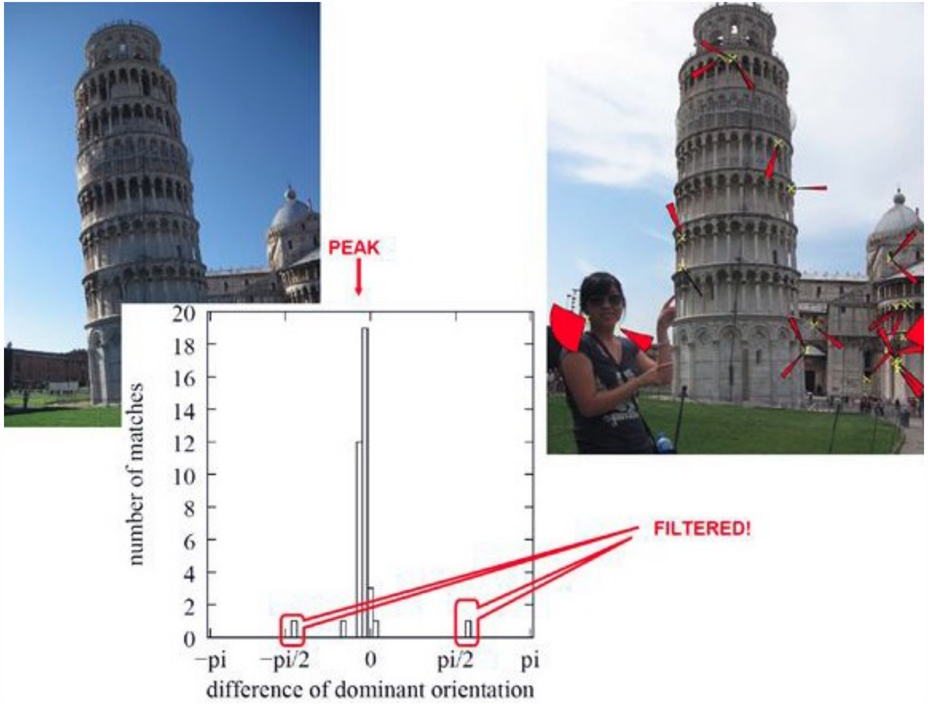
Weak Geometric Consistency (WGC)

- Orientation consistency
- Example



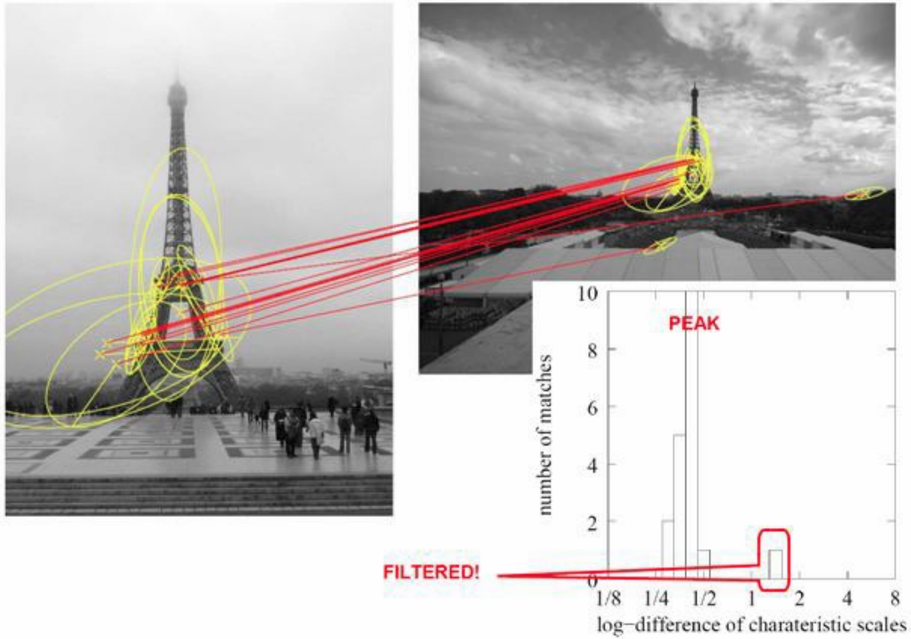
Weak Geometric Consistency (WGC)

- Orientation consistency
- Example



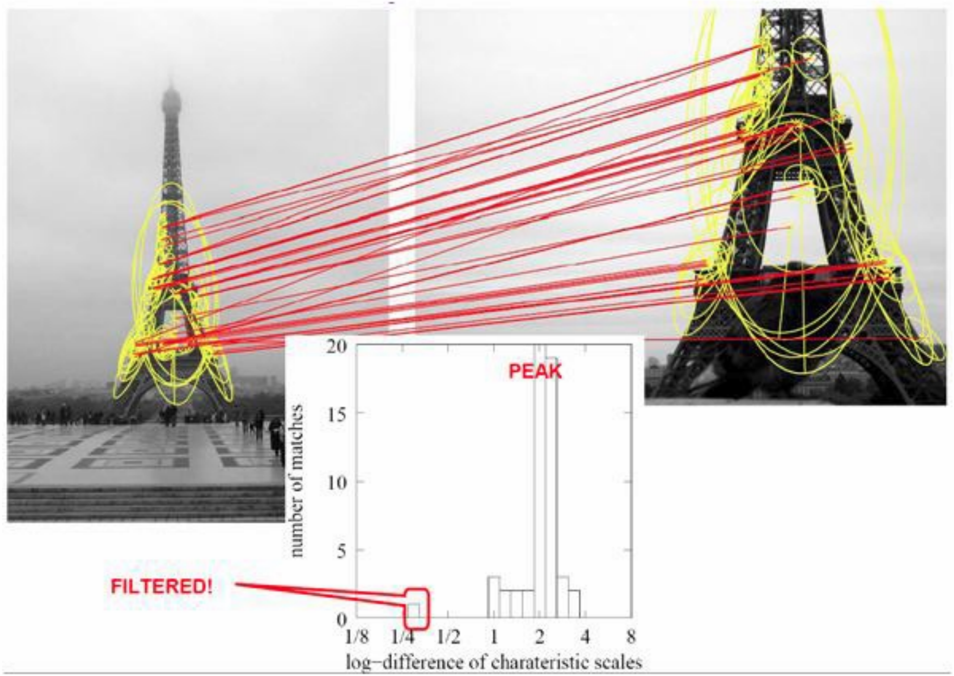
Weak Geometric Consistency (WGC)

- Orientation consistency
- Example



Weak Geometric Consistency (WGC)

- Scale consistency
- Example

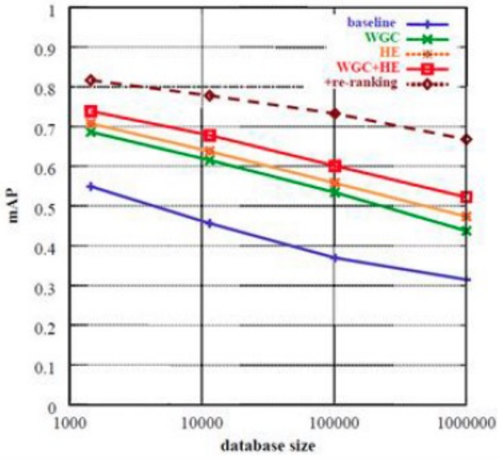


Weakly Geometry Consistency Method

- Scale and orientation are almost independent from each other
- Voting with discrete scale and rotation:
  - Separate weight for each combination (rotation angle/scale):
    - In fact, a histogram
  - We take the maxima in terms of rotation angle / scale
- Only matches that match in scale and orientation terms contribute to the final matching score

Results

- Each method: weakly geometry, hamming embedding, geometry re-ranking – significantly improves accuracy
- At the same time, the combined use of WGC and HE does not make it possible to achieve a speed comparable to the baseline



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

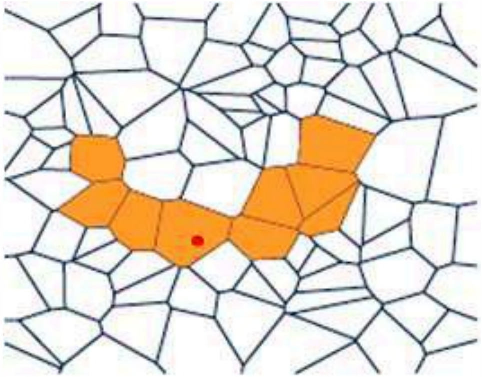
Alternative Words

- The SIFT descriptor is not always sufficiently invariant
- Strong perspective distortions lead to a significant increase in distance between descriptors of the same feature point
- The same feature point can be placed into different clusters

Descriptors of the same feature point on a series of images

Alternative Words

- At the training stage, store the “alternative words”, because such cells may also contain the "correct" matches
- At the search stage, vote not only for the same word, but also for alternative ones
- This will increase the index size by:
    - number of alternative words $*$ dictionary size



Processing the Search Results

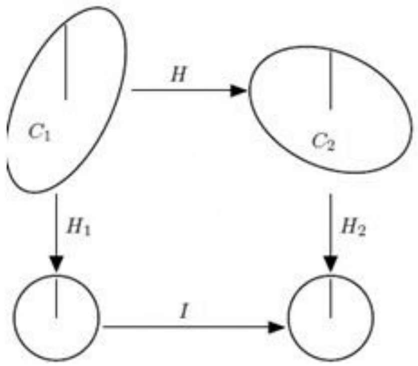
- Re-rank the search results list
- Search query expansion
  - If we solve the problem of finding objects, then the found images can be well matched with the request
  - The standard scheme (local features + robust transformation calculation) is too slow for exhaustive search
  - Instead, we can use it for post-processing - ranking the found images

Re-Rank Results

- Compare the features between the “query” and the images filtered by the search
- Filter outliers with LO-RANSAC (Locally-Optimized Random Sequence Consensus):
  - At first, build a simple model
  - Then, build a more complex model basing on inliers
- Refine a good model based on the found inliers:
  - Affine model
- Reorder images:
  - For those who matched:
    - To the beginning of the list
    - Reorder by the number of inliers (the more inliers – the higher is the position in the list)
  - For those who didn't match:
    - To the end of the list
    - Without reordering

Model Estimation By 1 Pair of Points

- One pair of matched points is enough to generate a hypothesis for RANSAC
- Up to 5 parameters can be estimated:
  - Shift (2)
  - Scale (1)
  - Rotation (1)
  - Proportions (ellipsoid)



Re-Ranking Results

- Percentage of desired images at the top of the re-ranked list after geometric matching:

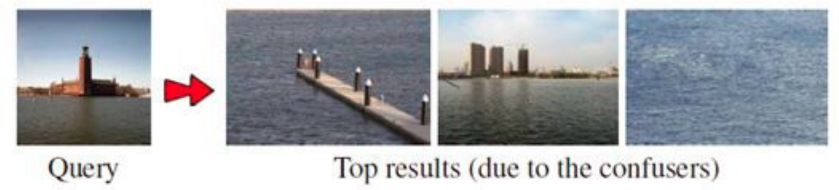
- When searching images of architecture, the re-ranking shows a significant increase in an accuracy

Search Query Expansion

- Transitive closure expansion (TCE)
  - Building a query tree
  - The root node is the original query
  - Children are the best-matched images from the query results
- Additive query expansion (AQE)
  - Display feature points from the found images to the original image
  - Use the modified image to search and expand the results
- Average query expansion
  - We average the descriptors of all found images and use them for the search

Confusing Features

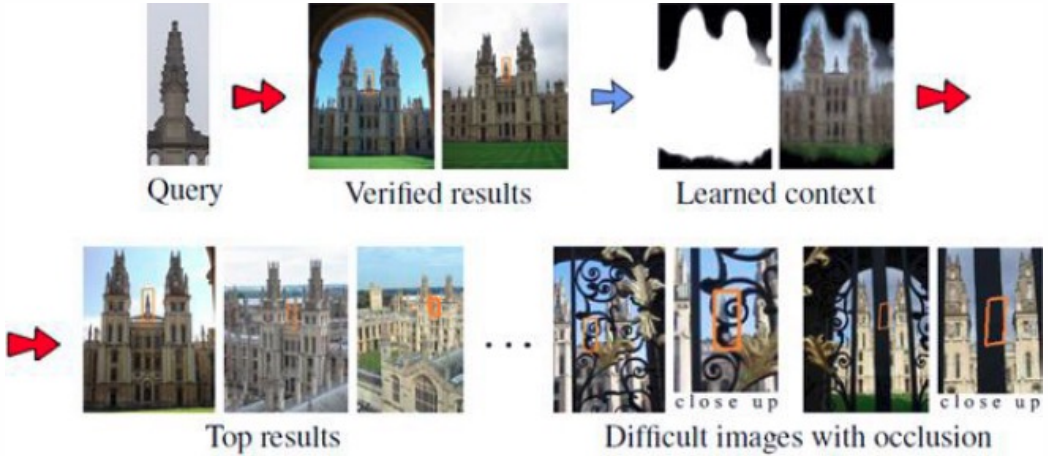
- In repeating semi-random textures (e.g., water) there are many feature points that are not related to the object, and they reduce the search quality



- **Idea:** in such cases, the images are very poorly comparable geometrically
- Have to detect such situations, teach the model of ”confusing features” and remove them from the search request

Chum, Ondřej, et al. "Total recall II: Query expansion revisited." *CVPR 2011*. IEEE, 2011.

Model Improvement



- **Model** – the set of features from the search query
- **Idea:** If we found a well-matched image, then it is worth adding features this image to the model
- The updated model will allow finding more similar images

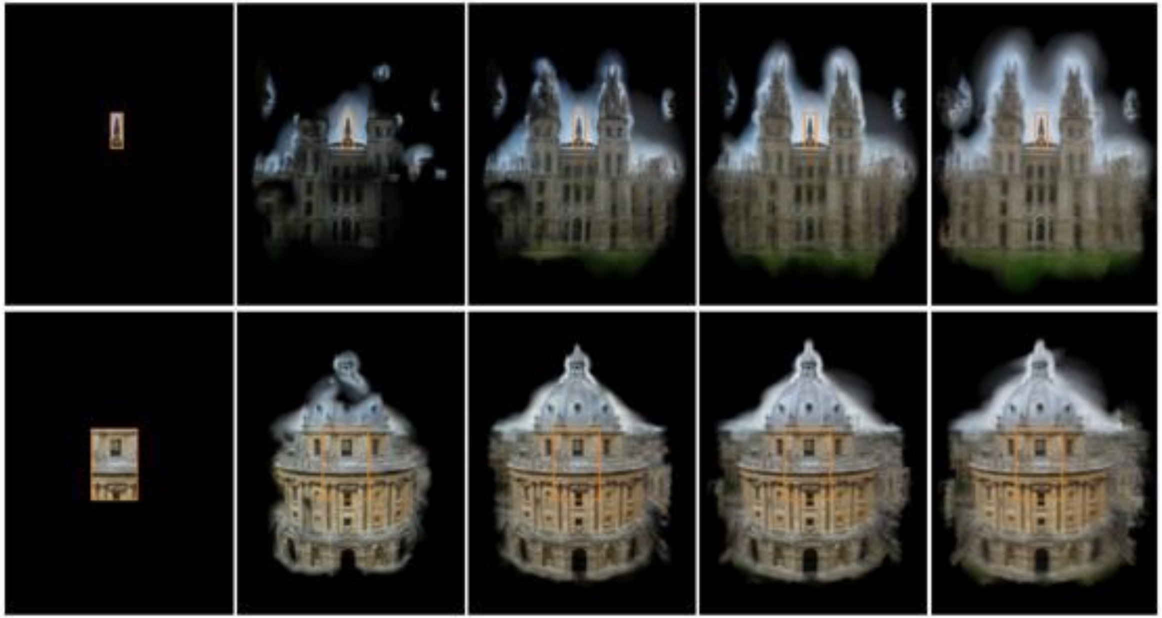
Incremental Spatial Re-Ranking (iSP)

- **Idea**:
  - If we have found a well-matched image, then it is worth adding features from it to the model
- **Method**:
  - We have the model M – features from the search query X
  - Iterate through the search results list S
  - If model M and search result $S[i]$ matched more than $T = 15$ features, then add features from result $S[i]$ to model M.

Incremental Spatial Re-Ranking (iSP)

ITMO

- iSP allows adding new features to the model, even when are outside the image-query



THANK YOU FOR YOUR TIME!

ITsMOre than a UNIVERSITY

s.shavetov@itmo.ru