Compressed local descriptors for fast image and video search in large databases

Matthijs Douze[2]

joint work with Hervé Jégou[1], Cordelia Schmid[2] and Patrick Pérez[3]

1: INRIA Rennes, TEXMEX team, France

2: INRIA Grenoble, LEAR team, France

3: Technicolor, France

INRIA

Problem setup: Image indexing

- Retrieval of images representing the same object/scene:
  - different viewpoints, backgrounds, …
  - copyright attacks: cropping, editing, …
  - short response time
  - **100s of millions** of images or 1000s of hours of video

queries

relevant answers

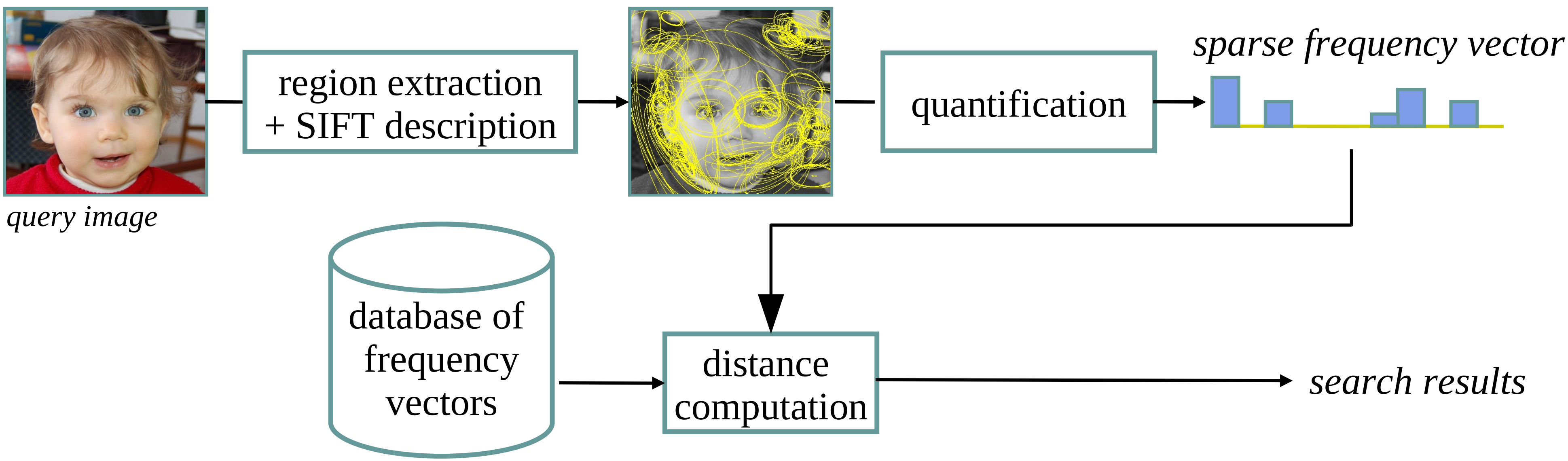
INRIA

Related work on large scale image search

- Global descriptors:
  - color/texture statistics
  - GIST descriptors with Spectral Hashing or similar techniques [Torralba & al 08]

→ very limited invariance to scale/rotation/crop

- Local descriptors → compact them: Bag of Features [Sivic & Zissermann 03]



  - Improvements: hierarchical vocabulary, compressed BoF, partial geometry...

    → But still hundreds of bytes are required to obtain a “reasonable quality”

Outline

Image description with VLAD

Indexing with the product quantizer

Porting to mobile devices

Video indexing

Objective and proposed approach [Jégou & al., CVPR 10]

- Aim: optimizing the trade-off between
  - search speed +
  - memory usage +
  - search quality -

$$\xrightarrow[\textit{SIFT}]{\textit{extract}} \underbrace{\begin{bmatrix}5\\0\\\vdots\\35\end{bmatrix} \begin{bmatrix}34\\6\\\vdots\\12\end{bmatrix} \cdots \begin{bmatrix}14\\20\\\vdots\\47\end{bmatrix}}_{n\ \textit{SIFTs (128 dim)}} \xrightarrow[\textit{descriptors}]{\textit{aggregate}} \overset{D}{\begin{bmatrix}-0.234\\0.452\\0.134\\\vdots\\0.001\end{bmatrix}} \xrightarrow[\textit{reduction}]{\textit{dimension}} \overset{D'}{\begin{bmatrix}0.155\\0.230\\\vdots\\0.435\end{bmatrix}} \xrightarrow[\textit{encoding /indexing}]{\textit{vector}} \text{code}$$

- Approach: joint optimization of three stages
  - local descriptor aggregation
  - dimension reduction
  - indexing algorithm

Aggregation of local descriptors

- Problem: represent an image by a single fixed-size vector:

set of *n* local descriptors → 1 vector

- Indexing:
  - similarity = distance between aggregated description vectors (preferably L2)
  - search = (approximate) nearest-neighbor search in descriptor space

- Most popular idea: BoF representation [Sivic & Zisserman 03]
  - sparse vector
  - highly dimensional

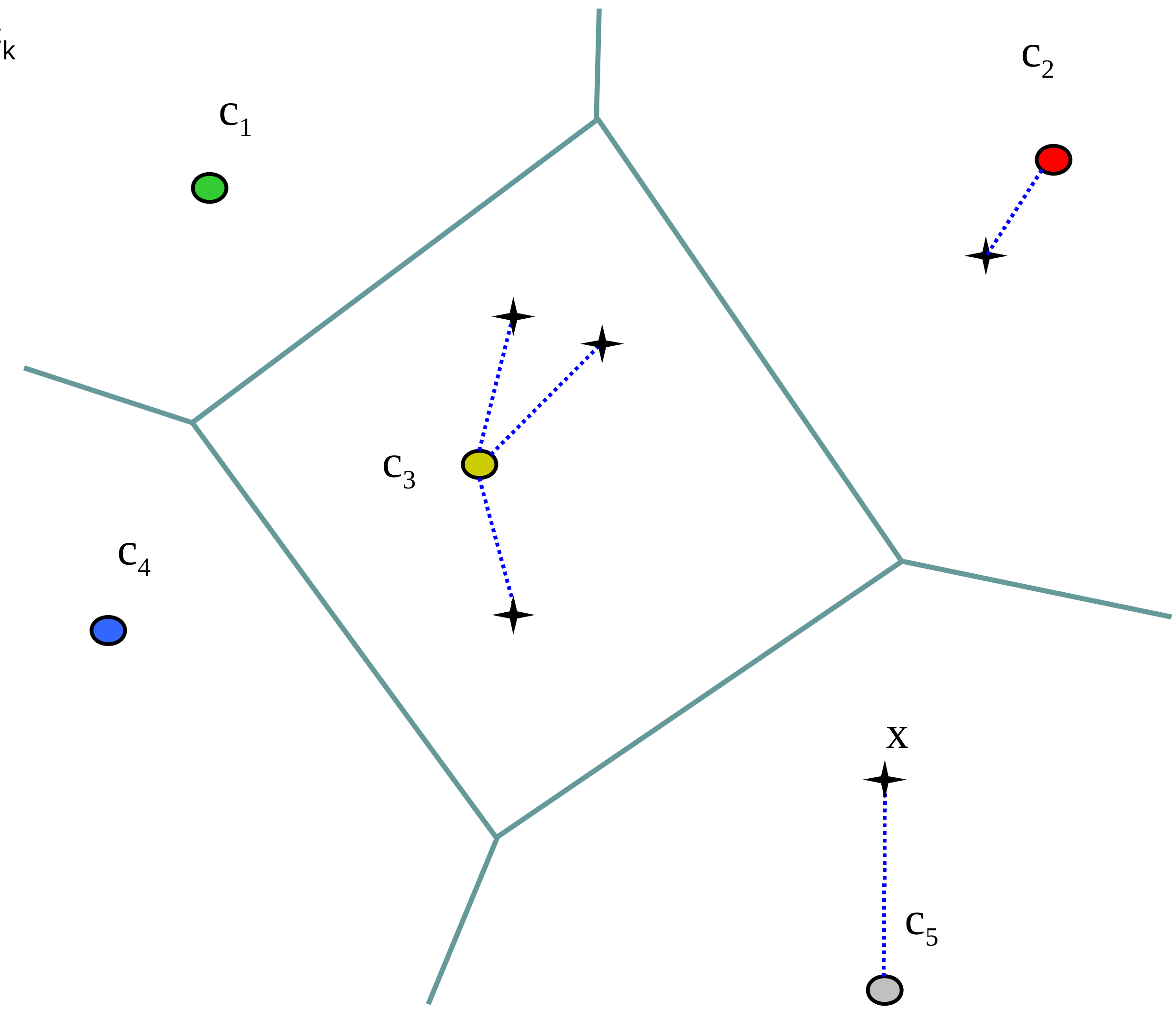
→ dimensionality reduction harms precision a lot

- Alternative: Fisher Kernels [Perronnin et al 07]
  - non sparse vector
  - excellent results with a small vector dimensionality

  → VLAD is in the spirit of this representation

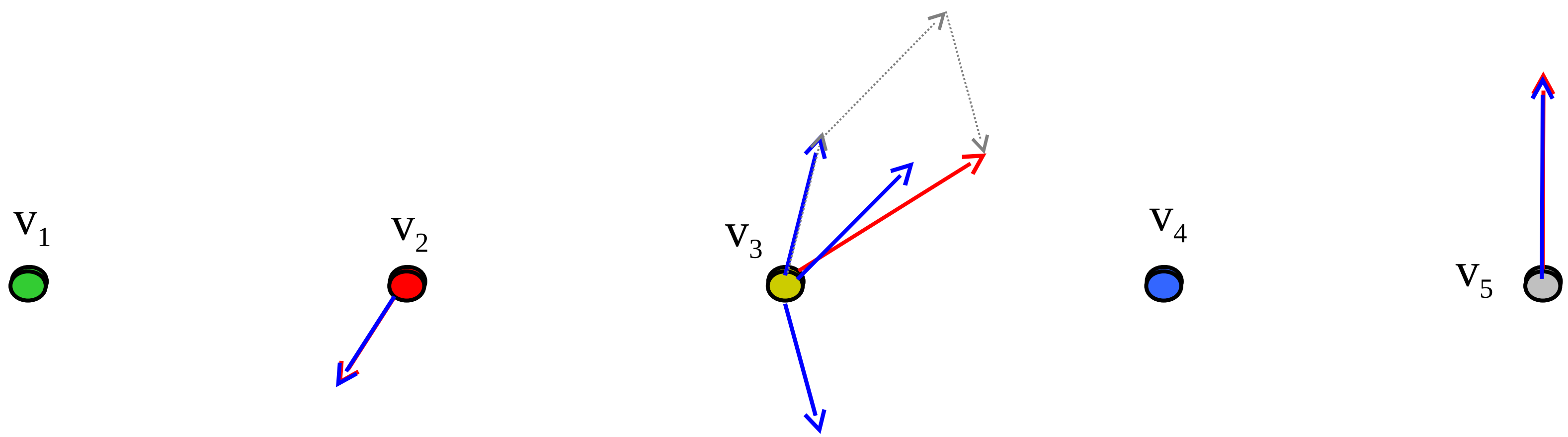
VLAD : Vector of Locally Aggregated Descriptors

- D-dimensional descriptor space (SIFT: D=128)
- *k* centroids : $c_1,\ldots,c_k$



VLAD : Vector of Locally Aggregated Descriptors

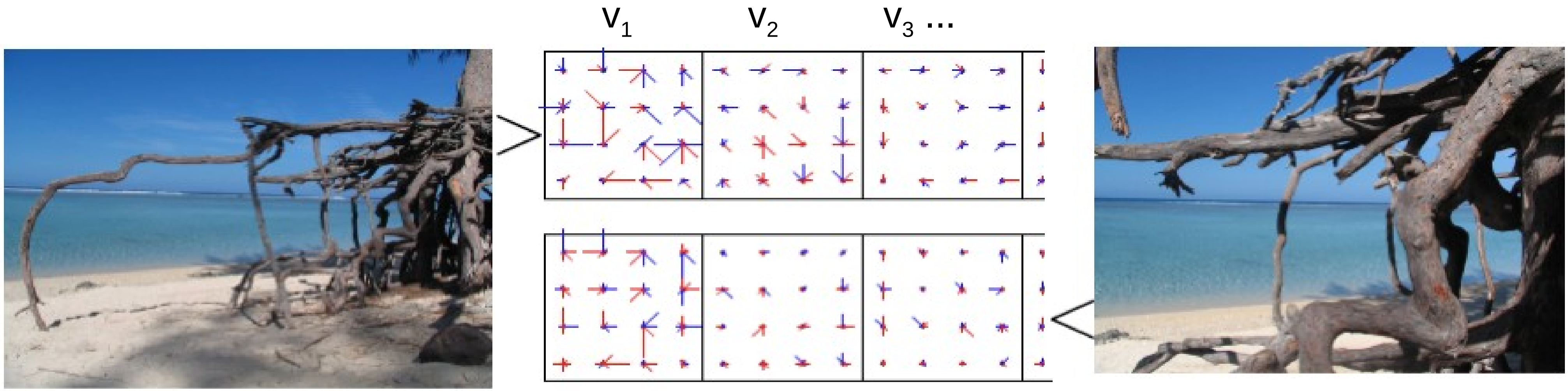
- D-dimensional descriptor space (SIFT: D=128)
- *k* centroids : $c_1,\ldots,c_k$



- Output: $v_1 \ldots v_k$ = descriptor of size k*D
- L2-normalized
- Typical *k* = 16 or 64 : descriptor in 2048 or 8192 D
- Similarity measure = L2 distance.

INRIA

VLADs for corresponding images



SIFT-like representation per centroid (>0 components: blue, <0 components: red)

- good coincidence of energy & orientations

VLAD performance and dimensionality reduction

- We compare VLAD descriptors with BoF: INRIA Holidays Dataset (mAP,%)
- Dimension is reduced to from D to D' dimensions with PCA

Aggregator	k	D	D'=D (no reduction)	D'=128	D'=64
BoF	1,000	1,000	41.4	44.4	43.4
BoF	20,000	20,000	44.6	45.2	44.5
BoF	200,000	200,000	54.9	43.2	41.6
VLAD	16	2,048	49.6	49.5	**49.4**
VLAD	64	8,192	52.6	**51.0**	47.7
VLAD	256	32,768	**57.5**	50.8	47.6

- Observations:
  - performance increases with k
  - VLAD better than BoF for a given descriptor size
  - if small D' needed: choose a smaller k

INRIA

Outline

Image description with VLAD

Indexing with the product quantizer

Porting to mobile devices

Video indexing

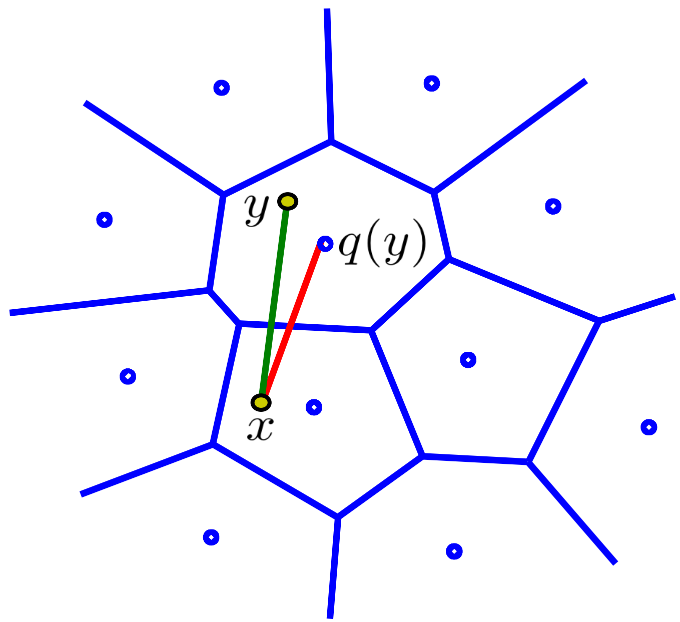
Indexing algorithm: searching with quantization [Jégou & al., PAMI to appear]

- Search/Indexing = distance approximation problem
- The distance between a query vector *x* and a database vector y is estimated by

$$d\,(x,y) \approx d\,(x,q(y))$$

where q(.) is a quantizer

→ vector-to-code distance



- The choice of the quantizer is critical
  - fine quantizer → need many centroids: typically 64-bit codes → $k=2^{64}$
  - regular (and approximate) k-means can not be used

Product quantization for nearest neighbor search

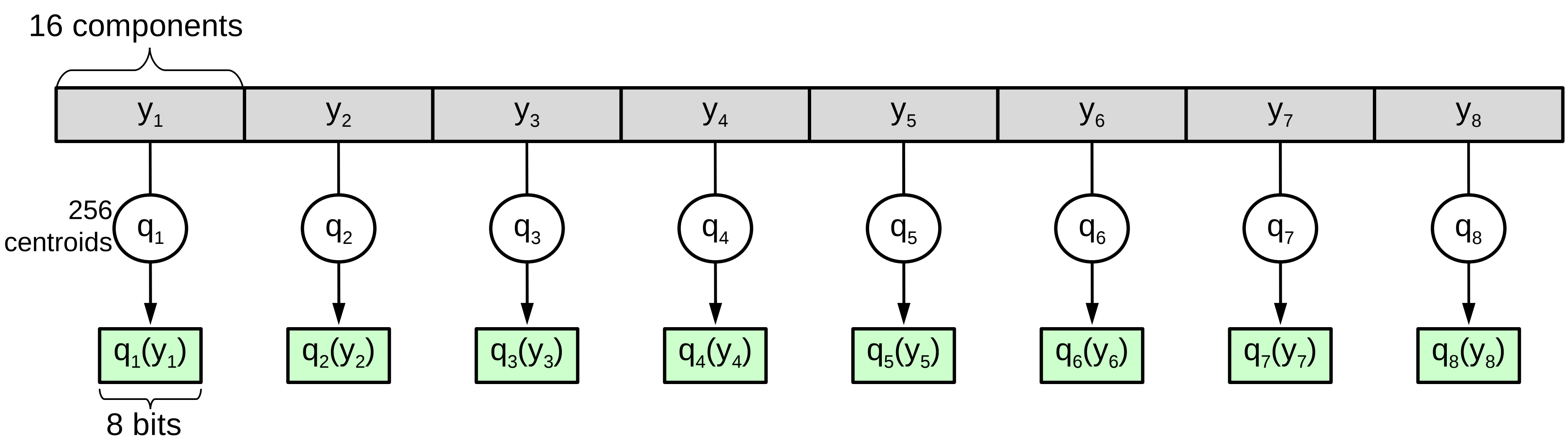
- Vector split into *m* subvectors: $y \rightarrow [y_1 | \ldots | y_m]$

- Subvectors are quantized separately

$$q(y) = [q_1(y_1) | \ldots | q_m(y_m)]$$

where each q_i is learned by *k*-means with a limited number of centroids

- Example: y = 128-dim vector split in 8 subvectors of dimension 16



⇒ 64-bit quantization index

Product quantization for nearest neighbor search

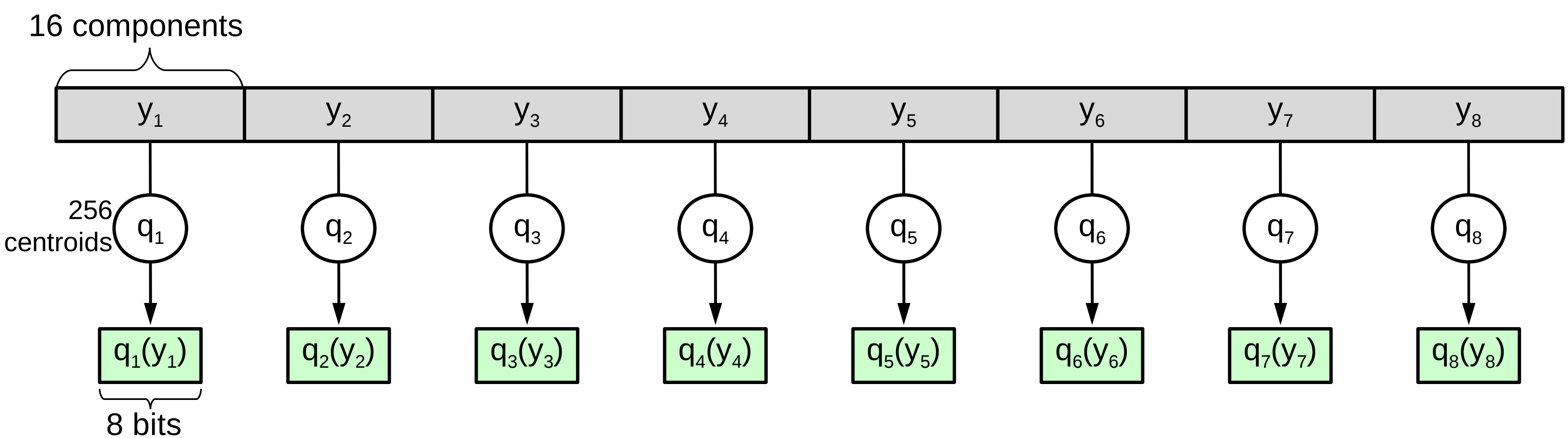
- Vector split into *m* subvectors: $y \rightarrow [y_1 | \ldots | y_m]$

- Subvectors are quantized separately

$$q(y) = [q_1(y_1) | \ldots | q_m(y_m)]$$

where each q_i is learned by *k*-means with a limited number of centroids

- Example: y = 128-dim vector split in 8 subvectors of dimension 16



⇒ 64-bit quantization index

INRIA

Product quantizer: asymmetric distance computation (ADC)

- Compute the distance approximation in the compressed domain

$$d(x,y)^2 \approx \sum_{i=1}^{m} d(x_i, q_i(y_i))^2$$

- To compute distance between query x and many codes
  - compute $d(x_i, c_{i,j})^2$ for each subvector x_i and all possible centroids

  → stored in look-up tables
  - for each database code: sum up the elementary squared distances

- Each 8x8=64-bits code requires only **m = 8 additions per distance**!

Results on standard datasets

- Datasets
  - University of Kentucky benchmark — score: nb relevant images, max: 4
  - INRIA Holidays dataset — score: mAP (%)

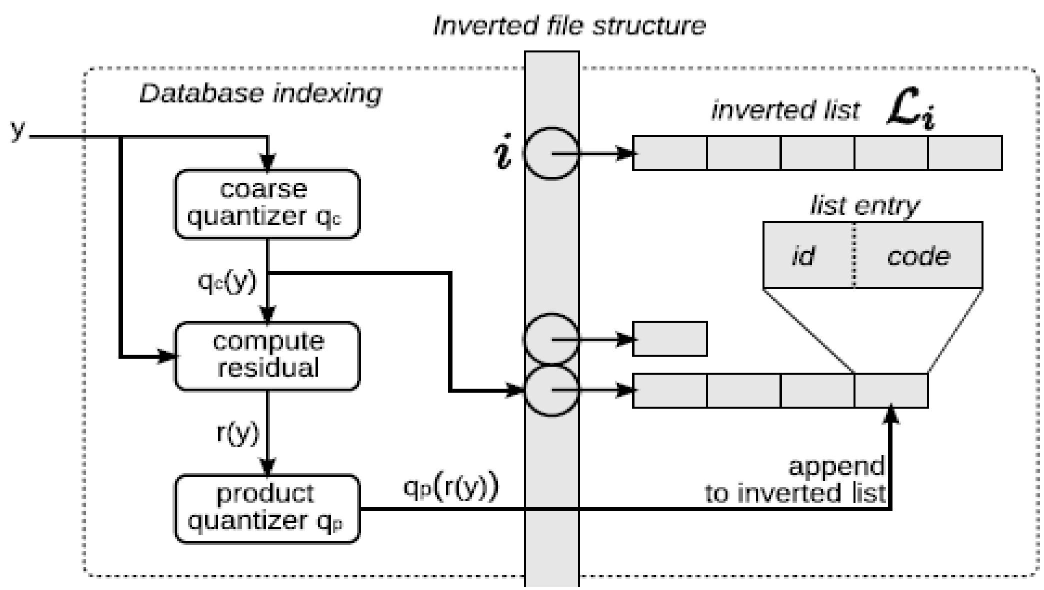
Method	bytes	UKB	Holidays
BoF, k=20,000	10K	2.92	44.6
BoF, k=200,000	12K	3.06	54.9
miniBOF	20	2.07	25.5
miniBOF	160	2.72	40.3
VLAD k=16, ADC	**16**	**2.88**	**46.0**
VLAD k=64, ADC	**64**	**3.10**	**49.5**

miniBOF: “Packing Bag-of-Features”, ICCV’09

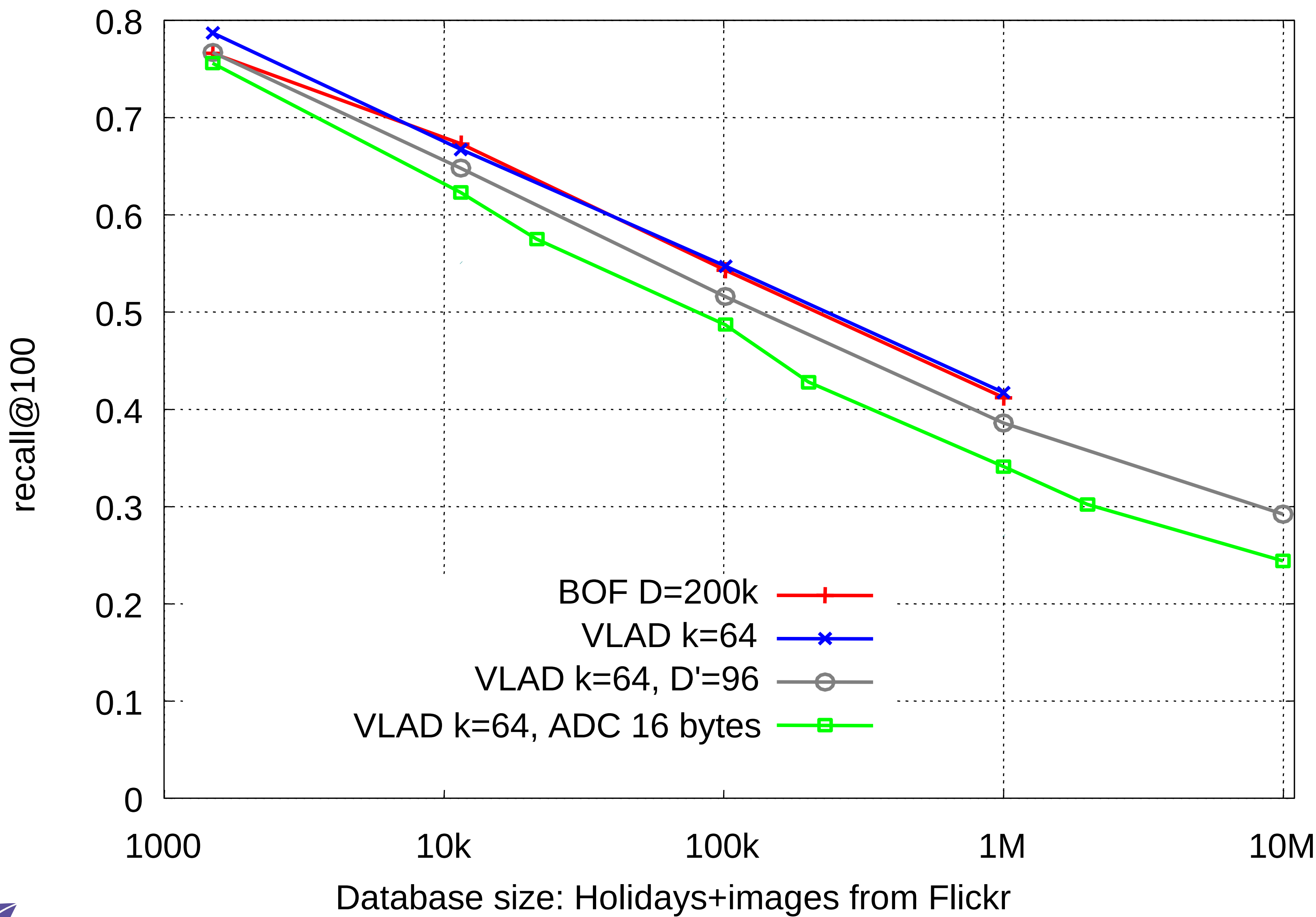
IVFADC: non-exhaustive ADC

- IVFADC
  - Additional quantization level
  - Combination with an inverted file
  - visits 1/128th of the dataset

- Timings for 10 M images
  - Exhaustive search with ADC: 0.286 s
  - Non-exhaustive search with IVFADC: 0.014 s



Large scale experiments (10 million images)



INRIA

Outline

Image description with VLAD

Indexing with the product quantizer

Porting to mobile devices

Video indexing

On the mobile

- Indexing on the server:

$$\xrightarrow[\textit{SIFT}]{\textit{extract}} \underbrace{\begin{bmatrix}5\\0\\\vdots\\35\end{bmatrix}\begin{bmatrix}34\\6\\\vdots\\12\end{bmatrix}\cdots\begin{bmatrix}14\\20\\\vdots\\47\end{bmatrix}}_{n \text{ SIFTs (128 dim)}} \xrightarrow[\textit{descriptors}]{\textit{aggregate}} \underbrace{\begin{bmatrix}-0.234\\0.452\\0.134\\\vdots\\0.001\end{bmatrix}}_{D} \xrightarrow[\textit{reduction}]{\textit{dimension}} \underbrace{\begin{bmatrix}0.155\\0.230\\\vdots\\0.435\end{bmatrix}}_{D'} \xrightarrow[\textit{server}]{\textit{send to}}$$

stage	image	SIFTs	VLAD	VLAD+PCA
data size	300 kB	512 kB	32 kB	384 bytes
computing time (relative)	NA	1.5 s (50 ms for CS-LBP)	5 ms	0.5 ms

- query from mobile
  - relatively cheap to compute
  - small bandwidth

Indexing on the mobile

- The database is stored on the device

- In addition to the previous:
  - database: 20 bytes per image in RAM
  - quantize query (find closest centroids + build look-up tables)
  - scan database to find nearest neighbors

- Adapt algorithms to optimize speed

db size (images)	exhaustive (ADC) / non-exhaust. (IVFADC)	precompute distance tables
<1000	ADC	no
<1M	ADC	yes
>1M	IVFADC	yes

INRIA

Outline

Image description with VLAD

Indexing with the product quantizer

Porting to mobile devices

Video indexing

Video indexing [Douze & al. ECCV 2010]

- video = image sequence
  - index VLAD descriptors for *all* images (CS-LBP instead of SIFT for speed)
  - temporal verification

- database side: images are grouped in segments
  - 1 VLAD descriptor represents each segment
  - frame represented as refinement w.r.t. this descriptor

- query = seach all frames of the query video

- Frame matches → alignment of query with database video
  - Hough transform on $\delta t = t_q - t_{db}$
  - Output: most likely δt → alignments
  - map back to frame matches to find aligned video segments

INRIA

Video indexing results

- Comparison with Trecvid 2008 copy detection task
  - 200 h indexed video
  - 2000 queries
  - 10 “attacks” = video editing, clutter, frame dropping, camcording...
  - state of the art: competition results (score = NDCR, lower = better)

transformation	**best**	**ours**	**rank (/23)**
camcording	0.08	0.22	**2**
picture in picture	0.02	0.32	4
insertion of patterns	0.02	0.08	3
strong re-encoding	0.02	0.06	2
geometric attacks	0.07	0.14	2
5 random transformations	0.20	0.54	**2**

- Observations:
  - Always among 5 first results
  - 5 times faster and 100 times less memory than competing methods
  - Best localization results (due to dense temporal sampling)

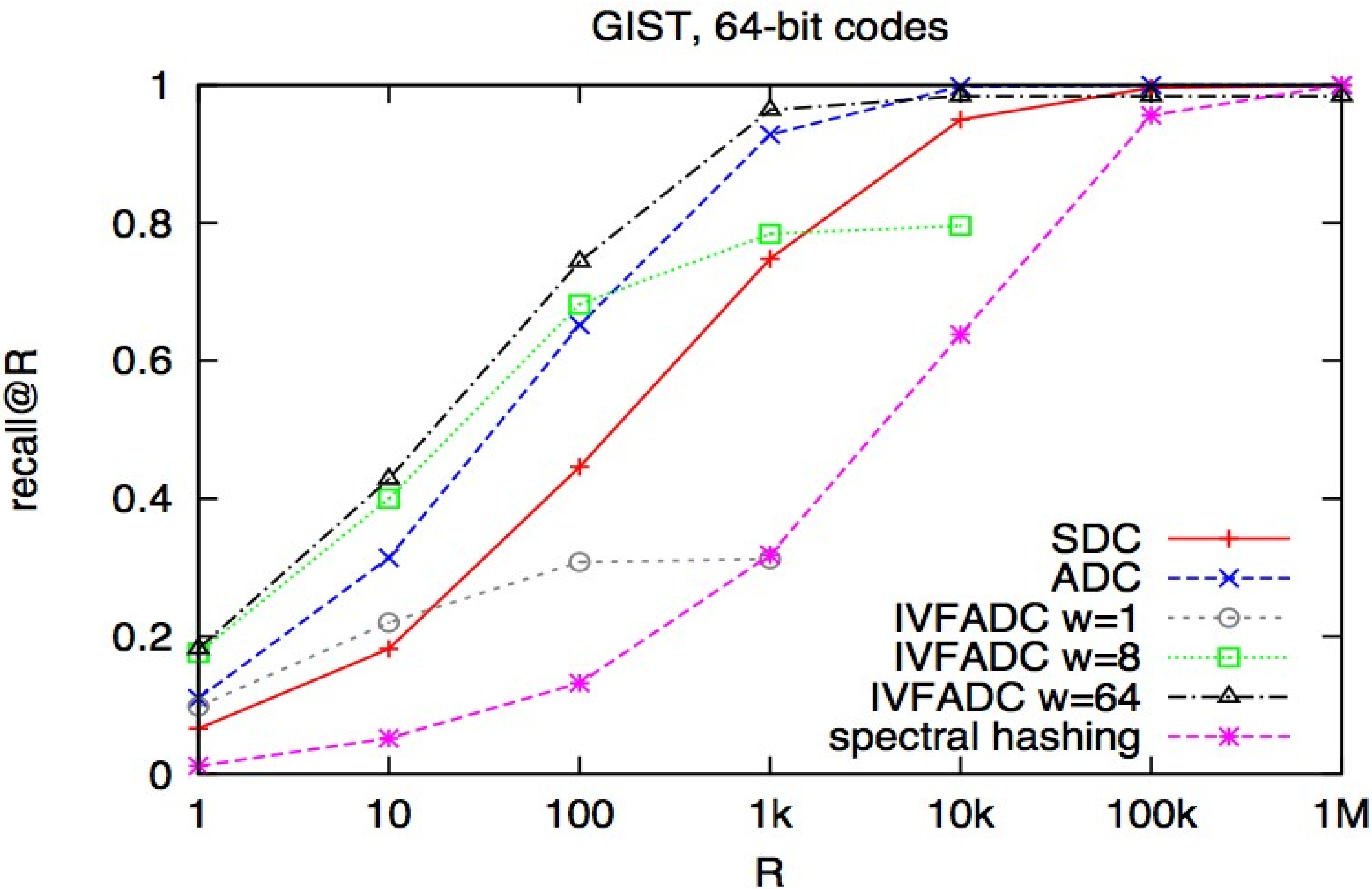
INRIA

Conclusion

- VLAD: compact & discriminative image descriptor
  - aggregation of SIFT, CS-LBP, SURF (ongoing),...
- Product Quantizer: generic indexing method with nearest-neighbor search function
  - works with local descriptors and GIST, audio features (ongoing)...
- Standard image and datasets
  - Holidays (different viewpoints)
  - Copydays (copyright attacks)
- Compatible with mobile applications:
  - compact descriptor, cheap to compute
- Code for VLAD and Product quantizer at http://www.irisa.fr/texmex/people/jegou/src.php
- Demo!

END

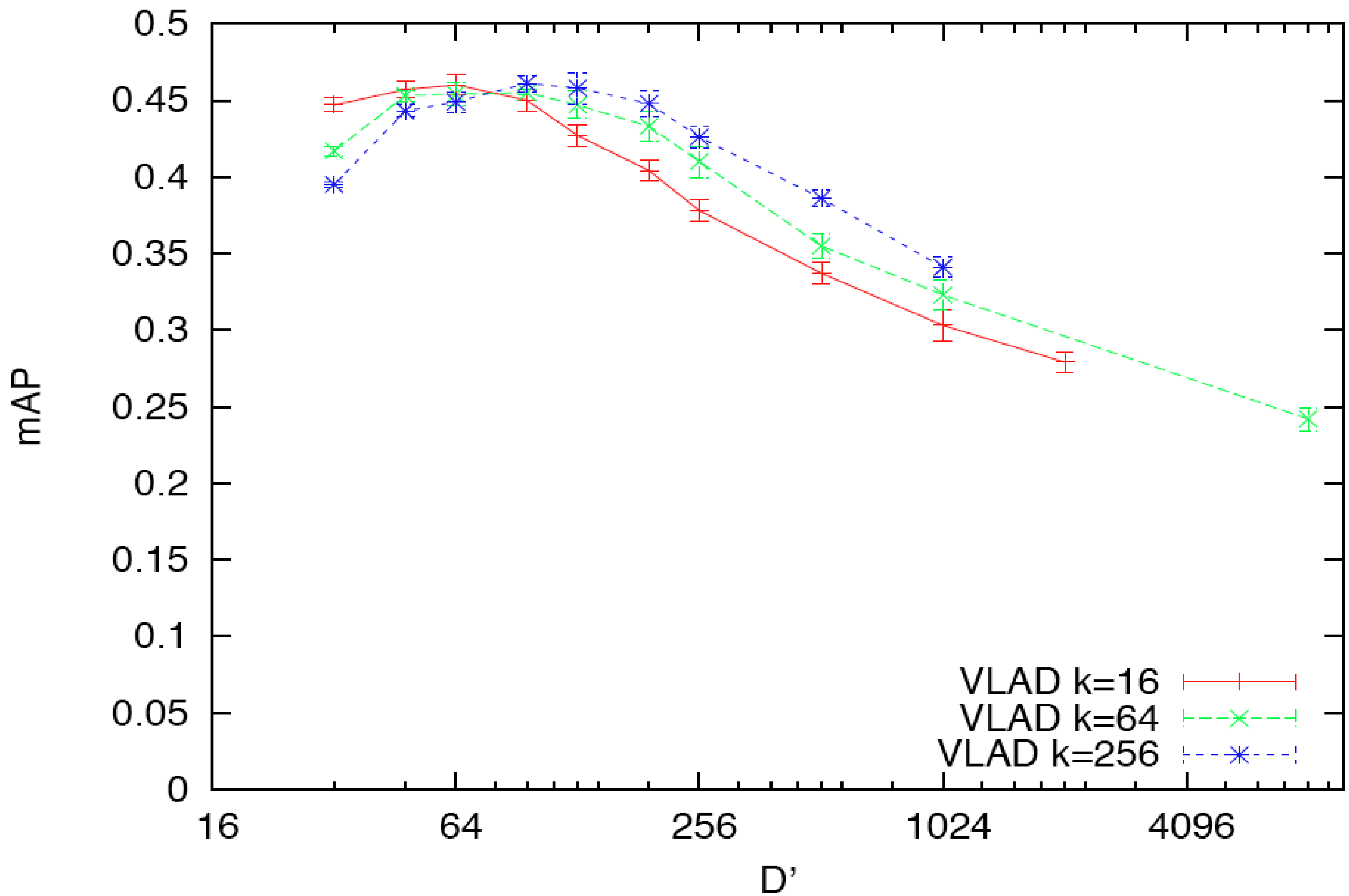
Searching with quantization: comparison with spectral Hashing



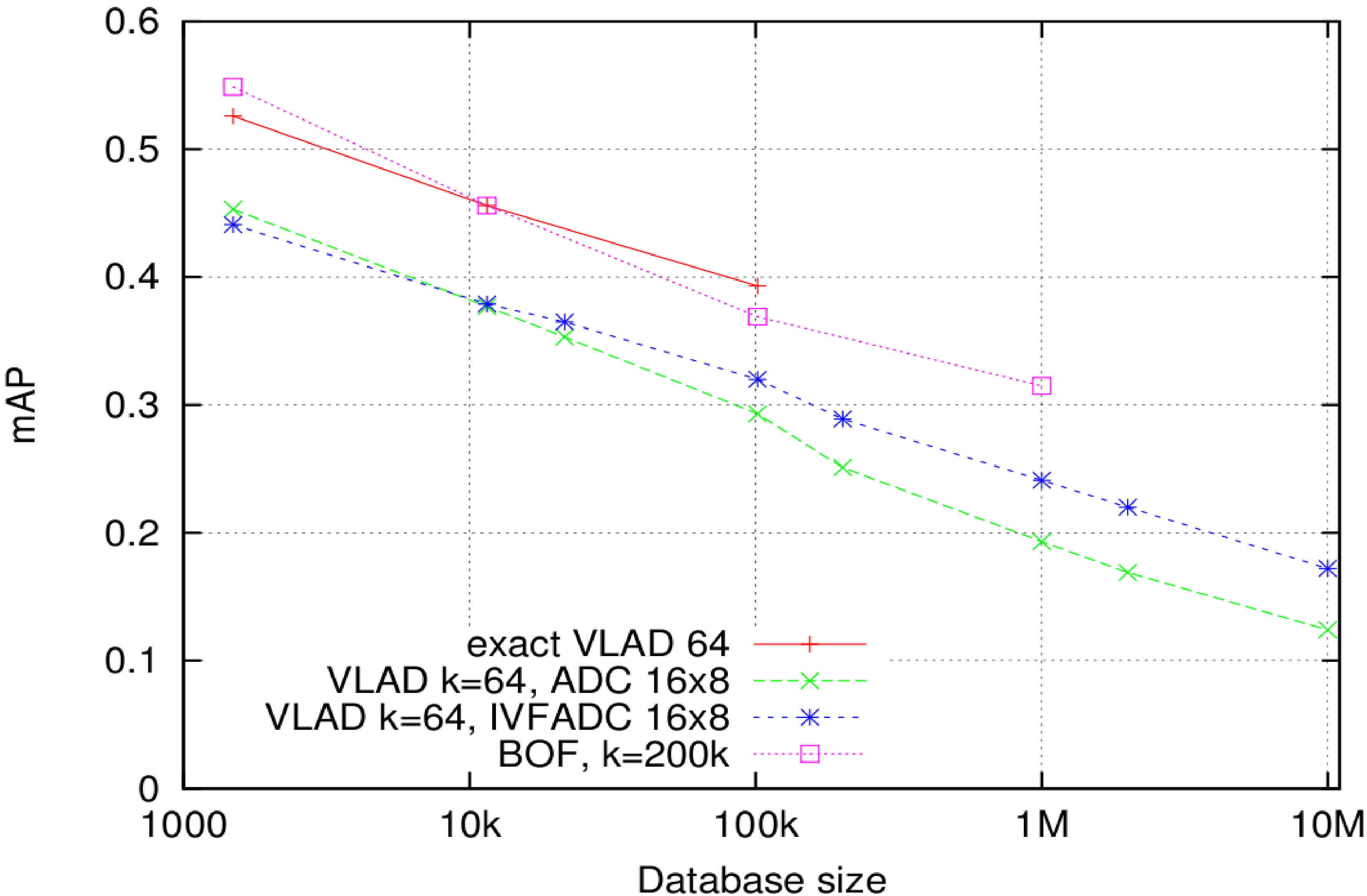
INRIA

Impact of D’ on image retrieval

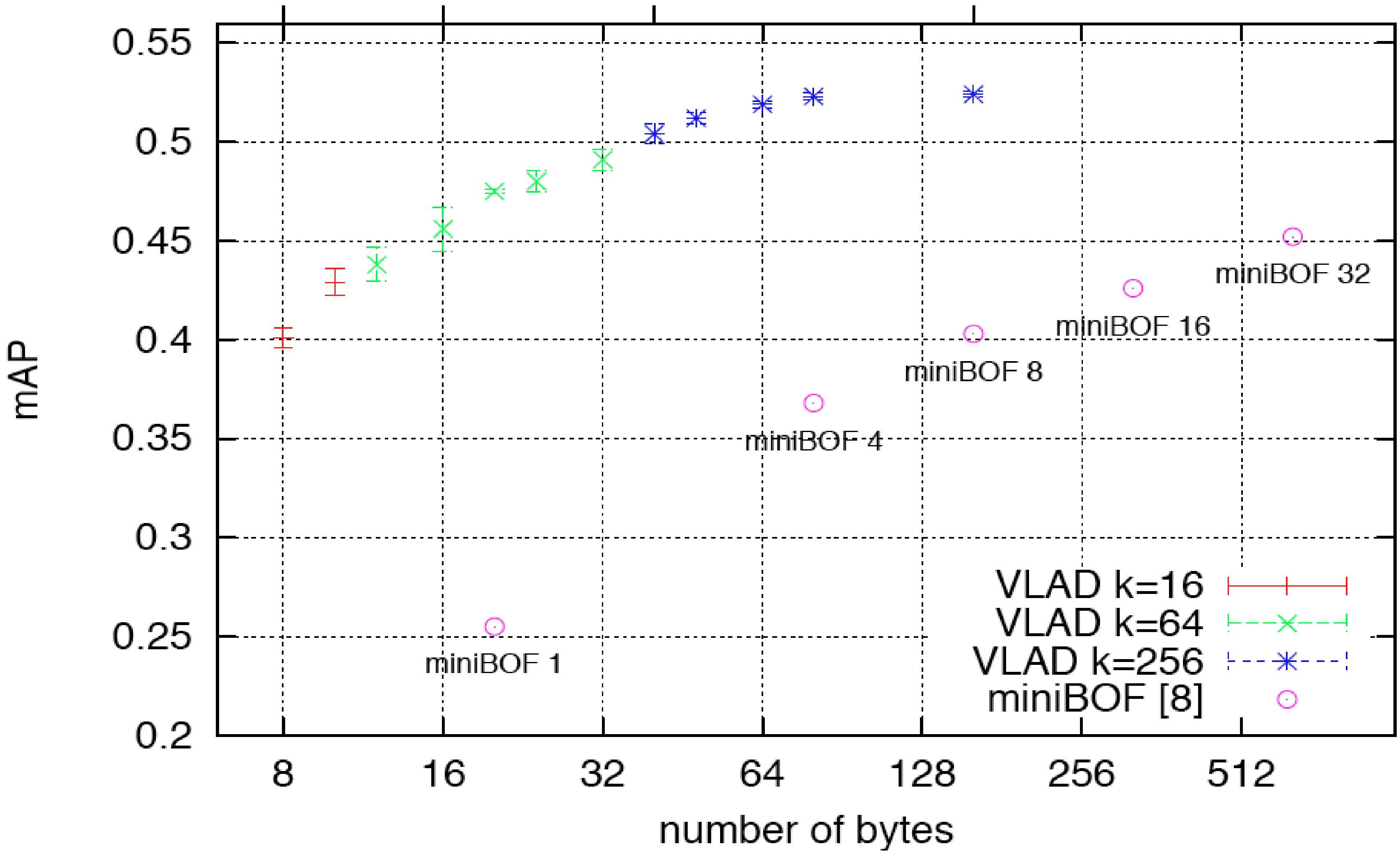
- The best choice of D’ found by minimizing the square error criterion is reasonably consistent with the optimum obtained when measuring the image search quality



Results on 10 million images

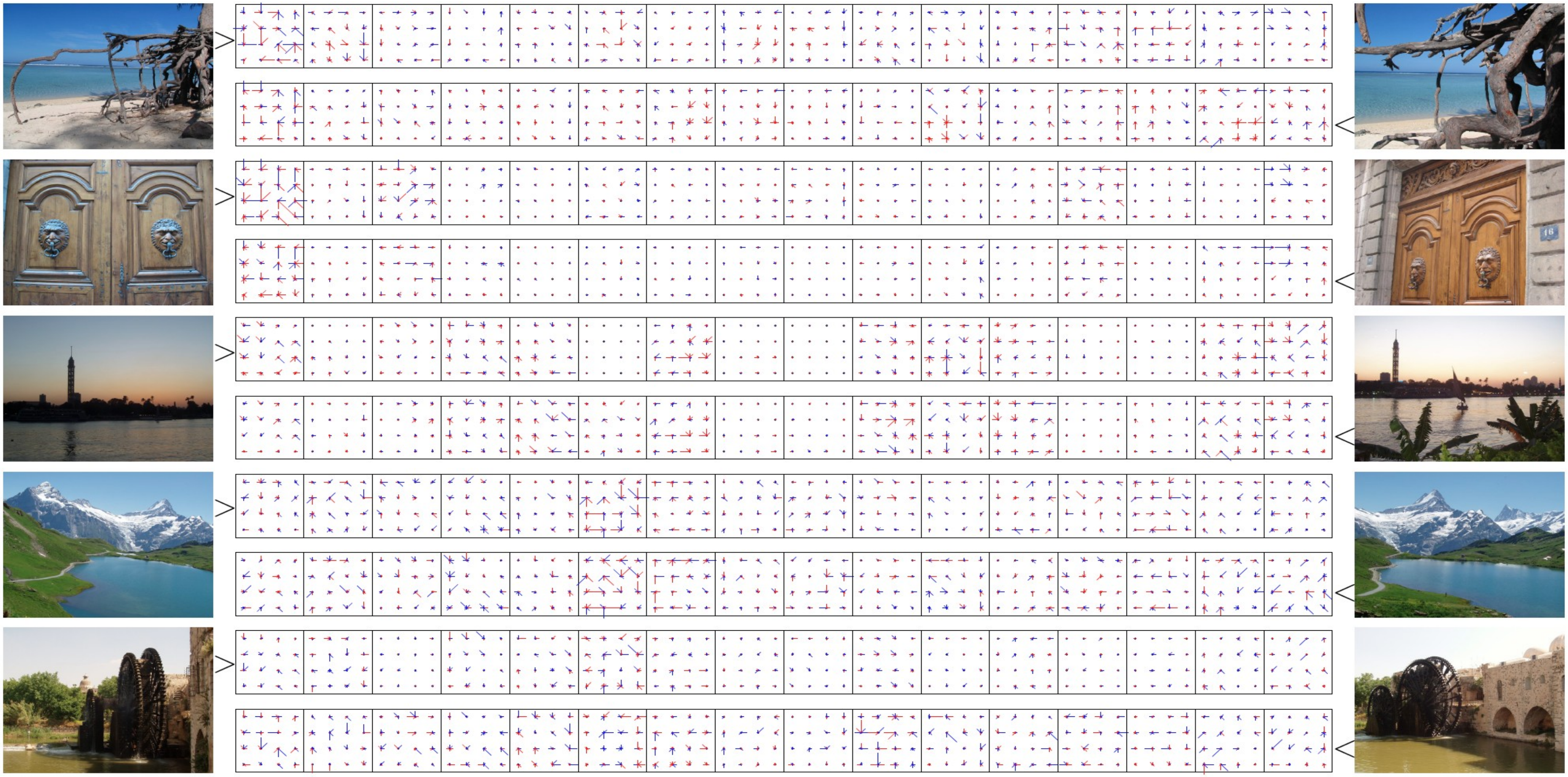


Results: comparison with « Packing BOF » (Holidays dataset)

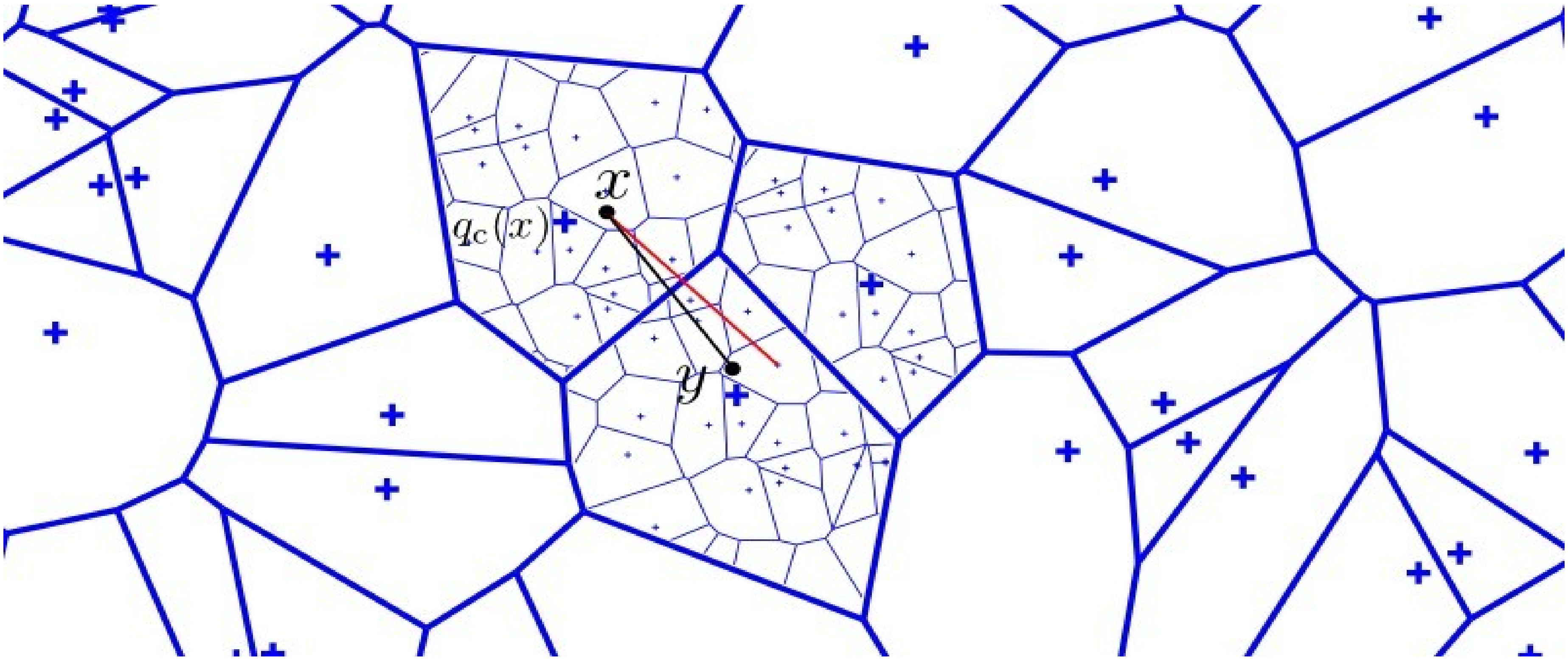


INRIA

VLAD: other examples



Combination with an inverted file



INRIA

Related work on large scale image search

- Global descriptors:
  - GIST descriptors with Spectral Hashing or similar techniques [Torralba & al 08]

→ very limited invariance to scale/rotation/crop: use local descriptors

- Bag-of-features [Sivic & Zisserman 03]
  - Large (hierarchical) vocabularies [Nister Stewenius 06]
  - Improved descriptor representation [Jégou et al 08, Philbin et al 08]
  - Geometry used in index [Jégou et al 08, Perdoc'h et al 09]
  - Query expansion [Chum et al 07]

→ memory tractable for a few million images only

- Efficiency improved by
  - Min-hash and Geometrical min-hash [Chum et al. 07-09]
  - compressing the BoF representation [Jégou et al. 09]

→ But still hundreds of bytes are required to obtain a "reasonable quality"