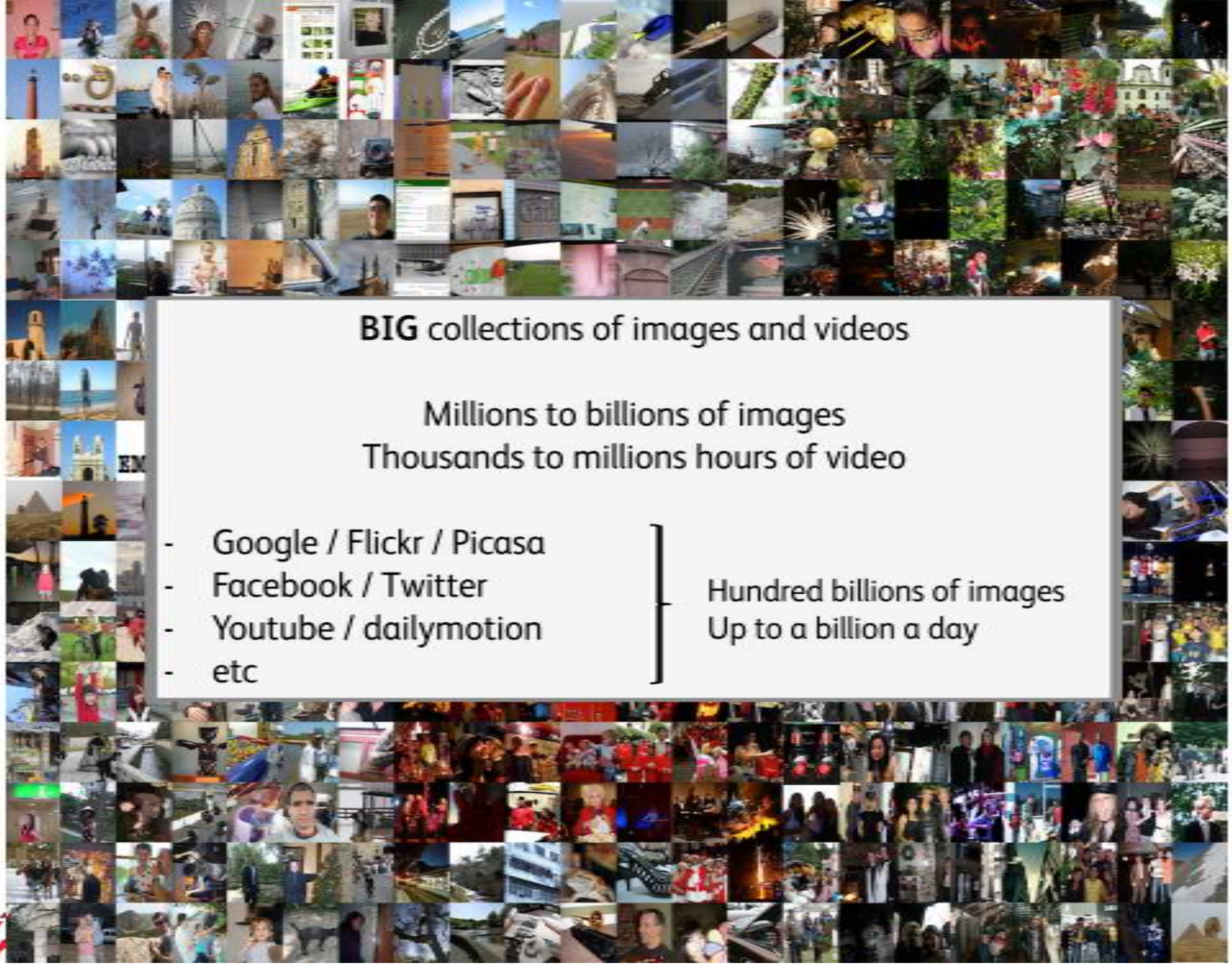
From thousands to billions: scaling up image indexing

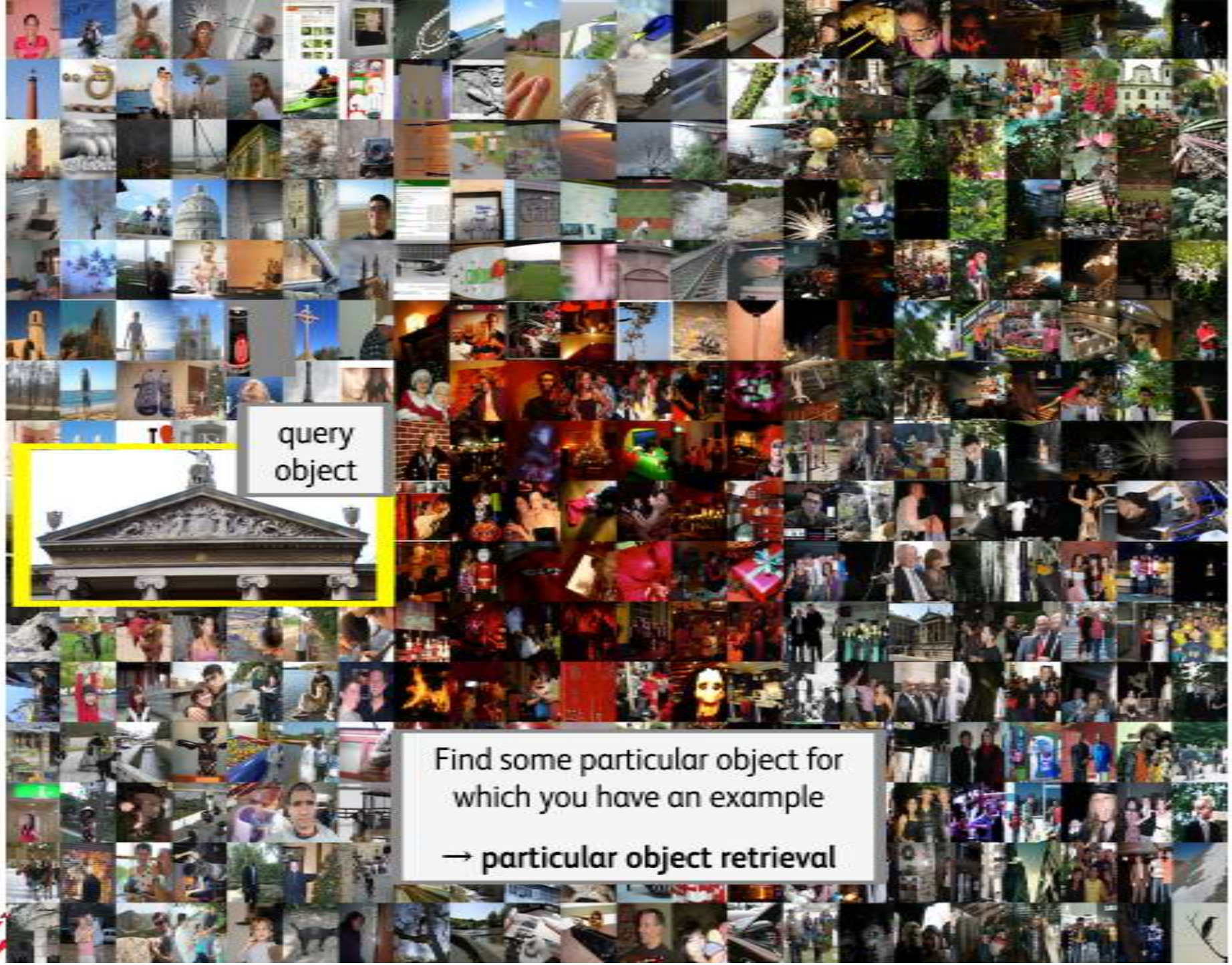
Hervé Jégou

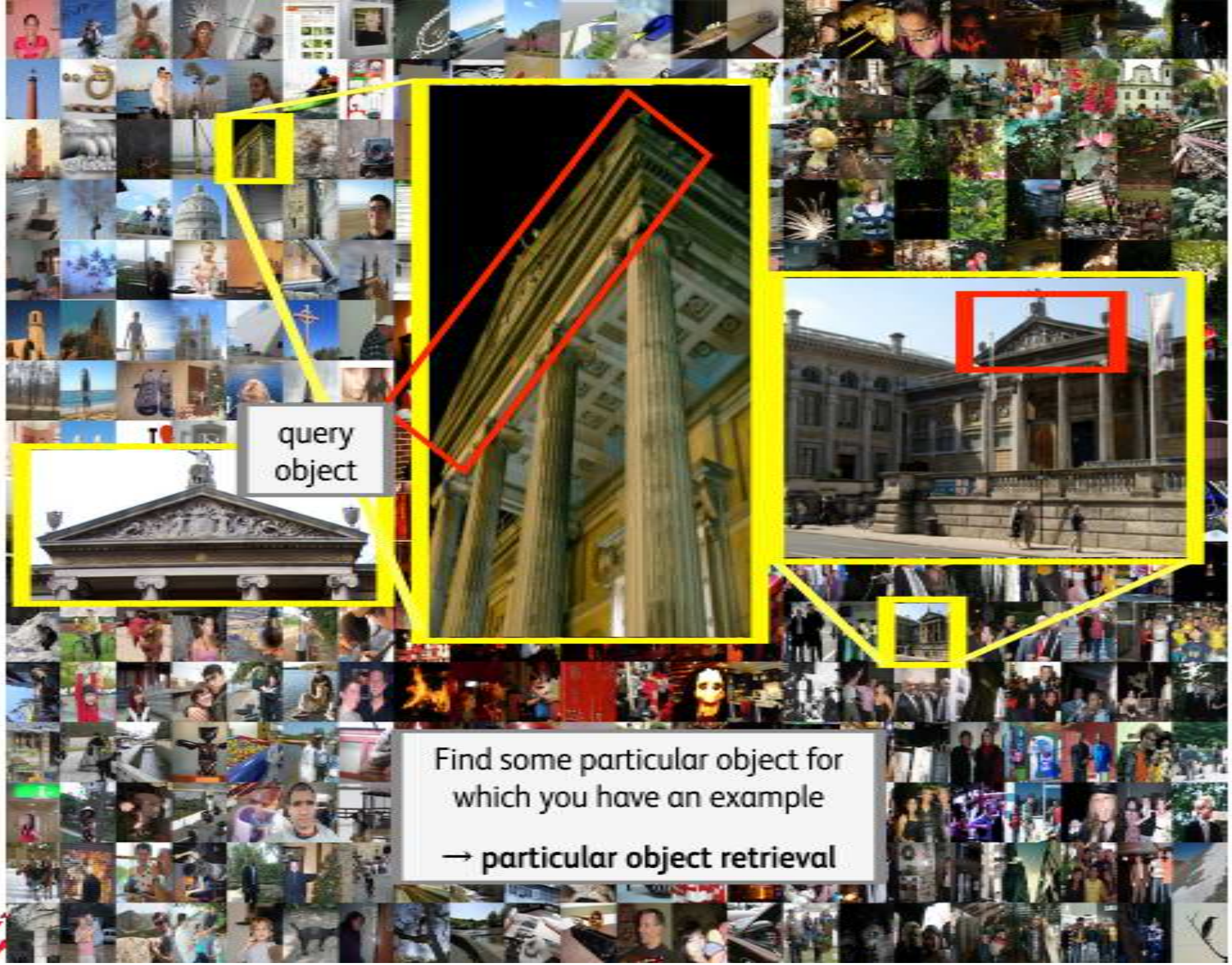
Journée CominLabs, March 23rd 2015


BIG collections of images and videos
Millions to billions of images
Thousands to millions hours of video
- Google / Flickr / Picasa
- Facebook / Twitter
- Youtube / dailymotion
- etc
Hundred billions of images
Up to a billion a day


query
image
Find all images that depict visual
content similar to the query
(same scene, location, etc)
→ image retrieval/annotation

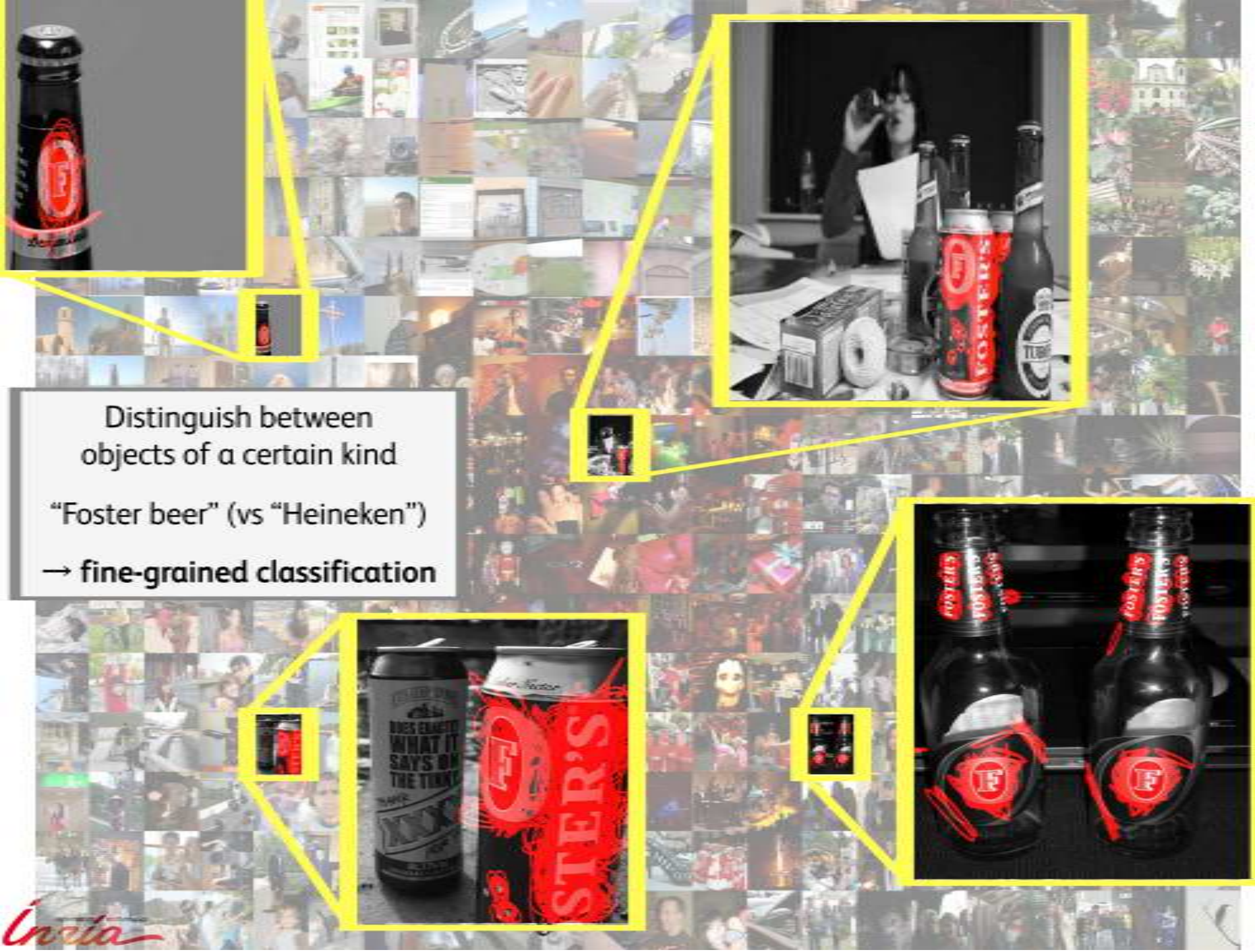

query
image
Find all images that depict visual
content similar to the query
(same scene, location, etc)
→ image retrieval/annotation


query object
Find some particular object for which you have an example
→ particular object retrieval


query object
Find some particular object for which you have an example
→ particular object retrieval


Find all images where an object of a "class" appears:
Find all "**trees**"
→ **image classification**


Find all images where an object of a "class" appears:
Find all "**trees**"
→ **image classification**


Distinguish between
objects of a certain kind
"Foster beer" (vs "Heineken")
→ fine-grained classification

Visual Search

- Google goggles on Android

The QBIC system

- Introduced by IBM in 1995

QBIC* lets users find pictorial information in large image and video databases based on color, shape, texture, and sketches. QBIC technology is part of several IBM products.

- Search in 7500 images

Query by Image and Video Content: The QBIC System

Myron Flickner, Harpreet Sawhney, Wayne Niblack, Jonathan Ashley, Qian Huang, Byron Dom, Monika Gorkani, Jim Hafner, Denis Lee, Dragutin Petkovic, David Steele, and Peter Yanker

IBM Almaden Research Center

QBIC* lets users find pictorial information in large image and video databases based on color, shape, texture, and sketches. QBIC technology is part of several IBM products.

Picture yourself as a fashion designer needing images of fabrics with a particular mixture of colors, a museum cataloger looking for artifacts of a particular shape and textured pattern, or a movie producer needing a video clip of a red car-like object moving from right to left with the camera zooming. How do you find these images? Even though today's technology enables us to acquire, manipulate, transmit, and store vast on-line image and video collections, the search methodologies used to find pictorial information are still limited due to difficult research problems (see "Semantic versus nonsemantic" sidebar). Typically, these methodologies depend on file IDs, keywords, or text associated with the images. And, although powerful, they

- don't allow queries based directly on the visual properties of the images,
- are dependent on the particular vocabulary used, and
- don't provide queries for images similar to a given image.

Research on ways to extend and improve query methods for image databases is widespread, and results have been presented in workshops, conferences,[1,2] and surveys.

We have developed the QBIC (Query by Image Content) system to explore content-based retrieval methods. QBIC allows queries on large image and video databases based on

- example images,
- user-constructed sketches and drawings,
- selected color and texture patterns,

Semantic versus nonsemantic information

At first glance, content-based querying appears deceptively simple because we humans seem to be so good at it. If a program can be written to extract semantically relevant text phrases from images, the problem may be solved by using currently available text-search technology. Unfortunately, in an unconstrained environment, the task of writing this program is beyond the reach of current technology in image understanding. At an artificial intelligence conference several years ago, a challenge was issued to the audience to write a program that would identify all the dogs pictured in a children's book, a task most 3-year-olds can easily accomplish. Nobody in the audience accepted the challenge, and this remains an open problem.

Perceptual organization—the process of grouping image features into meaningful objects and attaching semantic descriptions to scenes through model matching—is an unsolved problem in image understanding. Humans are much better than computers at extracting semantic descriptions from pictures. Computers, however, are better than humans at measuring properties and retaining these in long-term memory.

One of the guiding principles used by QBIC is to let computers do what they do best—quantifiable measurement—and let humans do what they do best—attaching semantic meaning. QBIC can find "fish-shaped objects," since shape is a measurable property that can be extracted. However, since fish occur in many shapes, the only fish that will be found will have a shape close to the drawn shape. This is not the same as the much harder semantical query of finding all the pictures of fish in a pictorial database.

The QBIC system

- Introduced by IBM in 1995

QBIC* lets users

find pictorial information

in large image and video

databases based on color,

shape, texture, and sketches.

QBIC technology is part of

several IBM products.

- Search in 7500 images

1995 → 2015: 20 years

Corrected by Moore law:
7500 → 30 million images

Query by Image and Video Content: The QBIC System

Myron Flickner, Harpreet Sawhney, Wayne Niblack, Jonathan Ashley, Qian Huang, Byron Dom, Monika Gorkani, Jim Hafner, Denis Lee, Dragutin Petkovic, David Steele, and Peter Yanker

IBM Almaden Research Center

QBIC* lets users find pictorial information in large image and video databases based on color, shape, texture, and sketches. QBIC technology is part of several IBM products.

Picture yourself as a fashion designer needing images of fabrics with a particular mixture of colors, a museum cataloger looking for artifacts of a particular shape and textured pattern, or a movie producer needing a video clip of a red car-like object moving from right to left with the camera zooming. How do you find these images? Even though today's technology enables us to acquire, manipulate, transmit, and store vast on-line image and video collections, the search methodologies used to find pictorial information are still limited due to difficult research problems (see "Semantic versus nonsemantic" sidebar). Typically, these methodologies depend on file IDs, keywords, or text associated with the images. And, although powerful, they

- don't allow queries based directly on the visual properties of the images,
- are dependent on the particular vocabulary used, and
- don't provide queries for images similar to a given image.

Research on ways to extend and improve query methods for image databases is widespread, and results have been presented in workshops, conferences,[1,2] and surveys.

We have developed the QBIC (Query by Image Content) system to explore content-based retrieval methods. QBIC allows queries on large image and video databases based on

- example images,
- user-constructed sketches and drawings,
- selected color and texture patterns,

Semantic versus nonsemantic information

At first glance, content-based querying appears deceptively simple because we humans seem to be so good at it. If a program can be written to extract semantically relevant text phrases from images, the problem may be solved by using currently available text-search technology. Unfortunately, in an unconstrained environment, the task of writing this program is beyond the reach of current technology in image understanding. At an artificial intelligence conference several years ago, a challenge was issued to the audience to write a program that would identify all the dogs pictured in a children's book, a task most 3-year-olds can easily accomplish. Nobody in the audience accepted the challenge, and this remains an open problem.

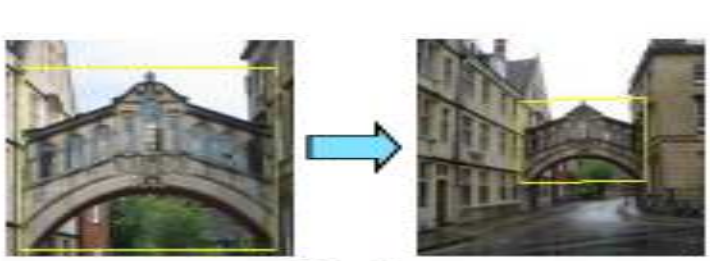
Perceptual organization—the process of grouping image features into meaningful objects and attaching semantic descriptions to scenes through model matching—is an unsolved problem in image understanding. Humans are much better than computers at extracting semantic descriptions from pictures. Computers, however, are better than humans at measuring properties and retaining these in long-term memory.

One of the guiding principles used by QBIC is to let computers do what they do best—quantifiable measurement—and let humans do what they do best—attaching semantic meaning. QBIC can find "fish-shaped objects," since shape is a measurable property that can be extracted. However, since fish occur in many shapes, the only fish that will be found will have a shape close to the drawn shape. This is not the same as the much harder semantical query of finding all the pictures of fish in a pictorial database.

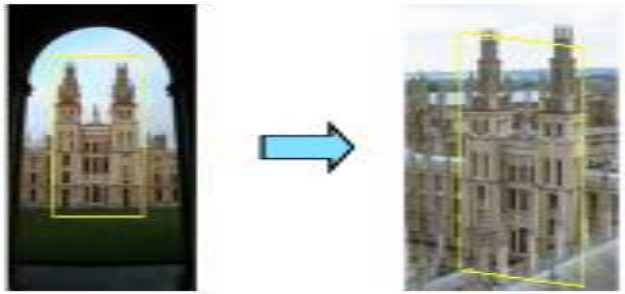
Image description

- Image processing : analysis step (=description)
  - Convert an image to a mathematical representation
  - Similar images have "similar" representations

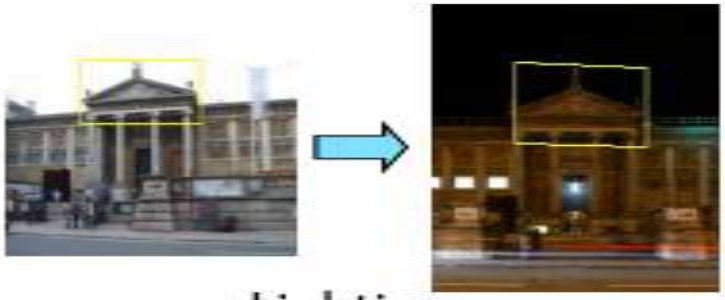
⇨ needs **invariant description**



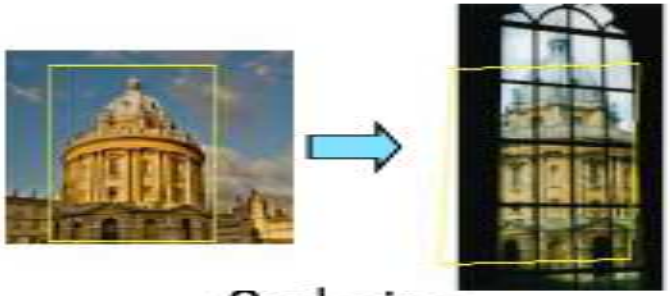
Scale



Viewpoint



Lighting



Occlusion

Inria

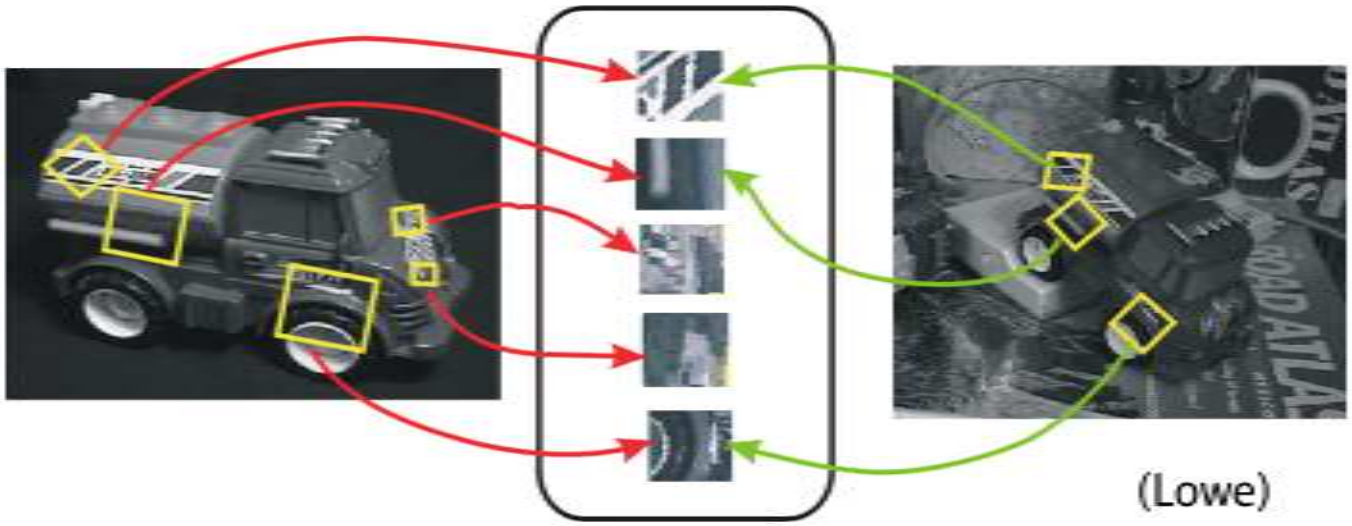
Image description

- But the representation should be discriminative enough
  ⇨ careful selection of what should be invariant for the application

- From this point of view, QBIC is limited (description not good enough)

Inria

Matching images with local descriptors



(Lowe)

Image content is transformed into local features that are invariant to geometric and photometric transformations

[Schmid'95,97] a voting algorithm based on parts
→ able to index 1000 images

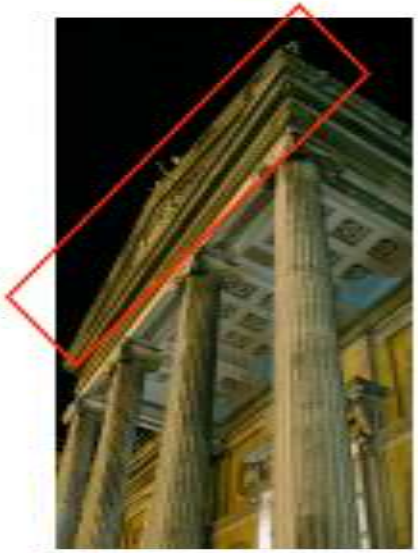
[Lowe'99,04] **the SIFT descriptor**

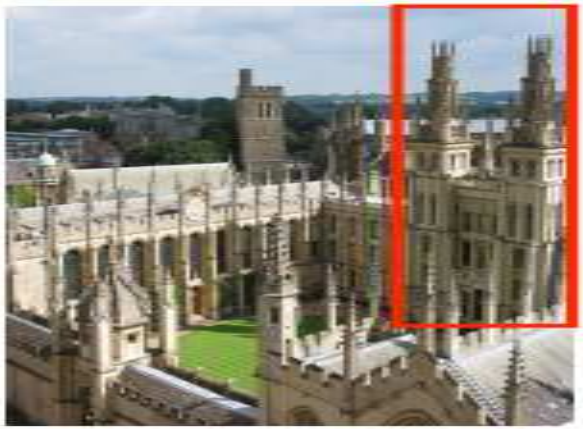
Inria

Local description is much more powerful

Query (objects)

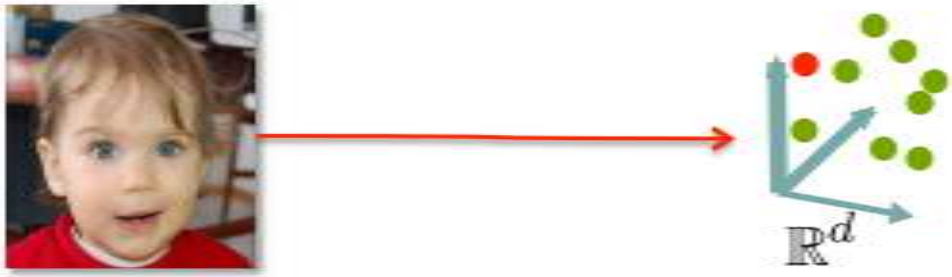
Images marked as relevant



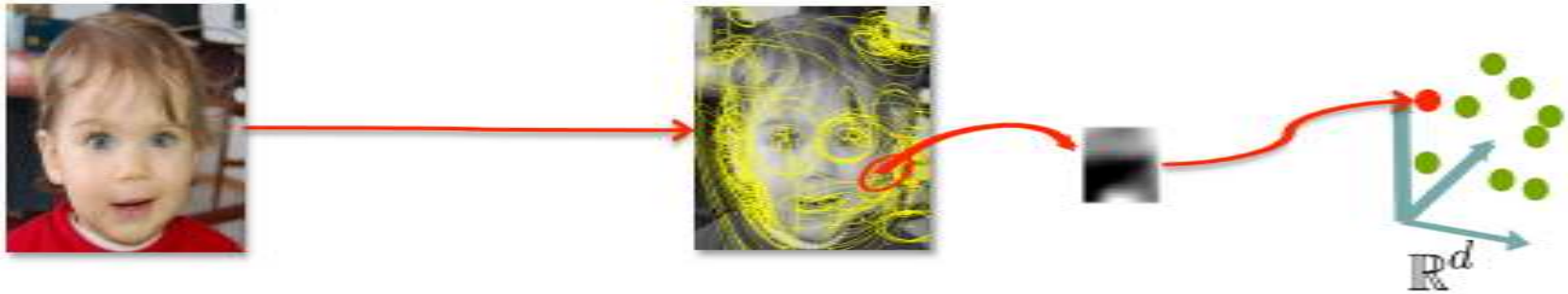


http://www.robots.ox.ac.uk/~vgg/data/oxbuildings/index.html

Glocal description vs local description



Global description
A single vector is extracted from the image



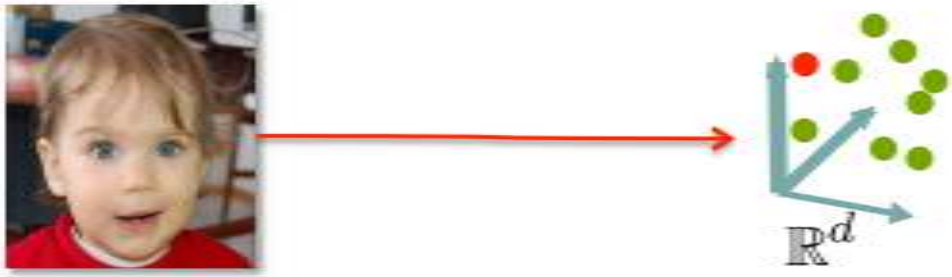
Local description
An image is represented by a set of vectors

$$\mathcal{X} = \{x_1, \ldots, x_m\}$$

→ This enables finding small objects in images

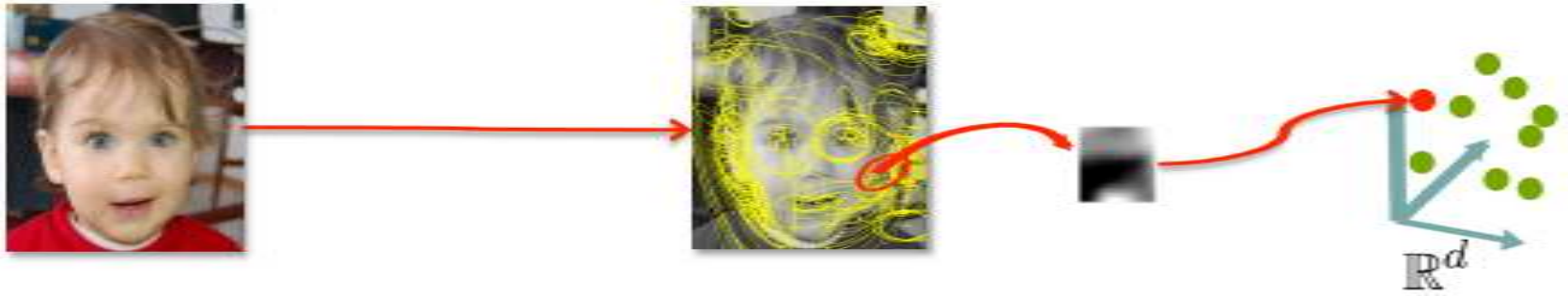
Inria

Glocal description vs local description



Global description
A single vector is extracted from the image

Search complexity
O(N*d) operations



Local description
An image is represented by a set of vectors

$$\mathcal{X} = \{x_1, \ldots, x_m\}$$

→ This enables finding small objects in images

Search complexity
O(N*d*m²) operations

(m=1000, typically)

Inria

Need for efficient search algorithms

- Local description: more powerful but much more costly

- Example: our Trecvid'2010 participation to video copy detection
  - Visual channel: 3.45 billion image descriptors (SIFT, d=128)
  - Audio channel: 140 million descriptors (filter banks, d=144)

- Challenge: Exhaustive linear search intractable for search:
  → **trillions** of high dimensional vector comparisons
  = in the order of 10^{14} elementary operations

Efficient indexing and similarity search is a critical step

Inria

Scalability
(on 1 machine)

- **Scalable systems with Global descriptors (image-level)**

- QBIC'95: 7.5k (but in 1995!)
- Cortina: Quack et al. – ACMM'04
  3 million images (10M)
- Torralba et al. – CVPR'08
  12.9M – 74ms with 30bit codes
- Douze et al.'2009 – CIVR'09
  110 million images – 180ms

Scalable systems local descriptors (object instance)

- Schmid 97 – PAMI'97
  1k images
- Sivic et al. – CVPR'03: "Video-Google"
  5k images
- Joly et al. – CIVR'03
  6M video keyframes – 120M descriptors
- Nister et al. – CVPR'06
  50k images (then 1M images)
- Chum et al. – PAMI 2010
  clustering 5M images
- J. et al. – CVPR'10
  10M images (extended to 100M in 2012)

Inria

Focus on similarity search

- Typical task: find the nearest neighbor of a given query vector:

$$\mathrm{NN}(x) = \arg\min_{y \in \mathcal{Y}} ||x - y||^2$$

- Problem: Exhaustive search has complexity O(n*d)
  - And no good exact strategy for high-dimensional vectors

- For large collections: **approximate** nearest neighbor search (ANN)
  - Find neighbors with high probability only

  → huge speed-up

Inria

Similarity search

- Typical task: find the nearest neighbor of a given query vector:

$$\mathrm{NN}(x) = \arg\min_{y \in \mathcal{Y}} ||x - y||^2$$

- Problem: Exhaustive search has complexity O(n*d)
  - And no good exact strategy for high-dimensional vectors

- For large collections: **approximate** nearest neighbor search (ANN)
  - Find neighbors with high probability only

  → huge speed-up

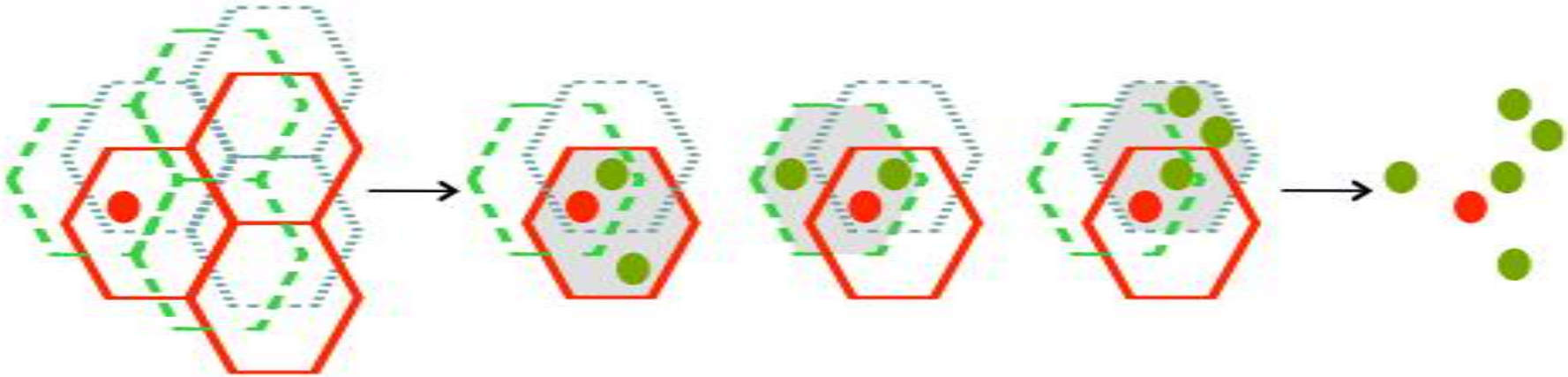
Three (contradictory) performance criteria for ANN schemes

- search quality
- speed
- **memory**

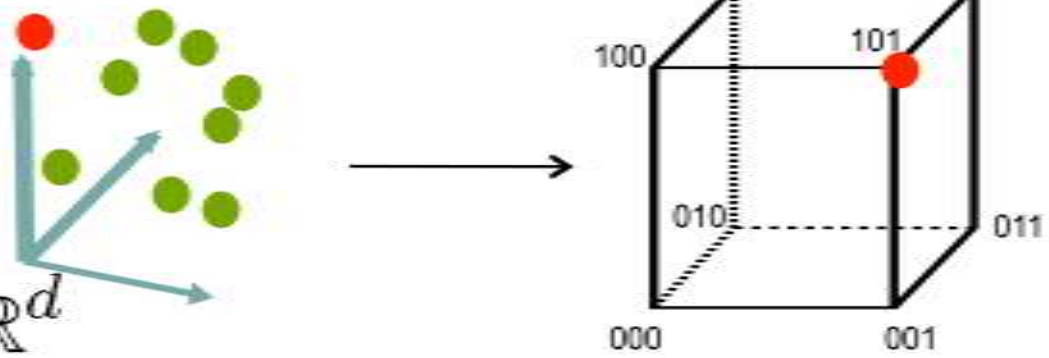
Inria

Two approaches

- Cell-probe model: avoid exhaustive search with space (multi-)partitioning
  e.g., E^{2}LSH [Indyk'98]



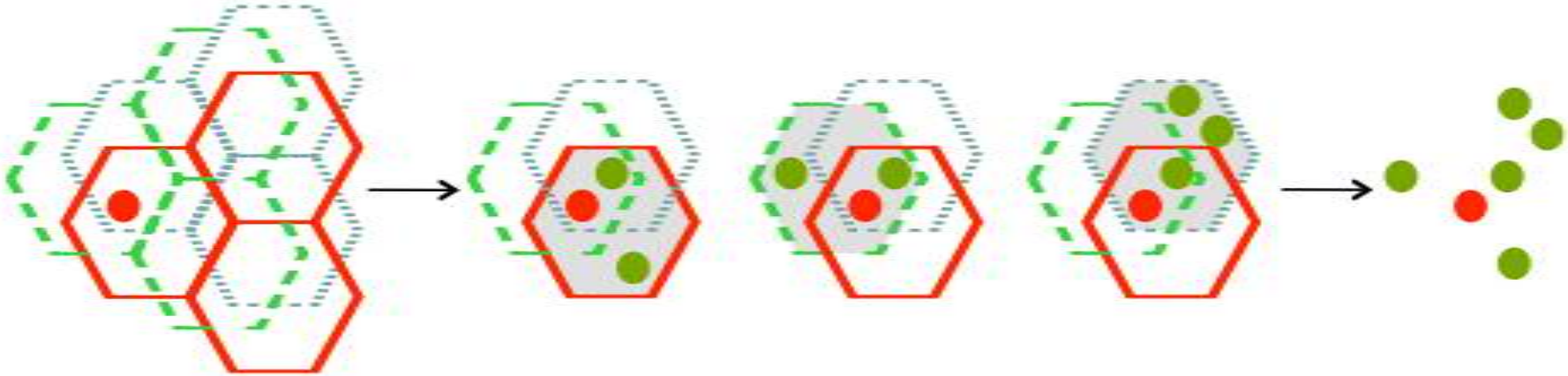
- "sketches": compact codes, approximate (faster) similarity
  - In particular, binarization techniques derived from [Charikar'02]



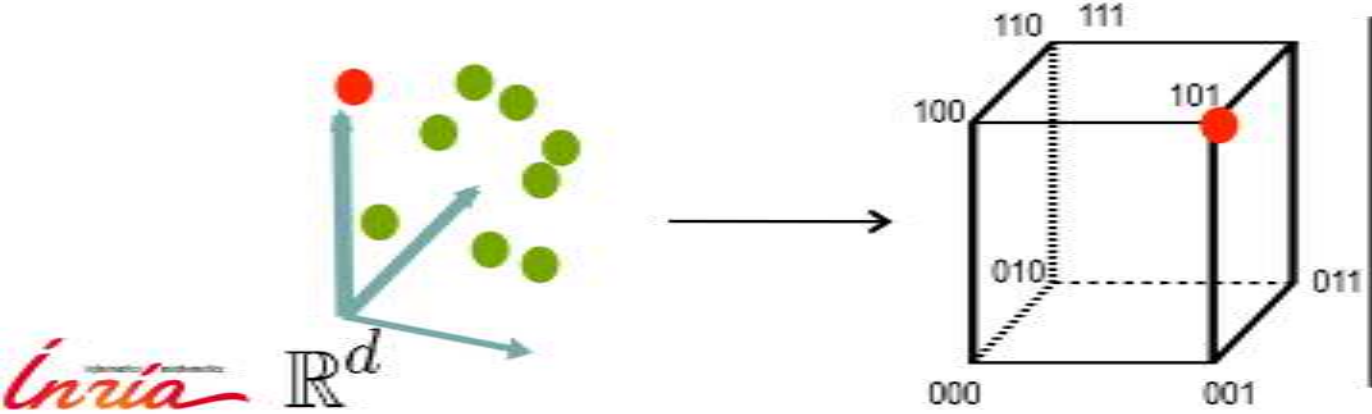
Inria

Two approaches

- Cell-probe model: avoid exhaustive search with space (multi-)partitioning
  e.g., E^{2}LSH [Indyk'98]



- "sketches": compact codes, approximate (faster) similarity
  - In particular, binarization techniques derived from [Charikar'02]



Advantage: limited memory requirements

More efficient comparison + use faster memory

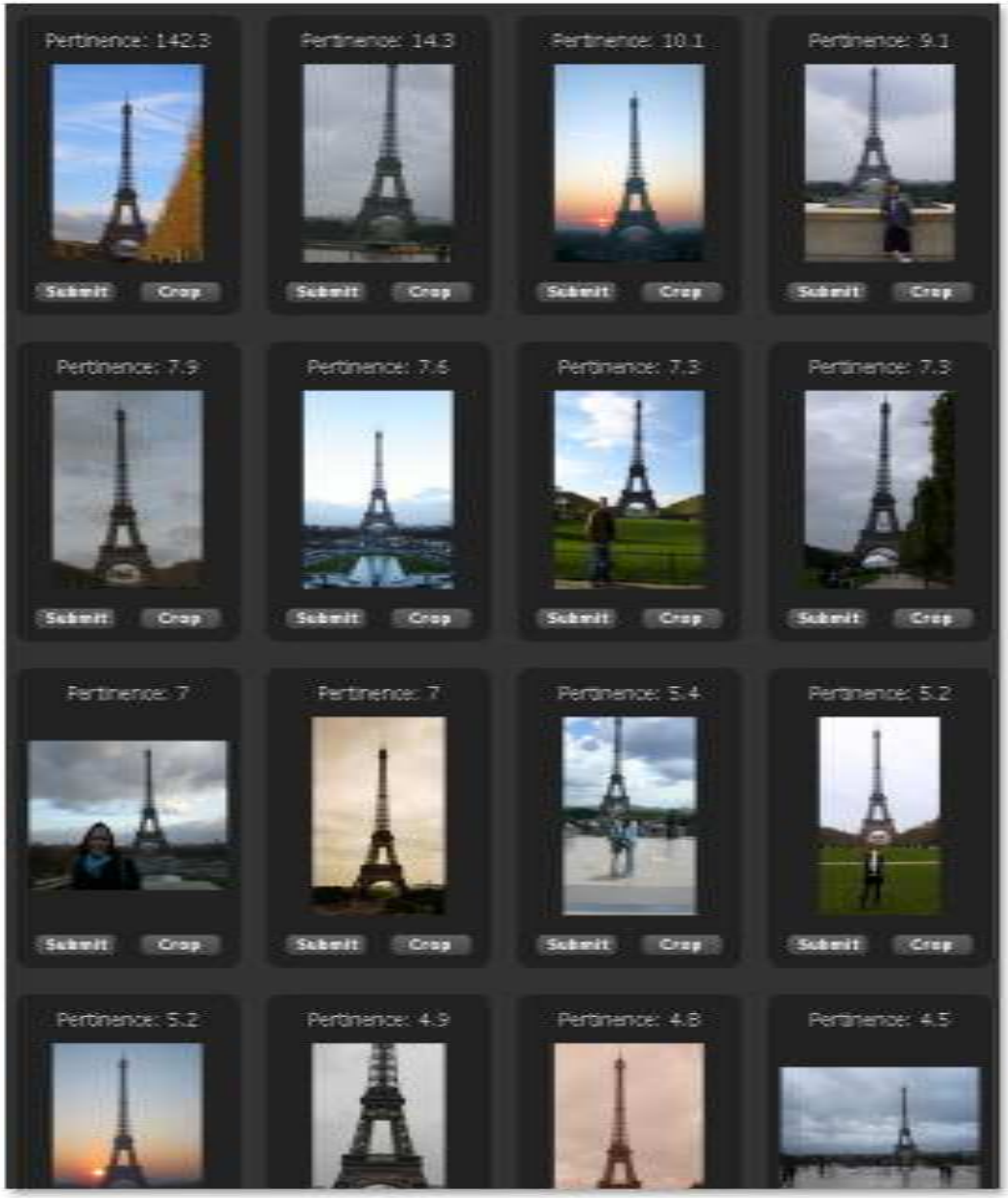
Combining cell-probe + sketches

- Inria's BigImBaz (2008-)

  http://bigimbaz.inrialpes.fr/

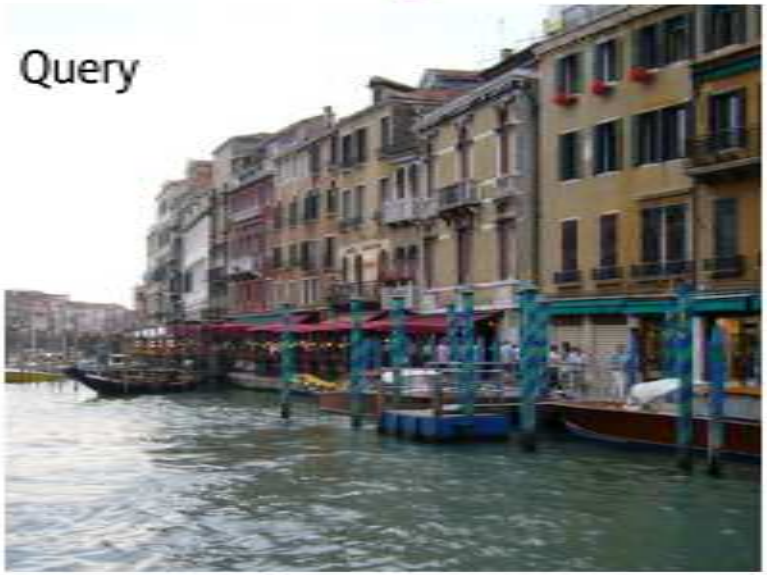
[J', Douze, Schmid, IJCV 10]

10 millions images on a big server
6kB per indexed image



Efficient pairwise matching in 1M images

Query



Ranked 1st

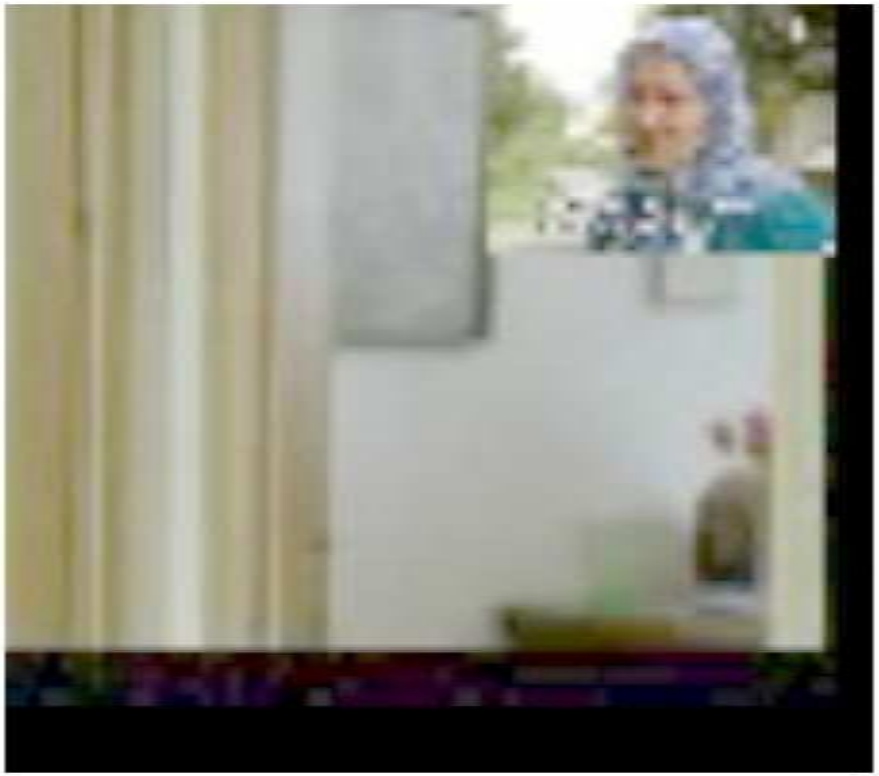
Ranked 4th

Ranked 5th

Obtained without any spatial verification

In videos

- 200H of videos in the database (720 000s)
- Task: find if a query video (3-30s) is in the database, and where

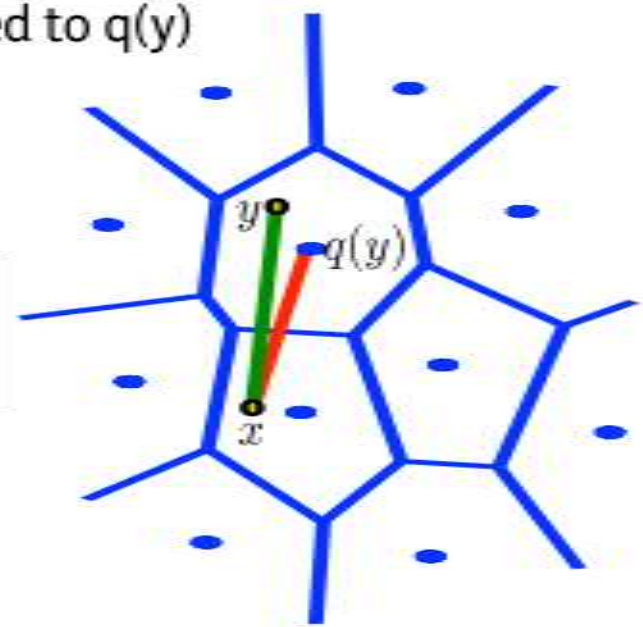


Search ≈ distance estimation ≈ compression

Joint work with Matthijs Douze and Cordelia Schmid

- x is a query vector, database vector y quantized to q(y)

$$d(x,y)^2 \approx d(x,q(y))^2$$



The error on square distances is statistically bounded by the quantization error

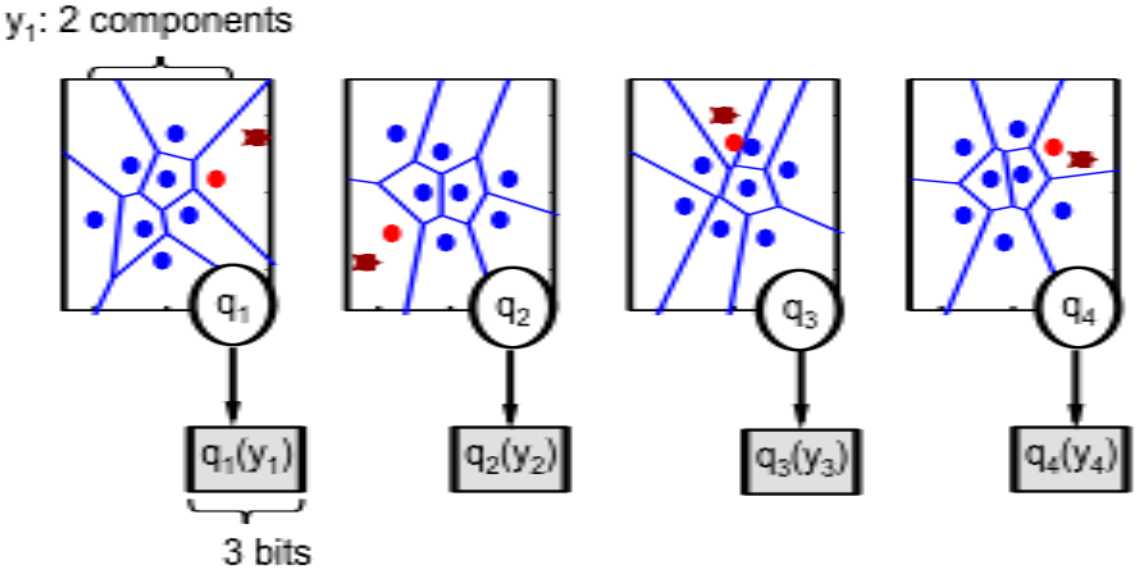
- **How to design q ?**
  - quantization should be fast enough
  - comparison should be done in the compressed domain
  - quantization is precise, i.e., many different possible indexes (ex: 2^{64})

- Regular k-means is not appropriate: not for k=2^{64} centroids

Inria

Product quantizer

- Vector split into m subvectors: $y \rightarrow [y_1 \ldots y_m]$
- Subvectors are quantized separately
- Toy example: y = 8-dim vector split into 4 subvectors of dimension 2



8^4=4,096 centroids induced

for a quantization cost equal to that of a k-means with 8 centroids

- In practice: 8 bits/subquantizer (256 centroids)
  - SIFT: m=4 (i.e., 4 bytes for a 128-dimensional descriptor)

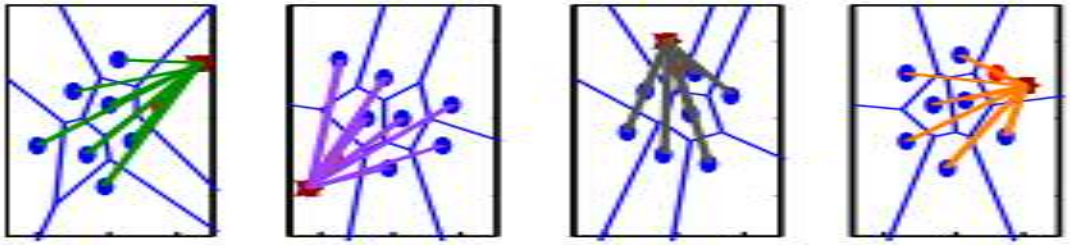
Inria

Searching with product quantization

- Estimate distances **in the compressed domain** $d(x,y)^2 \approx \sum_{i=1}^{m} d(x_i, q_i(y_i))^2$

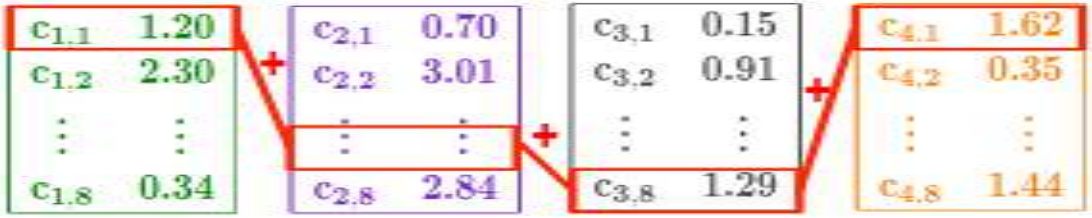
- To compute distances between query x and **many** codes:

I-



→ Precompute all distances between query subvectors and centroids:

$$d(x_i, c_{i,j})^2$$



→ Stored in look-up tables computed per query descriptor

II- For each database vector:
m-1 additions per distance estimation

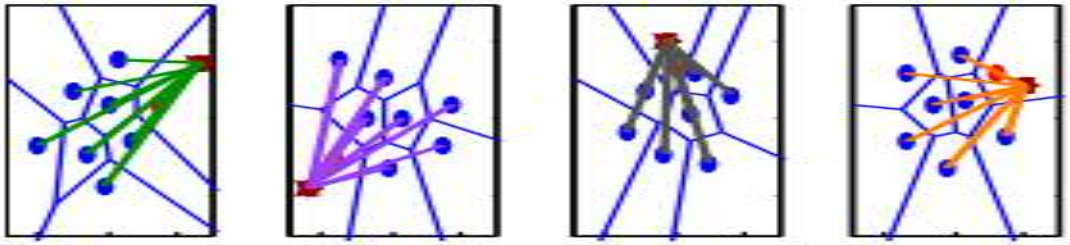
Inria

Searching with product quantization

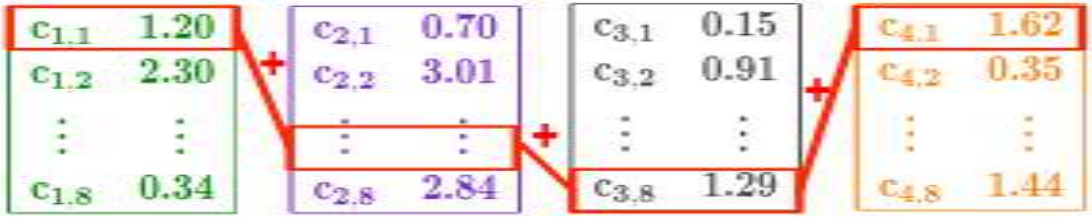
- Estimate distances **in the compressed domain** $d(x, y)^2 \approx \sum_{i=1}^{m} d(x_i, q_i(y_i))^2$

- To compute distances between query x and **many** codes:

I-



→ Precompute all distances between query subvectors and centroids: $d(x_i, c_{i,j})^2$



→ Stored in look-up tables computed per query descriptor

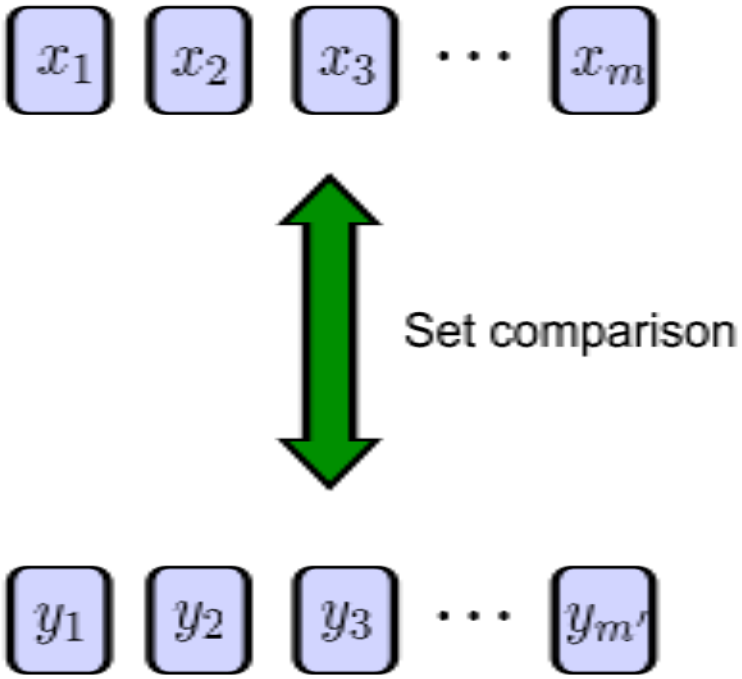
II- For each database vector:
m-1 additions per distance estimation

Similar strategies now used in
- Google Goggles
- Bing

Inria

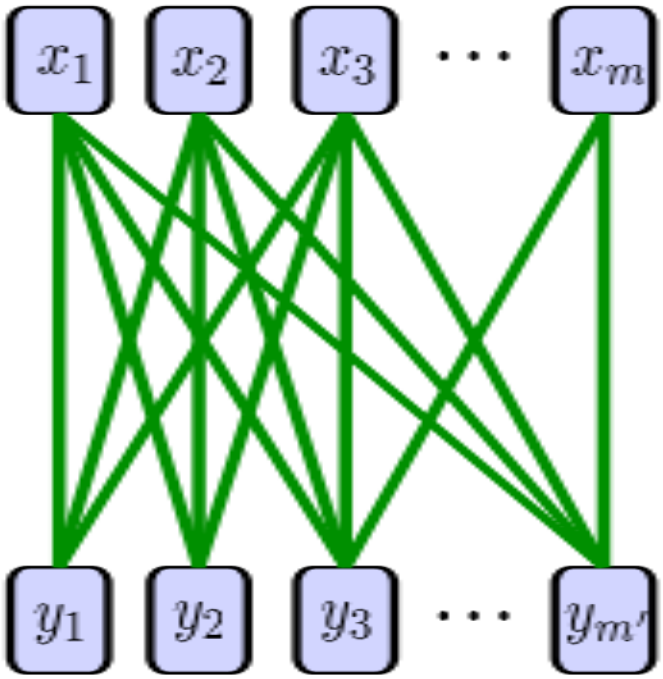
Beyond voting approaches: Match kernels

- Consider sets $\mathcal{X} = \{x_1, \ldots, x_m\}$ and $\mathcal{Y} = \{y_1, \ldots, y_{m'}\}$
  - Representing two images to be compared



Match kernels

- Consider sets $\mathcal{X} = \{x_1, \ldots, x_m\}$ and $\mathcal{Y} = \{y_1, \ldots, y_{m'}\}$
  - Representing two images to be compared

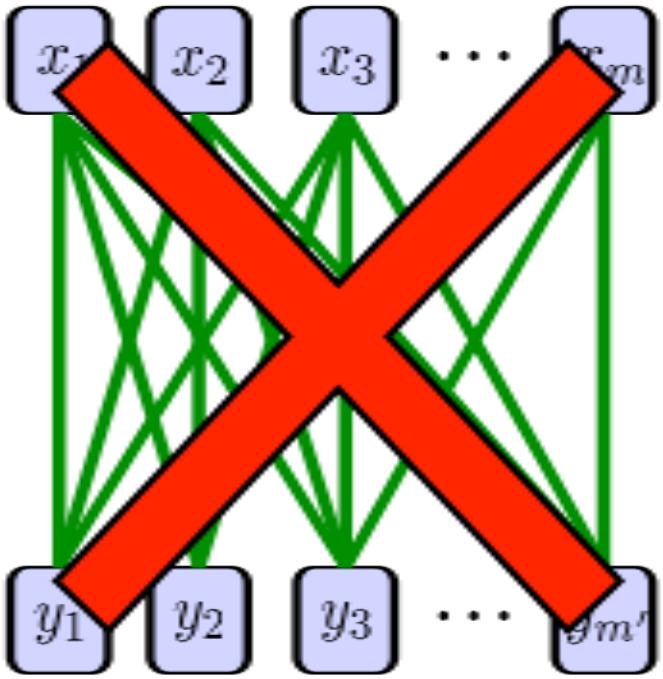


Direct pairwise matching

$$\mathcal{K}(\mathcal{X}, \mathcal{Y}) \propto \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} k(x, y)$$

Match kernels

- Consider sets $\mathcal{X} = \{x_1, \ldots, x_m\}$ and $\mathcal{Y} = \{y_1, \ldots, y_{m'}\}$
  - Representing two images to be compared



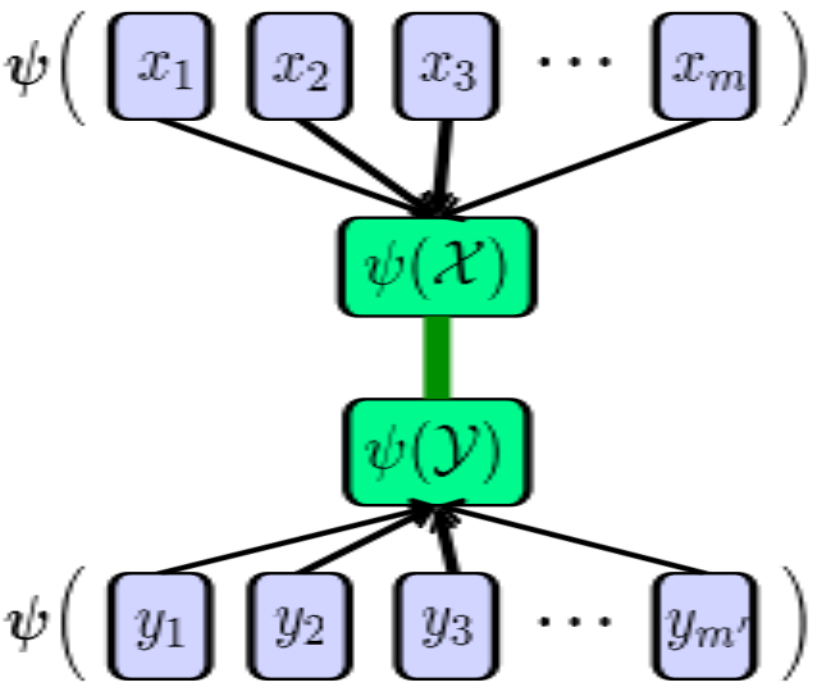
Direct pairwise matching

$$\mathcal{K}(\mathcal{X}, \mathcal{Y}) \propto \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} k(x, y)$$

Costly

$$\mathcal{O}(m \times m' \times d)$$

Match kernels: linearization



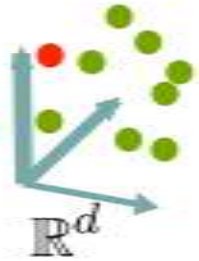
Linearize Match kernels [Bo 09]

$$\mathcal{K}(\mathcal{X},\mathcal{Y}) \propto \sum_{x\in\mathcal{X}} \sum_{y\in\mathcal{Y}} k(x,y)$$

$$\approx \boxed{\psi(\mathcal{X})^\top \psi(\mathcal{Y})}$$

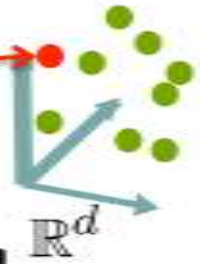
A single inner product

In fact: come back to a single descriptor



Global description
A single **high-dimensional** vector per image

Aggregation
Into 1 vector

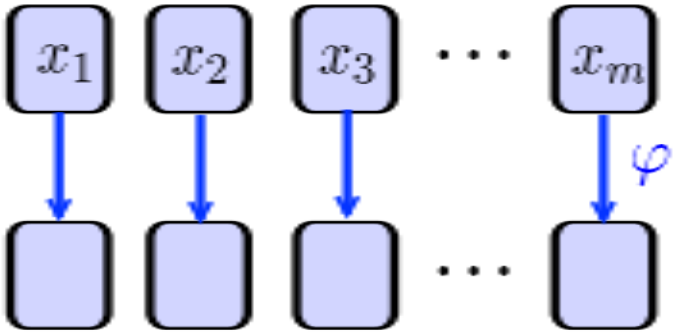


Local description
An image is represented by a set of vectors

$$\mathcal{X} = \{x_1, \ldots, x_m\}$$

Inria

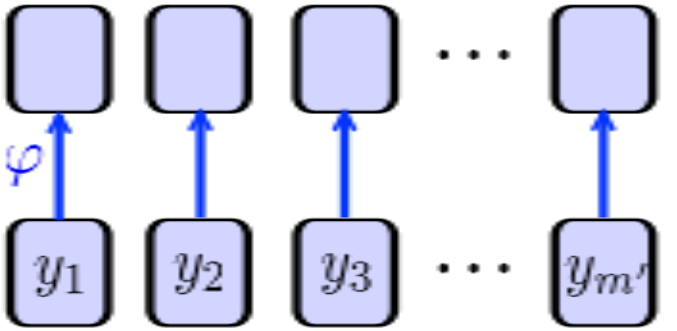
Match kernels: design



Typically implemented into 2 steps

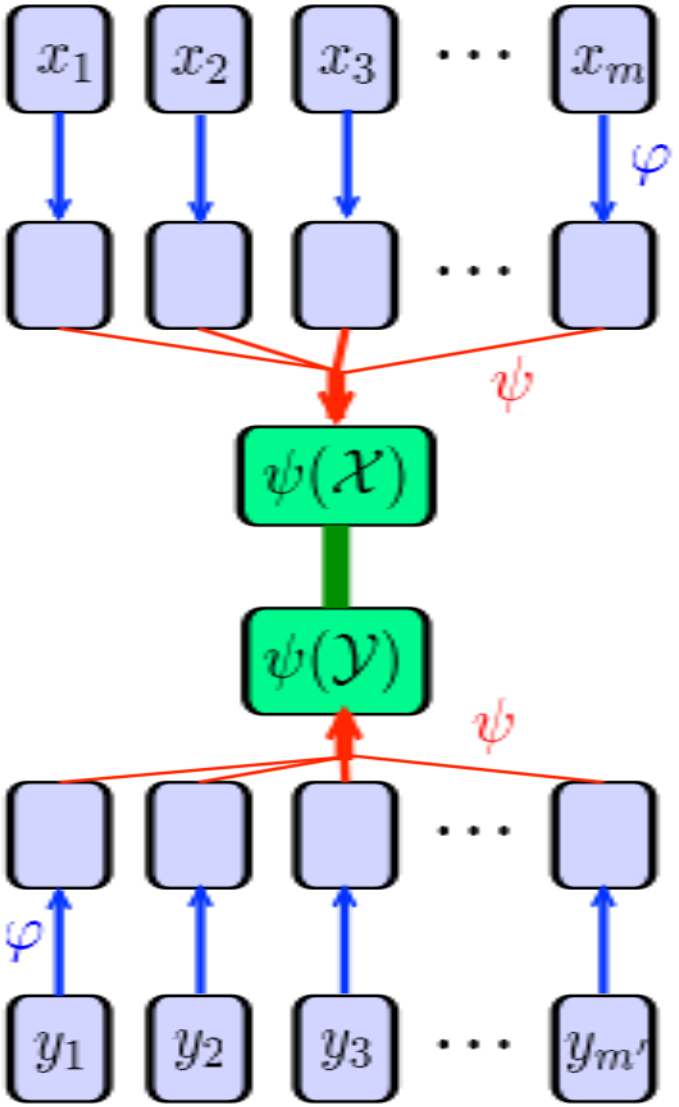
Embedding: $\varphi : \mathbb{R}^d \to \mathbb{R}^D$

$x \mapsto \varphi(x)$



Inria

Match kernels: design



Typically implemented into 2 steps

Embedding: $\varphi : \mathbb{R}^d \to \mathbb{R}^D$

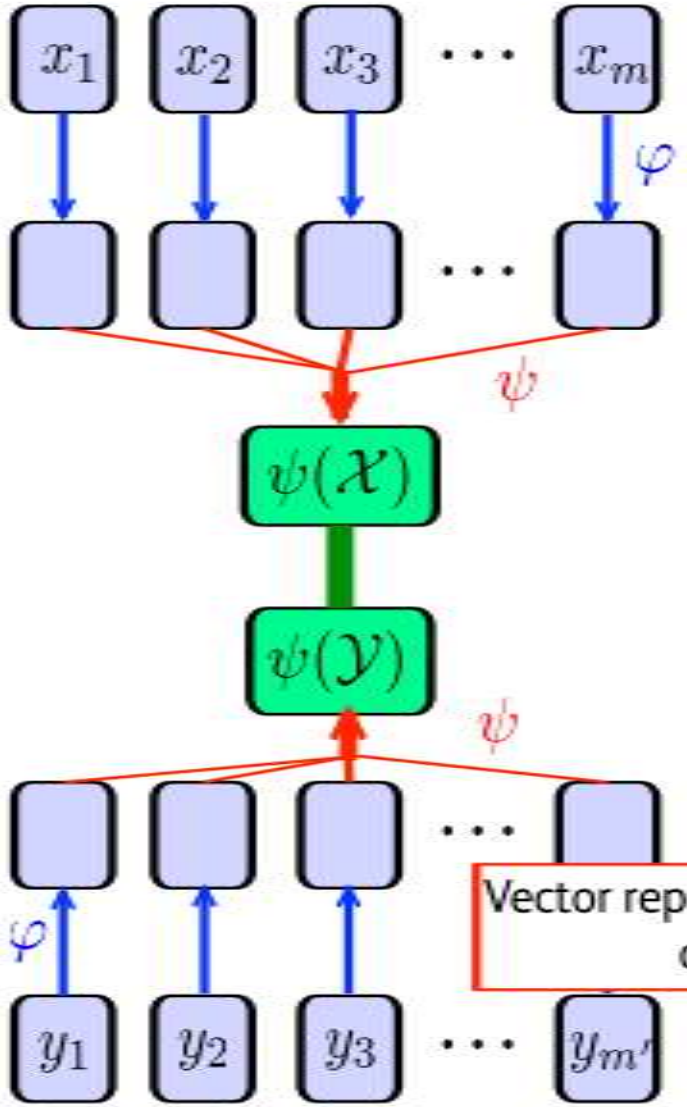
$x \mapsto \varphi(x)$

Aggregation: $\psi : (\mathbb{R}^D)^* \to \mathbb{R}^D$

$\mathcal{X} \mapsto \psi(\mathcal{X})$

Inria

Match kernels: design



Typically implemented into 2 steps

Embedding: $\varphi : \mathbb{R}^d \to \mathbb{R}^D$

$x \mapsto \varphi(x)$

Aggregation: $\psi : (\mathbb{R}^D)^* \to \mathbb{R}^D$

$\mathcal{X} \mapsto \psi(\mathcal{X})$

$$\mathcal{K}(\mathcal{X},\mathcal{Y}) \propto \left\langle \sum_{x\in\mathcal{X}} \varphi(x) \middle| \sum_{y\in\mathcal{Y}} \varphi(y) \right\rangle$$

$$\propto \sum_{x\in\mathcal{X}} \sum_{y\in\mathcal{Y}} \varphi(x)^\top \varphi(y)$$

Vector representation of $\mathcal{X}$

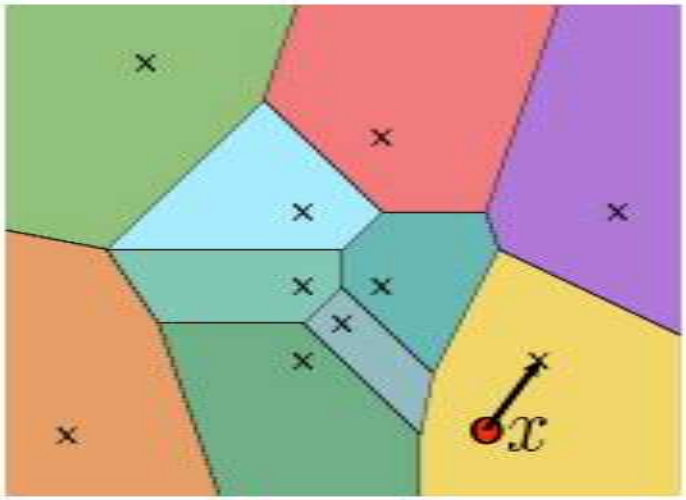
Voting interpretation

Inria

Match kernels: bag-of-words

Bag-of-words represents an image by an histogram of quantized SIFT descriptors

$$\varphi_{\text{bow}} : x \mapsto [0, \ldots, 0, 1, 0, \ldots]$$

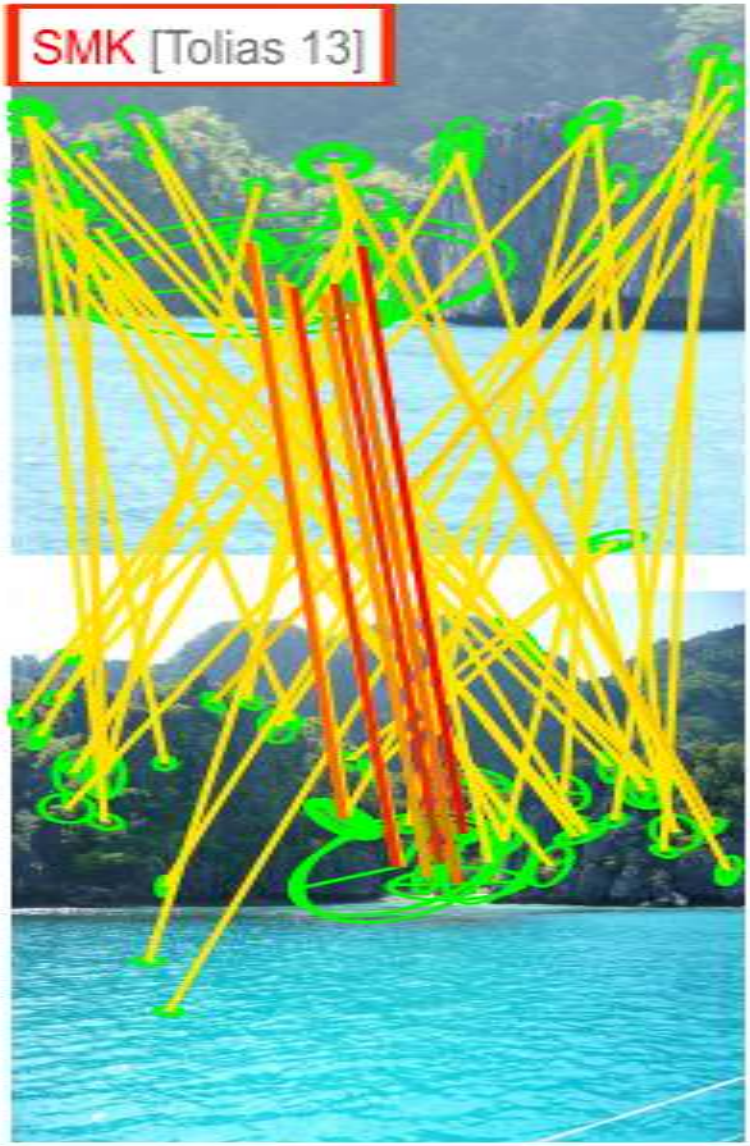


$$\psi_{\text{bow}}(\mathcal{X}) \propto \sum_{x \in \mathcal{X}} \varphi_{\text{bow}}(x)$$

Bag-of-words can also be seen as a voting method (a poor one)

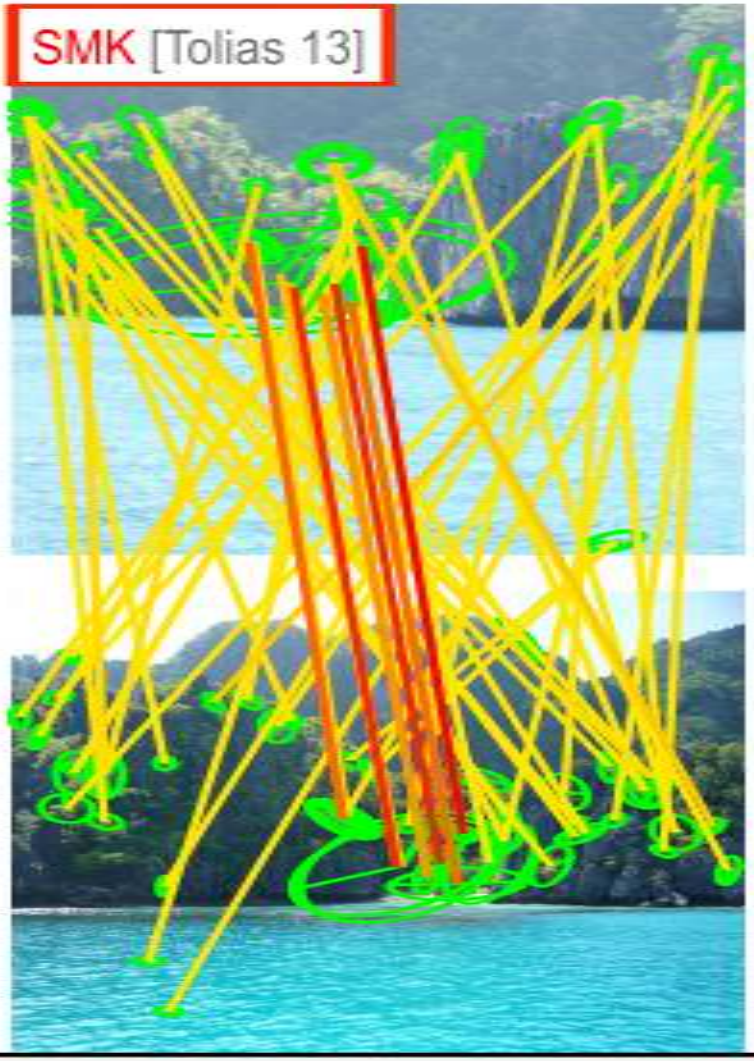
Inria

Selective match kernels



Inria

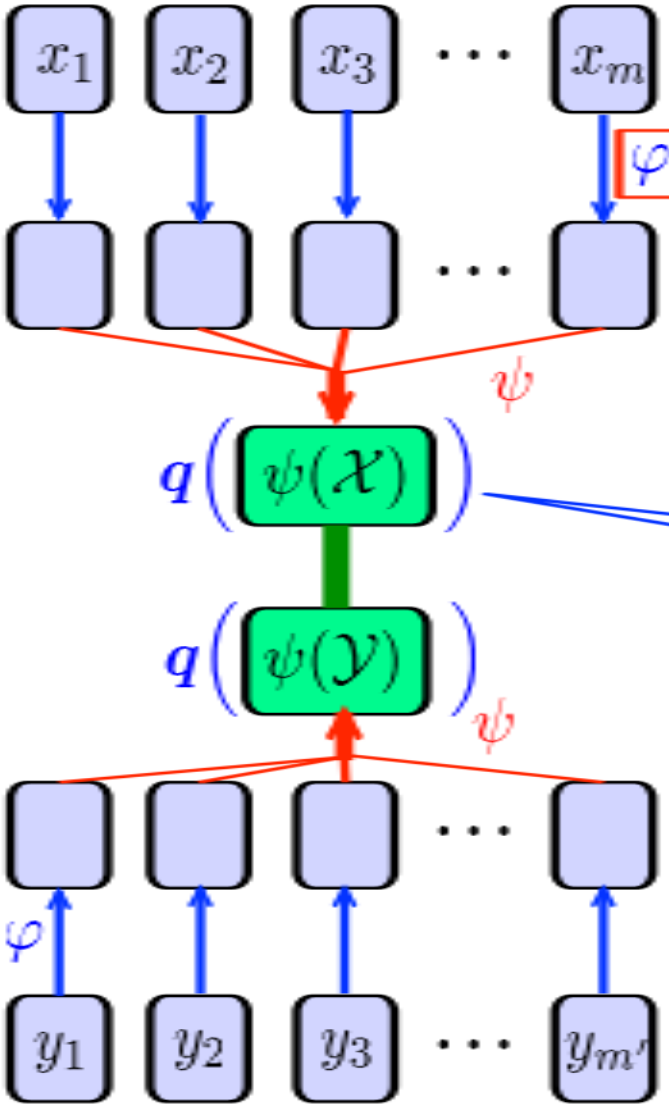
Selective match kernels



Good for large scale (millions of images) but memory demanding

Inria

Towards larger scale ?



Bag-of-words: limitation due to early quantization in the embedding step

More aggressive operating points?
Idea of VLAD-based method [CVPR 2010]:
Reversible embedding φ

Only the final representation is compressed (PCA and quantization)

Final image representation: **VLAD**
dozens to hundreds bytes per image

Inria

Very scalable system: VLAD match kernel

Joint work with Matthijs Douze, Patrick Pérez and Cordelia Schmid

- 10 million images on my laptop (CVPR'10)
  - 20 bytes per image
  - Search time: 20 ms (after description)

Inria

Scalability ≈ memory per image

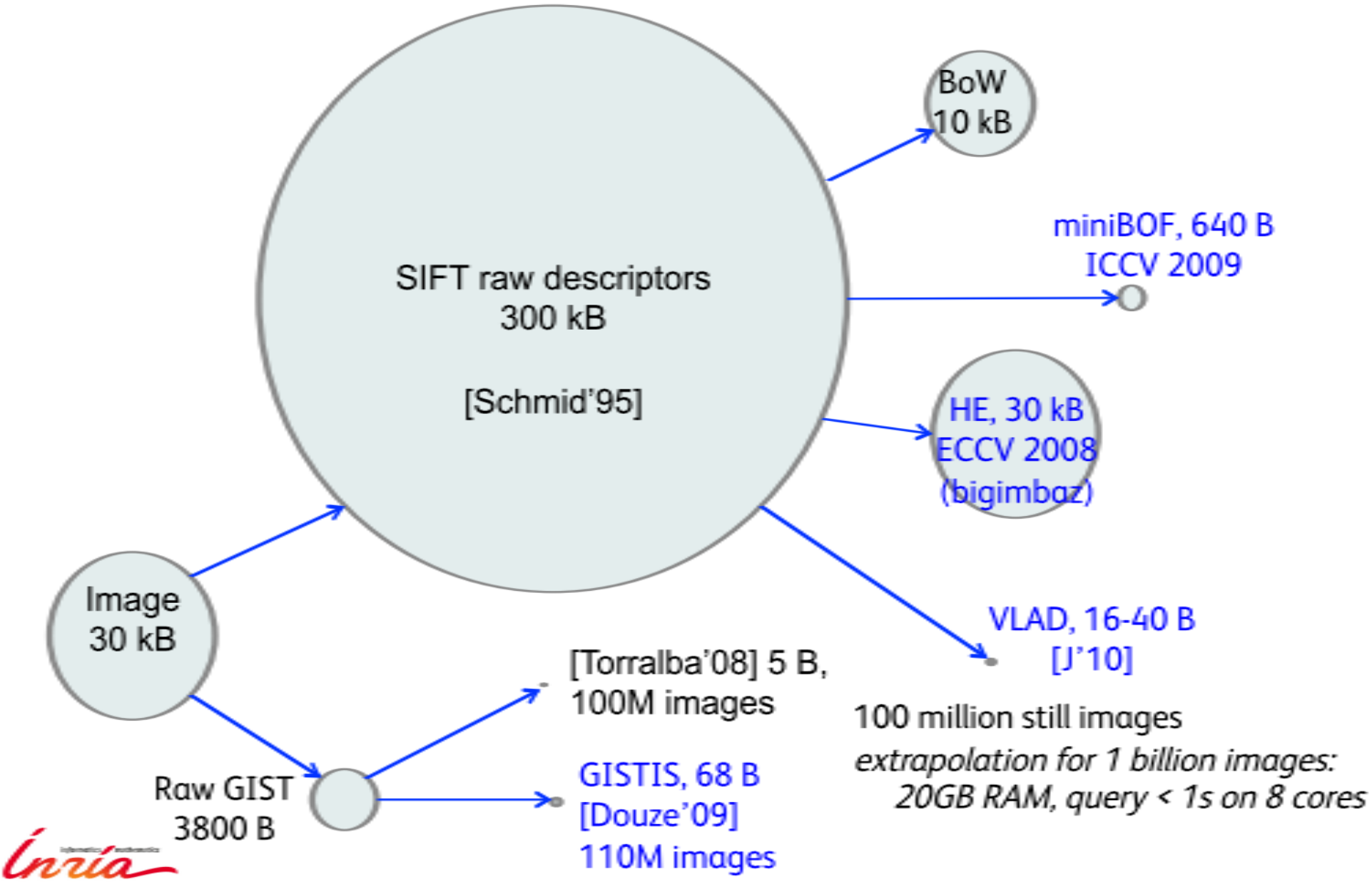
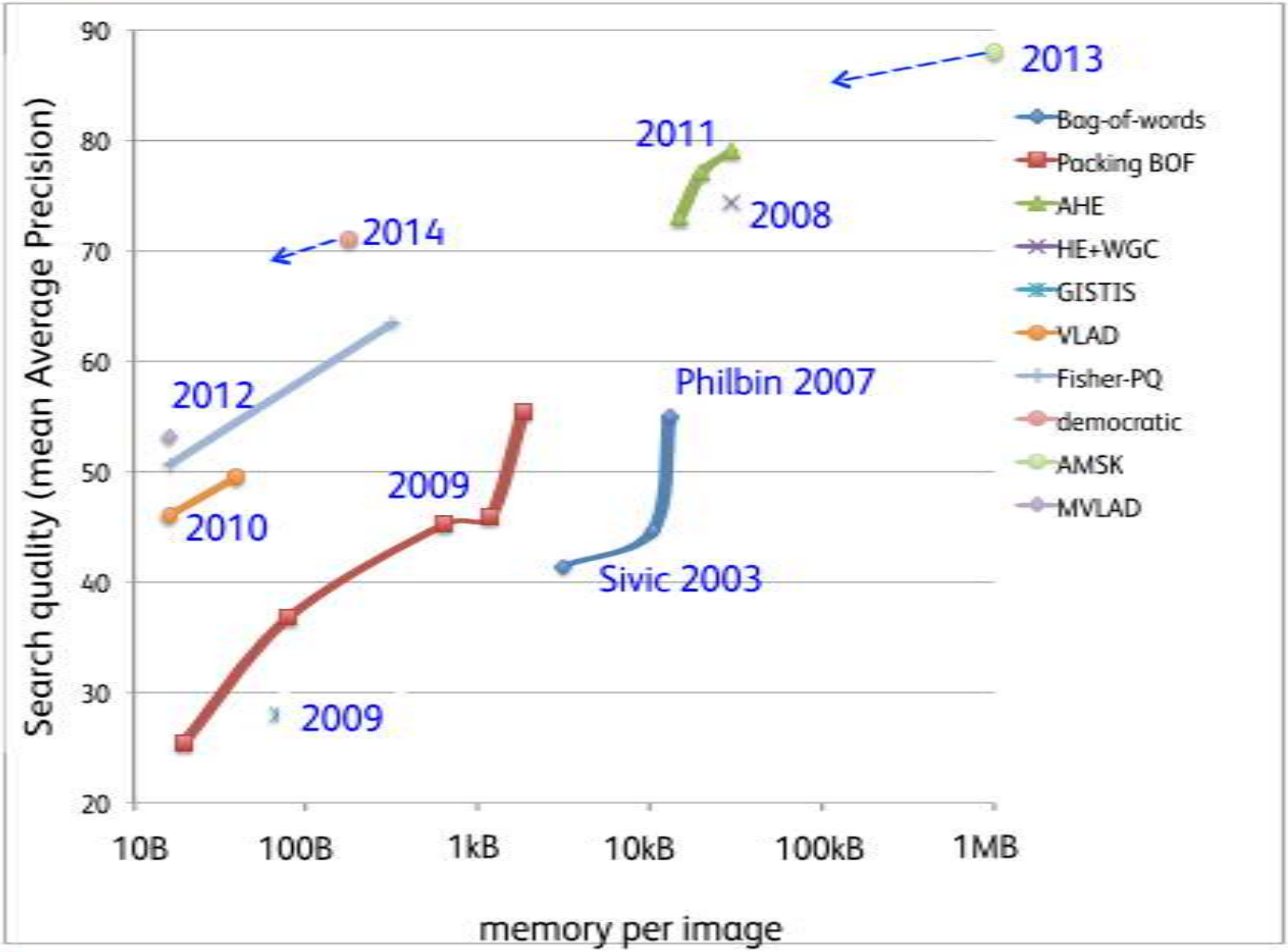


Image search: all about trade-offs



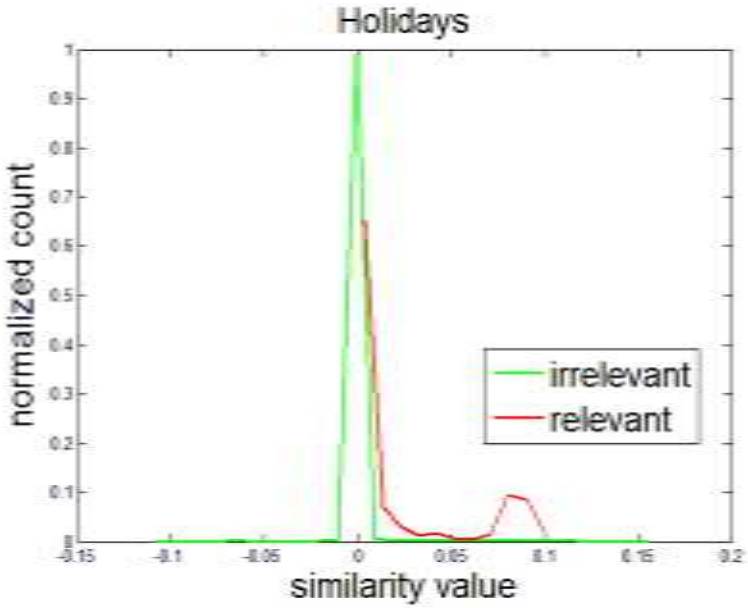
Inria

A recent work

- Recent image descriptors
  - Aggregated representations (like Fisher)
  - Deep learning features
  - better scalability: 100 million to 1 billion images on 1 server

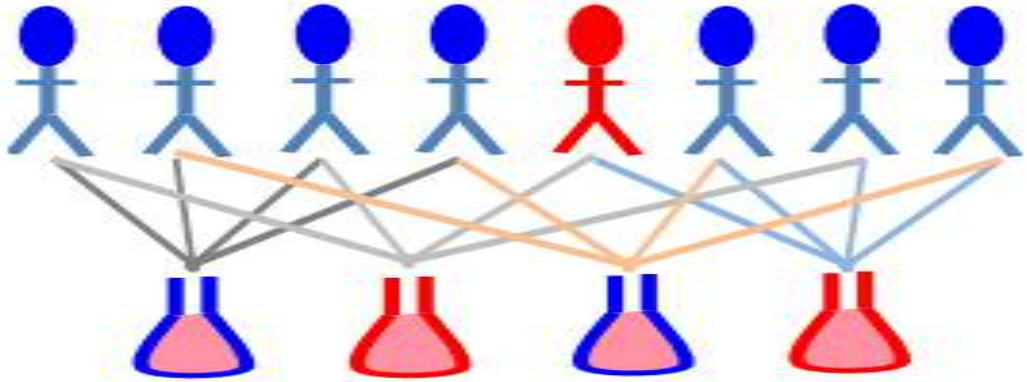
- Properties:
  - **Normalized, dense vectors**
  - **High-dimensional**: typically D=100 to 100,000
  - Compared with inner product

- How to exploit these properties?



World War II

Facing an increasing number of injuries, the US army decided to make blood donation mandatory for soldier recruits. However, a fraction of them were infected by syphilis. Unfortunately, syphilis testing back then was expensive.



This question gave birth to **group testing**, formalized by Dorfman in 1943

The principle: There is no point running an exhaustive testing procedure over a large population if a small fraction of individuals are infected. Blood can be pooled into **one mixture** and tested against syphilis. If the test is positive, the blood samples are individually tested.

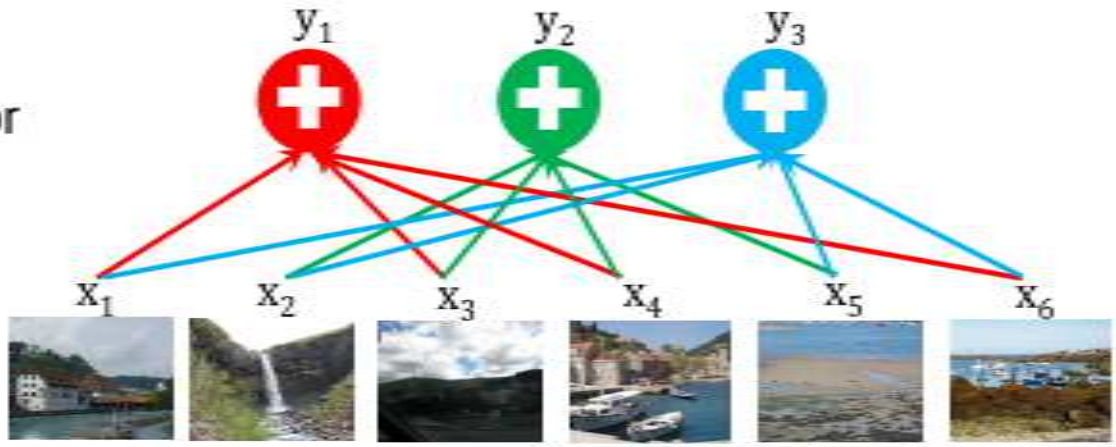
Group testing cast to image search

Joint work with Miaojing Shi and Teddy Furon

- Grouping operator: **+**

- produce the group vector

$$\mathbf{y}_i = \sum_{j \in \mathcal{H}_i} \mathbf{x}_j$$



- In matrix form:

$$\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_m] = \mathbf{X} \cdot \mathbf{G}$$

- $\mathbf{G}$ is the (sparse) mixing matrix

Inria

Group testing cast to image search

- **A group test**: we compare the query against the group vector

$$v_i = \langle \mathbf{y}_j | \mathbf{q} \rangle$$

- All tests: a matrix-vector multiplication

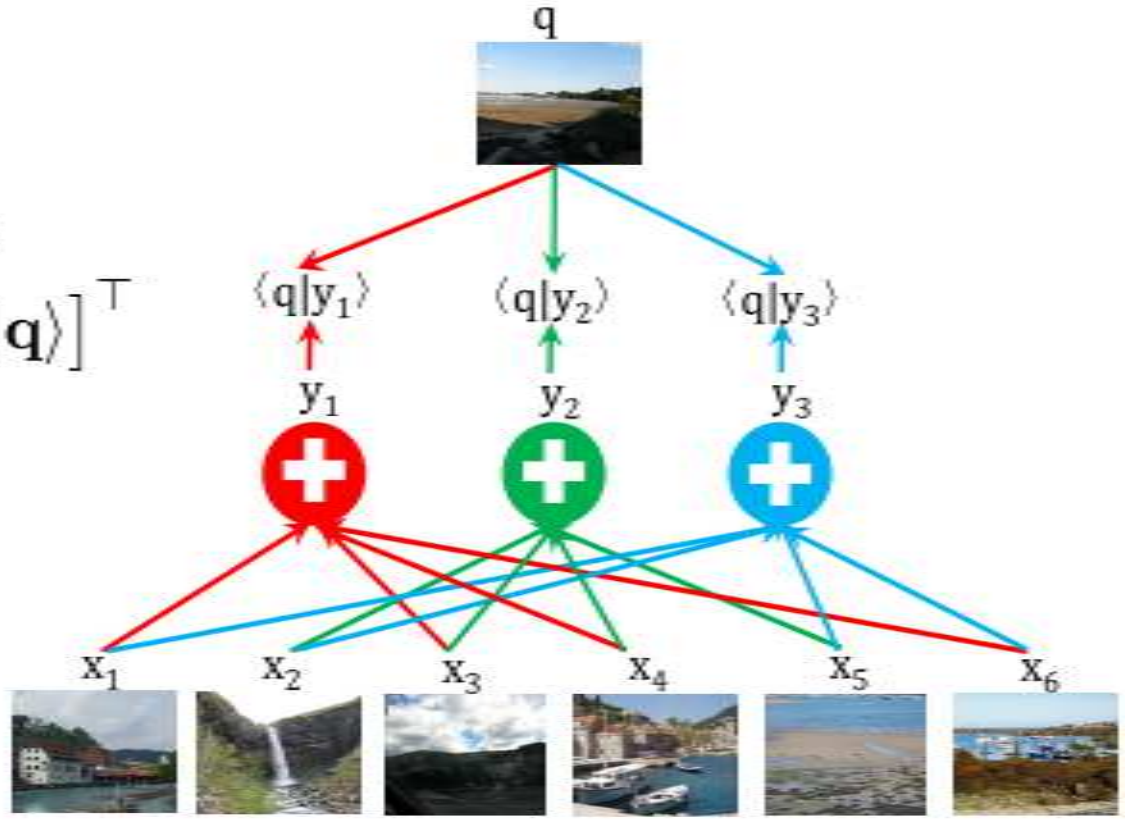
$$\mathbf{v} = \mathbf{Y}^\top \mathbf{q}$$

- Denoting the image scores

$$\mathbf{u} = \left[\langle \mathbf{x}_1 | \mathbf{q} \rangle, \ldots, \langle \mathbf{x}_n | \mathbf{q} \rangle \right]^\top$$

we have $\mathbf{v} = \mathbf{G} \cdot \mathbf{X}^\top \mathbf{q}$

$$= \mathbf{G}\mathbf{u}$$



In summary: an inverse problem

- We measure $\mathbf{v} = \mathbf{Y}^\top \mathbf{q}$
- We want to estimate $\mathbf{u} = \mathbf{X}^\top \mathbf{q}$
- knowing that $\mathbf{v} = \mathbf{G}\mathbf{u}$
- This is an undetermined system
  - we have less groups than images: $m < n$

First solution: pseudo-inverse

- The usual solution: pseudo-inverse

$$\tilde{\mathbf{u}} = (\mathbf{G}^\top \mathbf{G})^{-1} \mathbf{G}^\top \mathbf{v}$$

- **Drawbacks:**
  - Pseudo-inverse is not sparse (computationally costly in this case)
  - Large variance of the estimator = bad quality

- Most importantly: this estimator does not use the key assumption that we have high-dimensional vectors
  - Unrelated images: $u_j \approx 0$
  - Related images: $u_j \gg 0$

- In other terms, most of the components of $\mathbf{u}$ are close to 0

Inria

Second solution: sparse decoding

- Recall that pseudo-inverse solves

$$\tilde{\mathbf{u}} = \arg\min_{\mathbf{u}} \frac{1}{2}\|\mathbf{v} - \mathbf{G}\mathbf{u}\|^2 + \frac{\lambda}{2}\|\mathbf{u}\|_2^2$$

- Assuming that the vector has many components equal to zero, we can solve the problem

$$\tilde{\mathbf{u}} = \arg\min_{\mathbf{u}} \|\mathbf{v} - \mathbf{G}\mathbf{u}\|^2 + \lambda\|\mathbf{u}\|_1$$

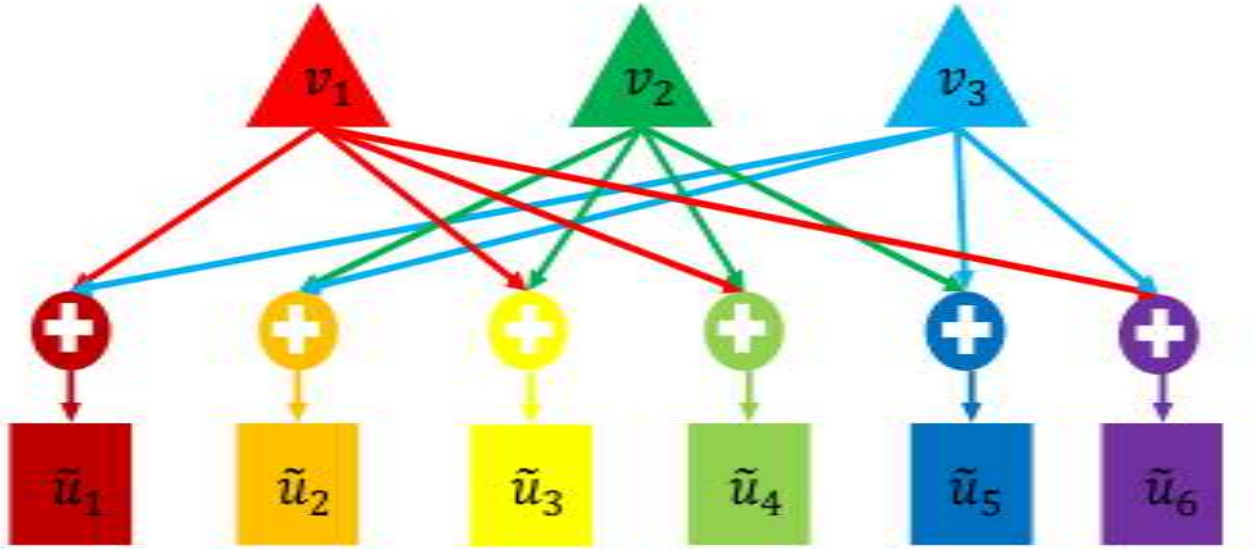
which produces a sparse vector

- The non-zeros are assumed to be image matches

→ we can add a positivity constraint: $\forall j,\ \tilde{u}_j \geq 0$

- **Problems**: slow, assumptions of 0s not satisfied

Group testing

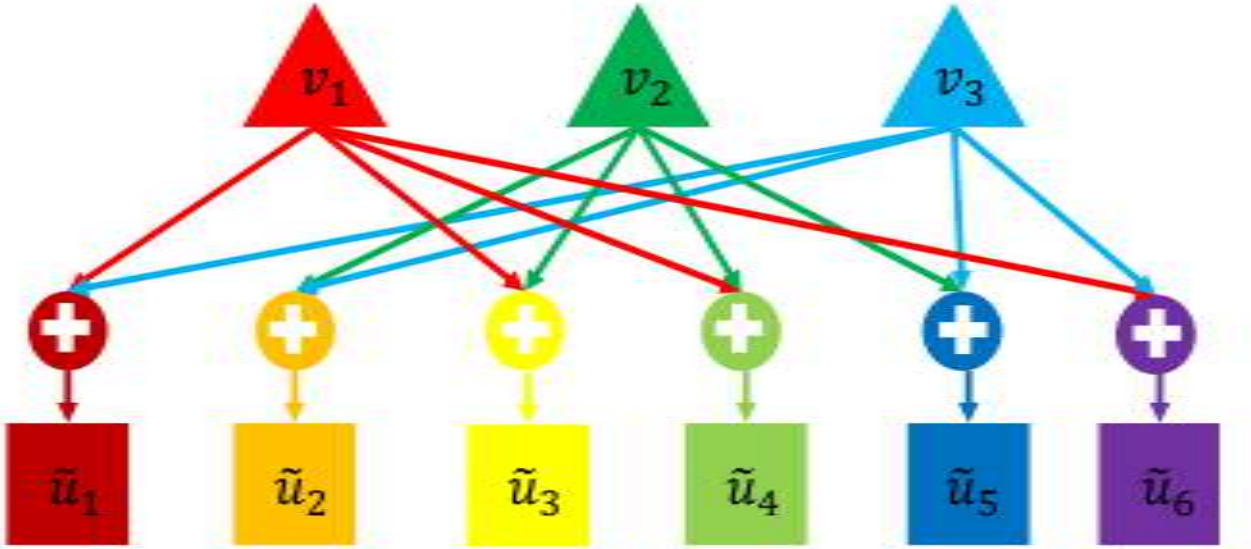


- Simplification of the decoder: $\tilde{\mathbf{u}} = \mathbf{G}^\top \mathbf{v}$

= just add the similarities of the groups to which the image belongs to

Inria

Adaptive group testing



- We find the largest estimated score → potential match
  - and then perform a direct measurement $u_j = \langle \mathbf{x}_j | \mathbf{q} \rangle$

- Thanks to linearity, we can now perform a ``**back-propagation**'' step
  - remove the contribution of the known measurement

- The procedure is iterated R times (possibly by batch)

Inria

Conclusion

More than 20 years of image indexing

- From global to local description
  - SIFT descriptors
- Indexing high-dimensional vectors: very important
  - Memory is critical for scalability
  - A distance estimation ≡ compression problem
- Group measurements
  - Back to a single vector per image with match kernels
  - Similarity estimation with group testing
- The field still evolves rapidly: deep learning features
  - Already state-of-the-art in image classification

Inria

erc