CS688: Web-Scale Image Retrieval

Bag-of-Words (BoW) Models for Local Descriptors

Sung-Eui Yoon
(윤성의)

Course URL:
http://sglab.kaist.ac.kr/~sungeui/IR

KAIST

Tentative Schedule

- 10/25: mid-term exam

10/27: Student pre. I

11/1

11/3

11/8

11/10: mid-pre

11/15: mid-pre

11/17: no class

11/22: Student pre. II

11/24

11/29

12/1

12/6

12/8: reserved

12/13: final pre.

12/15: final pre.

D: 10/4 (student schedule

KAIST

Deadlines

- **Declare project team members**
  - **By 10/3 at Noah**
- **Confirm schedules of paper talks and project talks at 10/4**
- **Declare two papers for student presentations**
  - **by 10/17 at Noah**
  - **Discuss them at the class of 10/18**

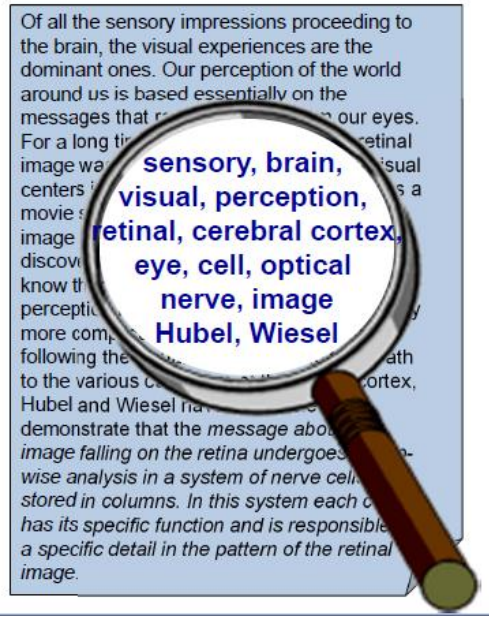
Class Objectives

- **Bag-of-visual-Word (BoW) model**
- **Understand approximate nearest neighbor search**
  - **Inverted index**
  - **Inverted multi-index**

Object → **Bag of 'words'**

Represent an image with a histogram of words

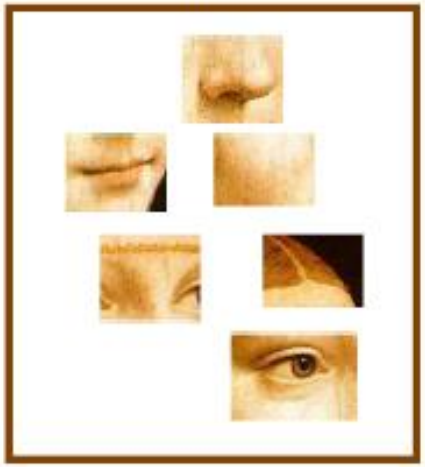
Inspired by text search



definition of “BoW”

– Independent features

face



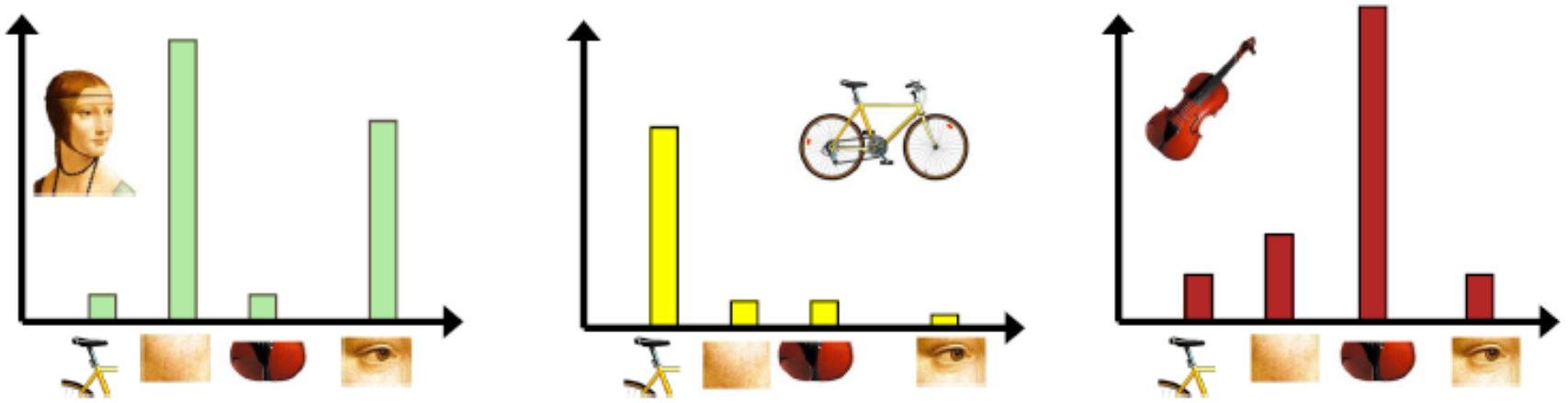
bike

violin

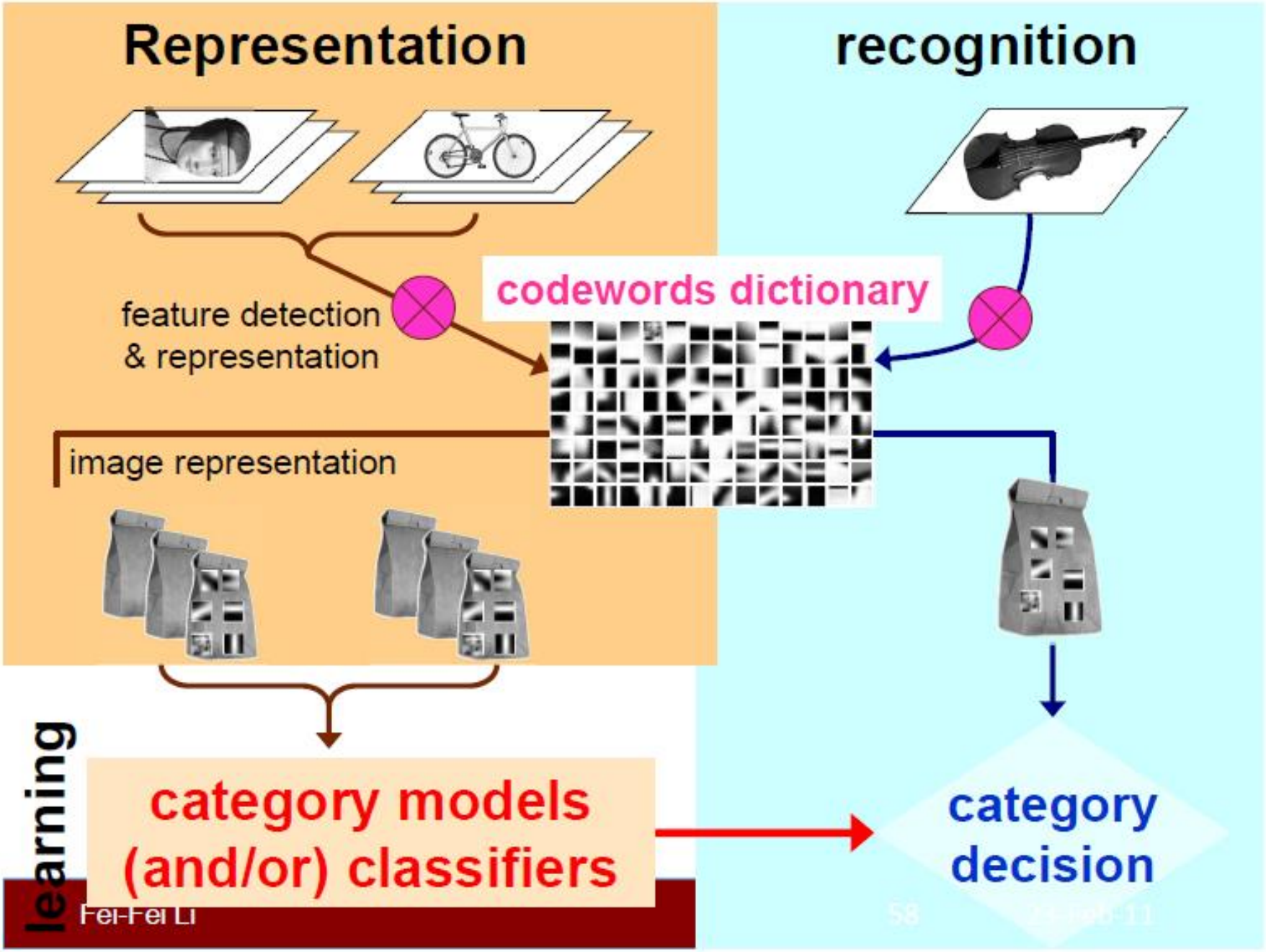
definition of “BoW”

– Independent features

– histogram representation

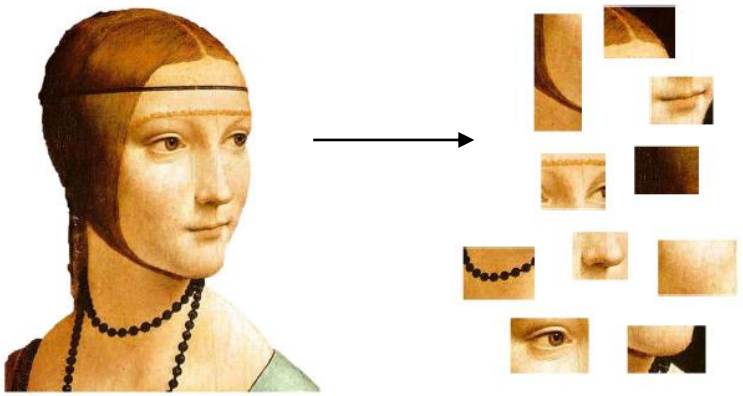


codewords dictionary

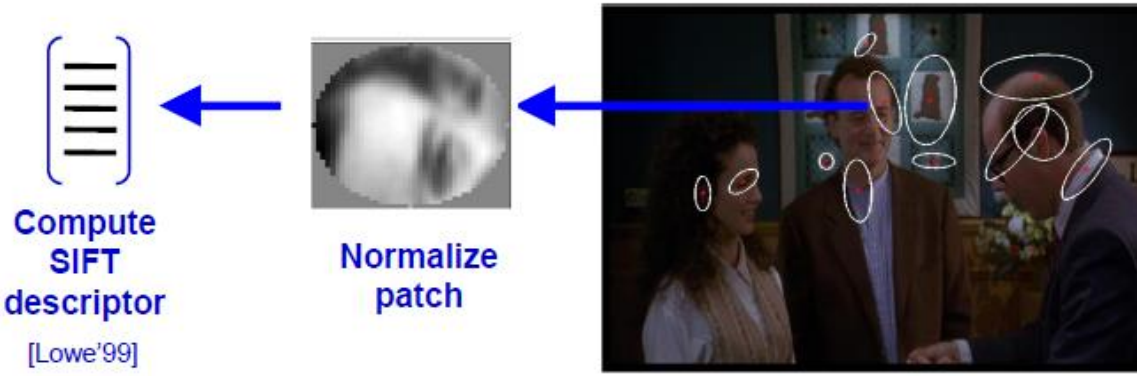

Representation
recognition
feature detection
& representation
codewords dictionary
image representation
learning
category models
(and/or) classifiers
category
decision

1. Feature Detection and Representations

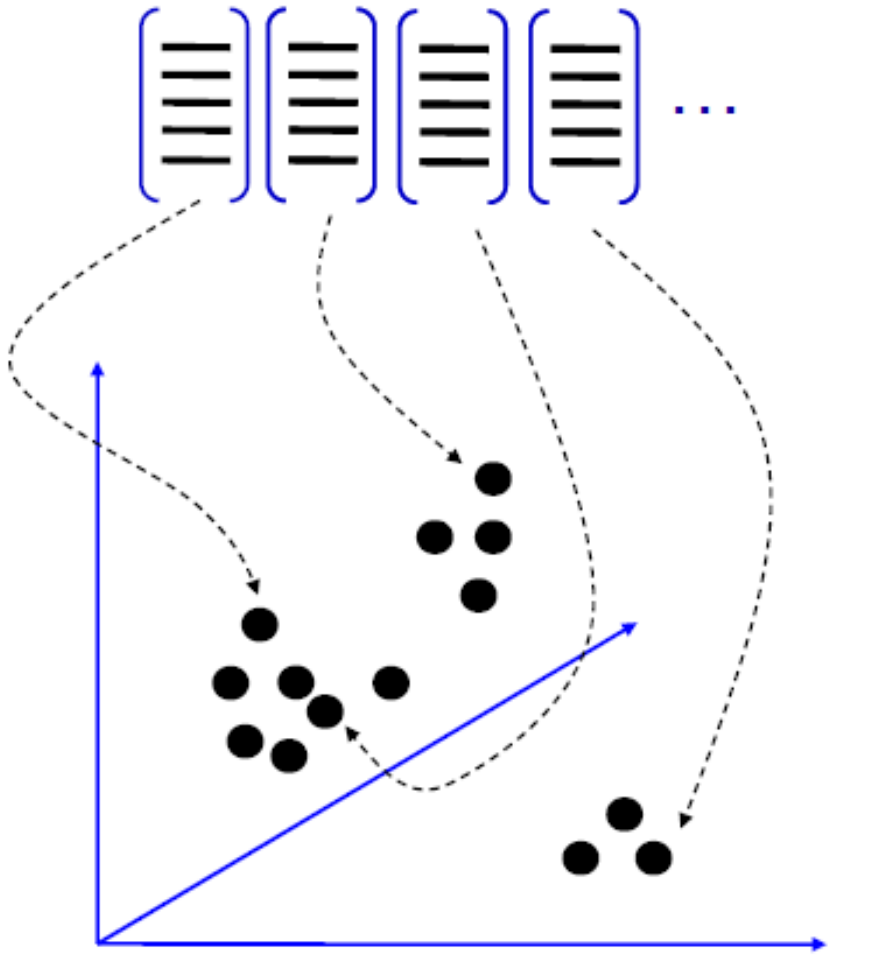
- **Assume many local features as an aggregation model**
  - **Global feature is not used**
- **Densely sampled or sampled only at key points**
  - **Detect patches extract features from them**



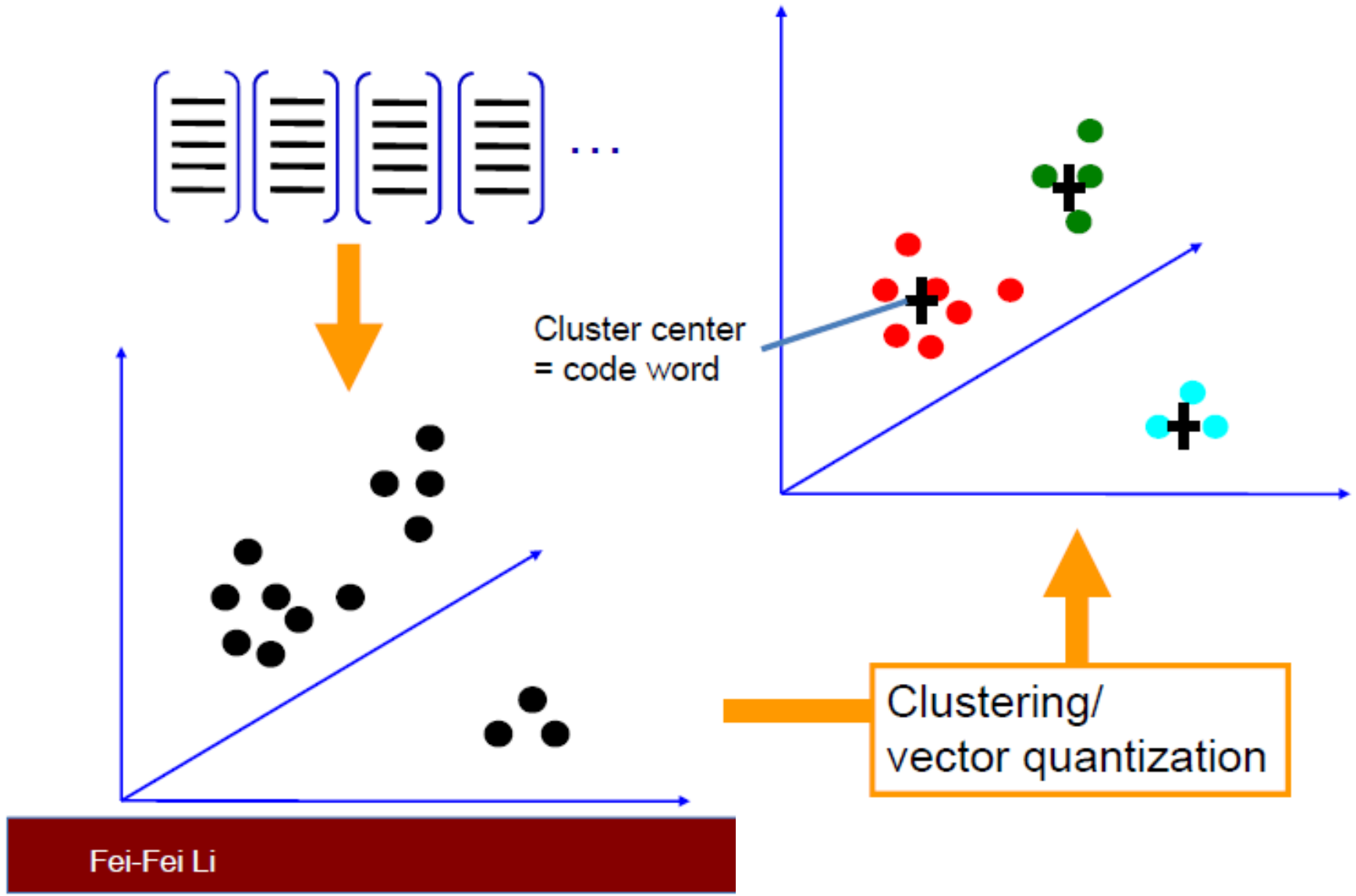
Ack.: Josef Sivic and Li Fei-Fei



2. Codewords dictionary formation



2. Codewords dictionary formation

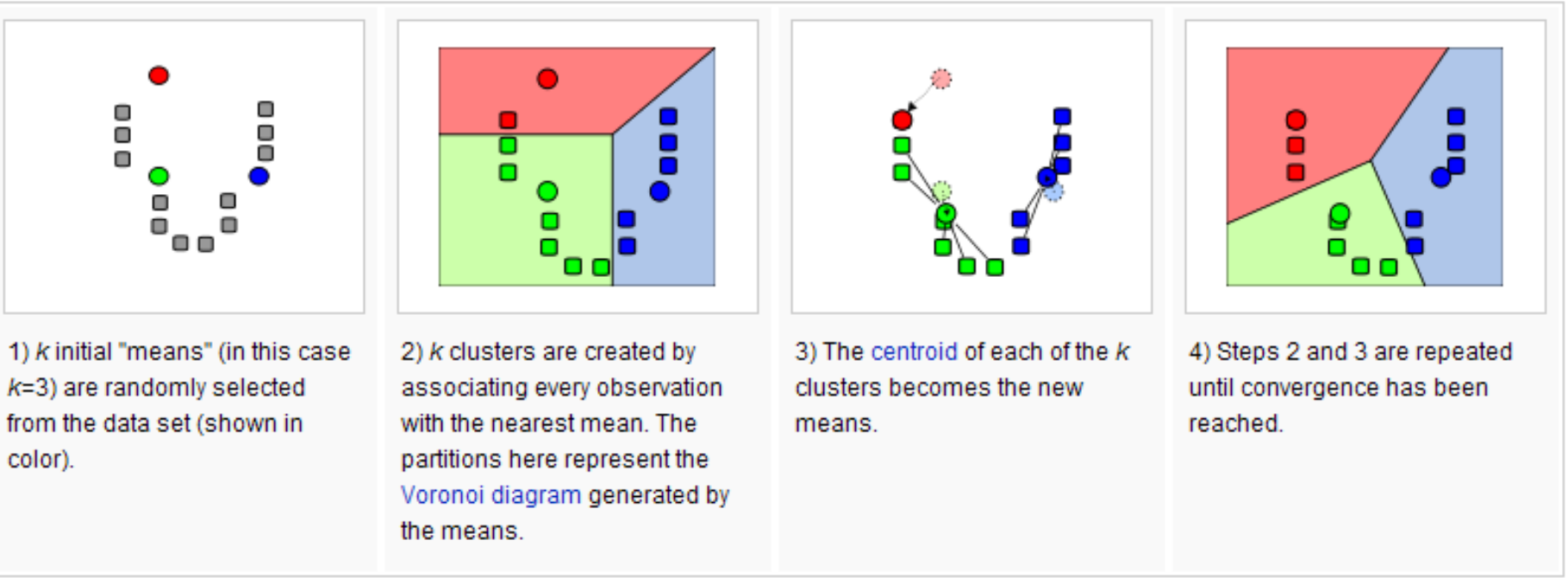


K-Means Clustering

- **An unsupervised learning**
- **Minimize the within-cluster sum of squares**

$$\underset{\mathbf{S}}{\arg\min} \sum_{i=1}^{k} \sum_{\mathbf{x}_j \in S_i} \left\| \mathbf{x}_j - \boldsymbol{\mu}_i \right\|^2$$

Demonstration of the standard algorithm



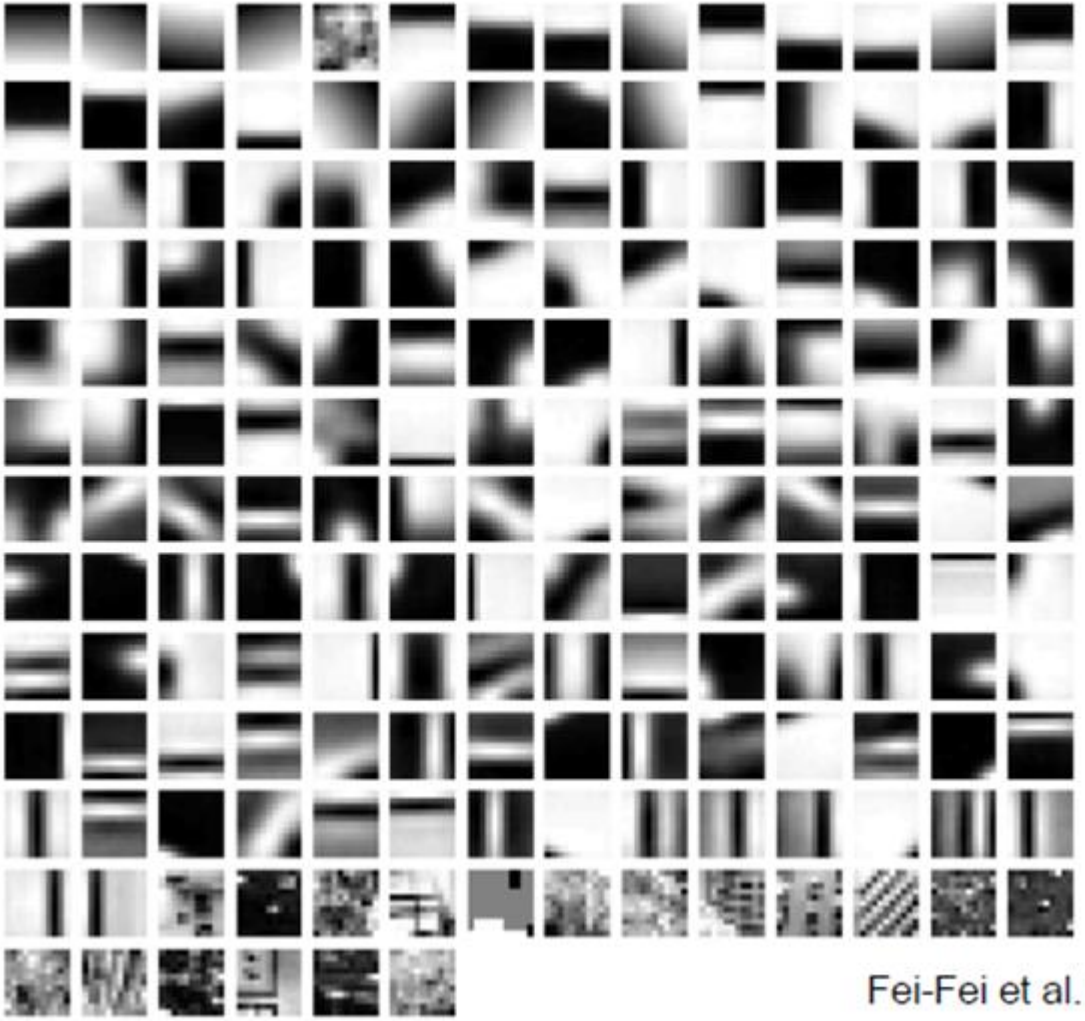
1) *k* initial "means" (in this case *k*=3) are randomly selected from the data set (shown in color).

2) *k* clusters are created by associating every observation with the nearest mean. The partitions here represent the Voronoi diagram generated by the means.

3) The centroid of each of the *k* clusters becomes the new means.

4) Steps 2 and 3 are repeated until convergence has been reached.

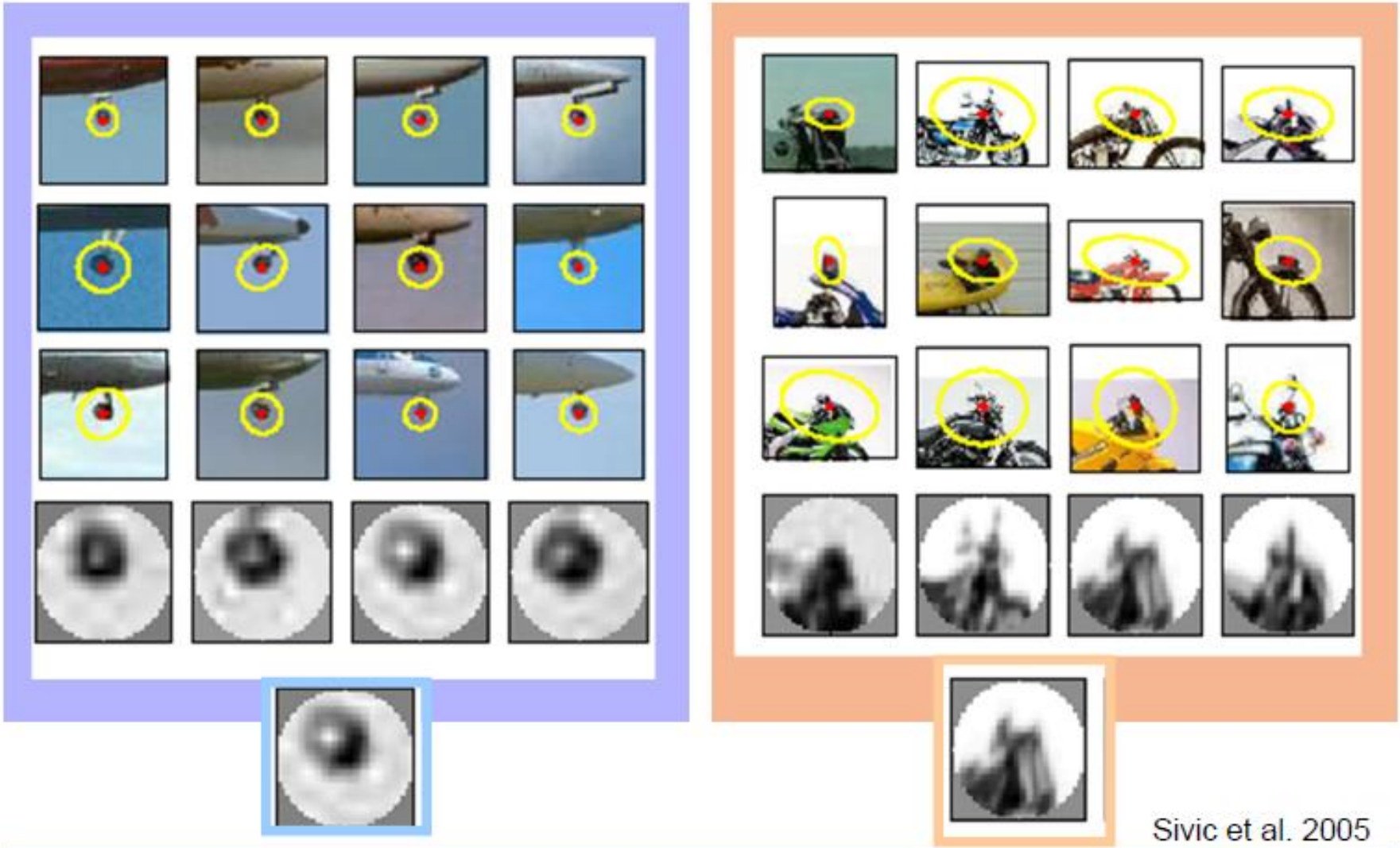
Codewords Dictionary Formation



Fei-Fei et al. 2005

KAIST

Image patch examples of codewords

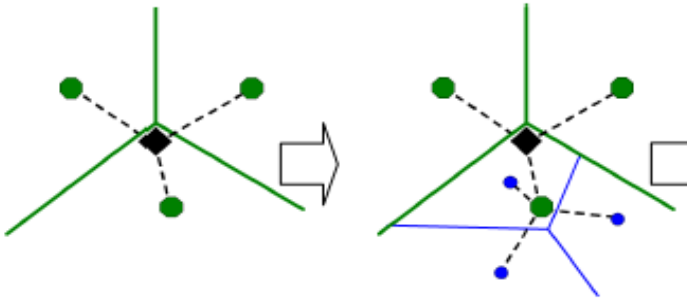


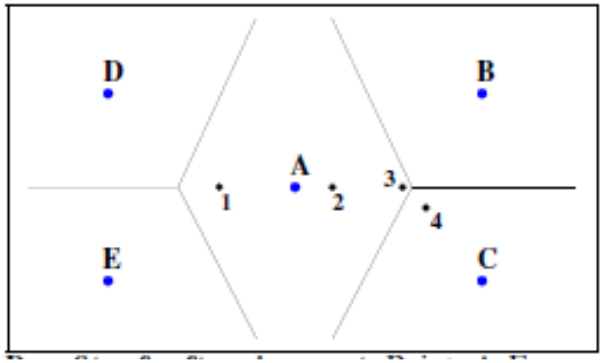
Sivic et al. 2005

Issues of Visual Vocabulary

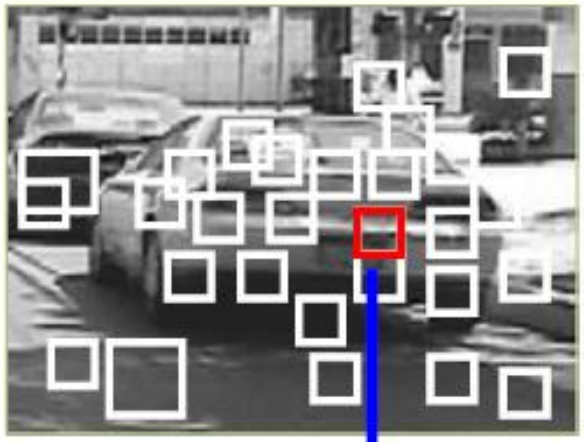
- **Related to quantization**
  - **Too many words: quantization artifacts**
  - **Too small words: not representative**
- **K-means also takes long computation times**

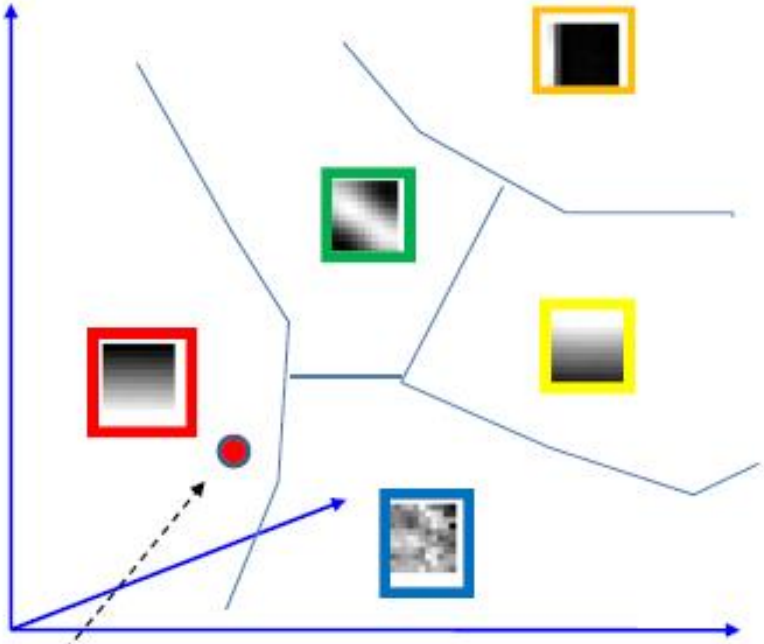
- **Alternatives**
  - **Faster performance: vocabulary tree, Nister et al.**
  - **Low quantization artifacts: soft quantization, Philbin et al.**

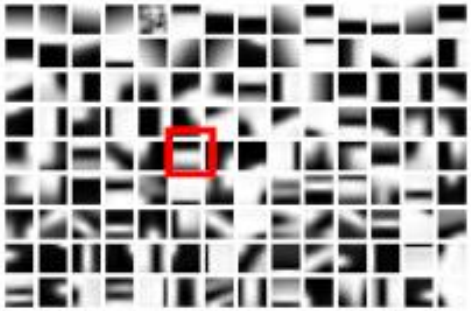




3. Bag of word representation



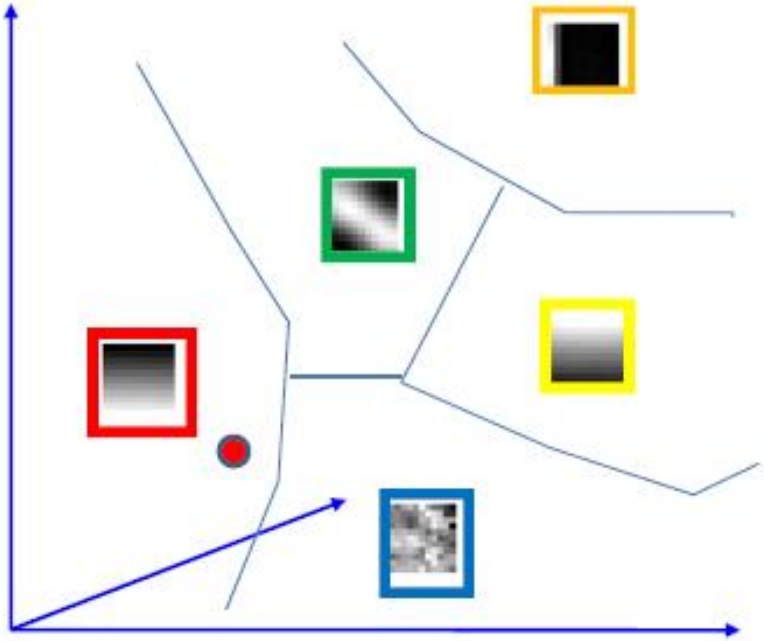


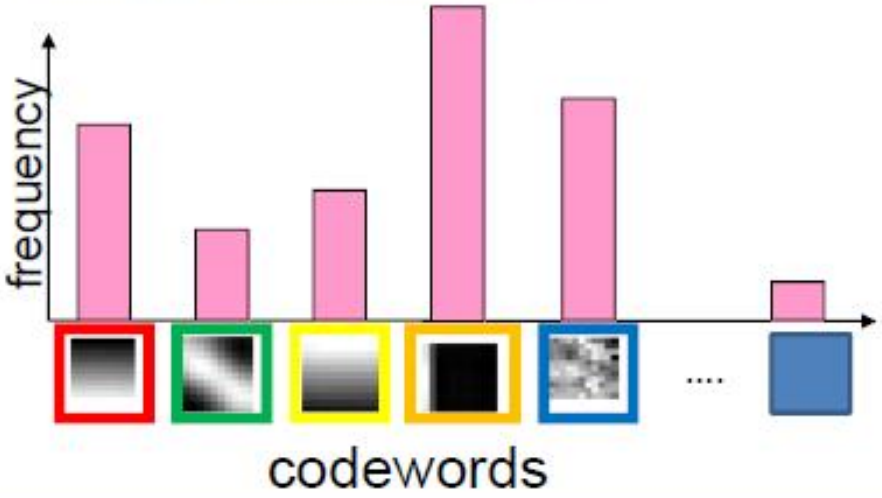


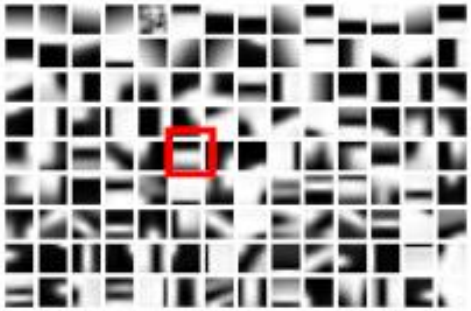
Codewords dictionary

- Nearest neighbors assignment
- K-D tree search strategy

3. Bag of word representation







Codewords dictionary

Representation

2.

codewords dictionary

1. feature detection & representation

image representation

3.

Learning and Recognition

codewords dictionary

category models (and/or) classifiers

category decision

TF-IDF

- **Adopted from text search**
  - **A kind of weighting and normalization process**
- **Assume a document to be represented by** $(t_1, \ldots, t_i, \ldots, t_k)^\top$
- **Weighted by TF (Term frequency) * log (IDF (Inverse Document Frequency))**

$$t_i = \frac{n_{id}}{n_d} \log \frac{N}{n_i}$$

  - **n_{id} : # of occurrences of word i in document d**
  - **n_d : total # of words in the document d**
  - **n_i : # of occurrences of term i in the whole database**
  - **N: # of documents in the whole database**

Similarity and Distance Functions

- **Dot product measuring the angle between two vectors**
- **L1 or Euclidean distance**

$$L1\,(h_1, h_2) = \sum_i \left| h_1^i - h_2^i \right|$$

- χ^2 distance

$$D(h_1, h_2) = \sum_{i=1}^{N} \frac{\left(h_1(i) - h_2(i)\right)^2}{h_1(i) + h_2(i)}$$

- Quadratic distance (*cross-bin*)

$$D(h_1, h_2) = \sum_{i,j} A_{ij} \left(h_1(i) - h_2(j)\right)^2$$

Jan Puzicha, Yossi Rubner, Carlo Tomasi, Joachim M. Buhmann: Empirical Evaluation of Dissimilarity Measures for Color and Texture. ICCV 1999

PA2

- **Understand and implement a basic image retrieval system**
- **Use the original UKBenchmark**
- **Measure its accuracy**

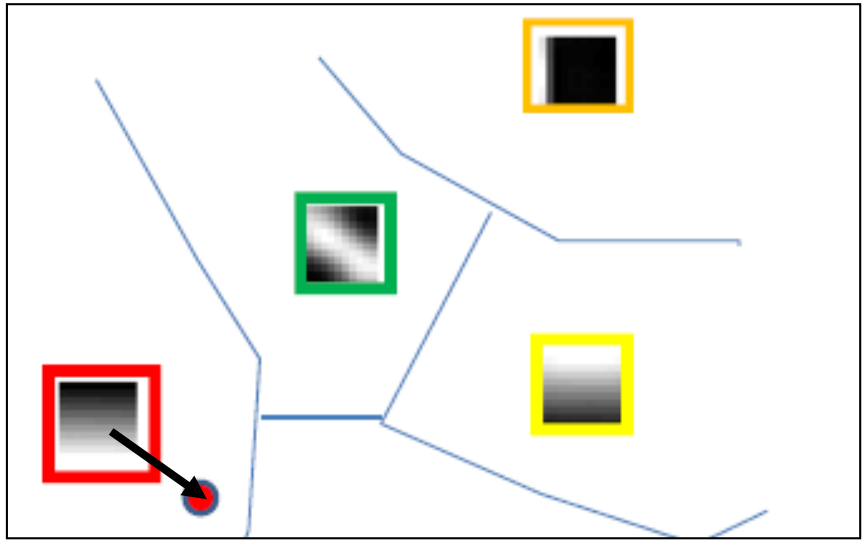
Query	First	Second	Third
			

KAIST

VLAD (Vector of Locally Aggregated Descriptors)

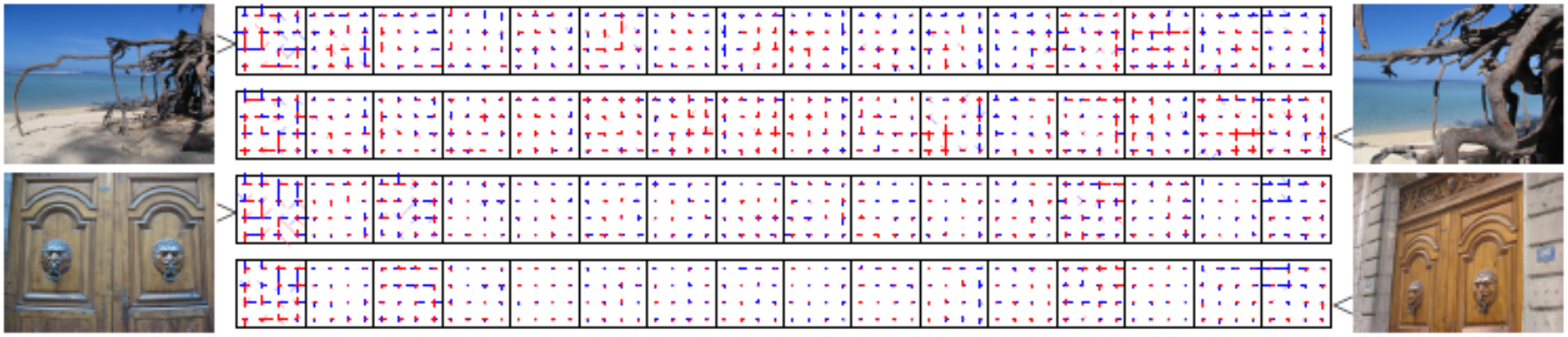
- **BoW**
  - **Count the number of SIFTs assigned to each cluster**
- **VLAD**
  - **Compute the difference between a SIFT and its cluster center**

$$v_{i,j} = \sum_{x \text{ such that } \mathrm{NN}(x)=c_i} x_j - c_{i,j}$$



KAIST

VLAD



- **VLAD descriptors w/ 16 clusters**
- **Show better accuracy than BoW**

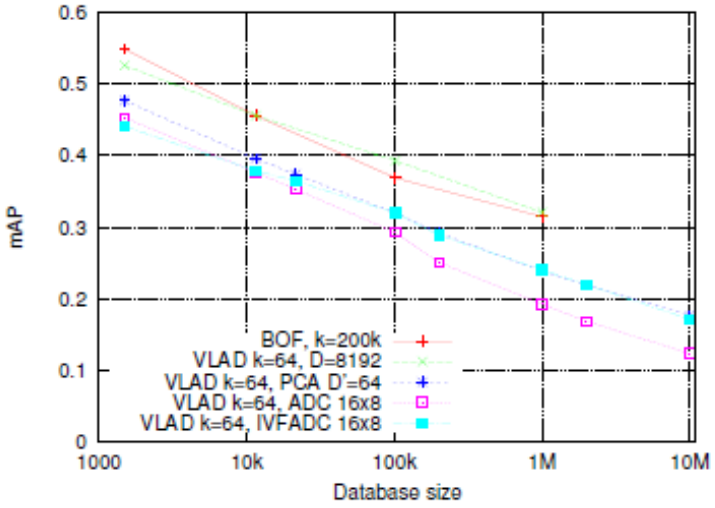


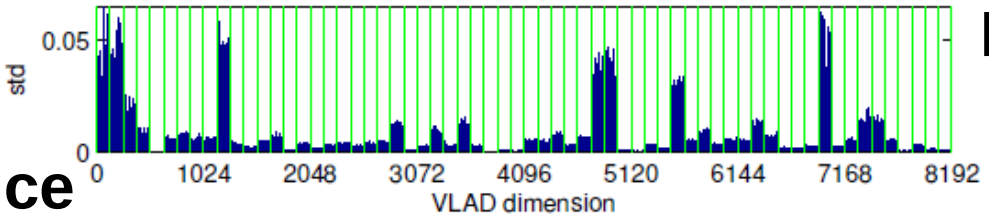
Figure 5. Search accuracy as a function of the database size.

Ack. Aggregating local descriptors ..

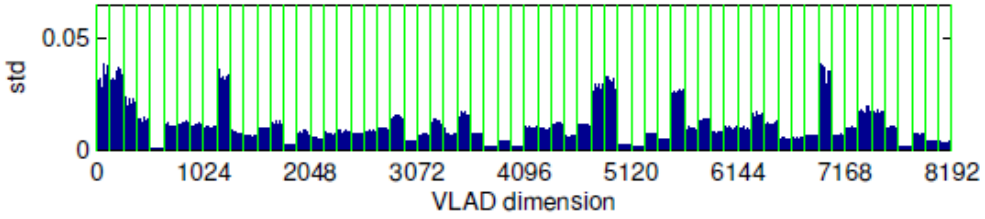
Normalization for VLAD

- **Results in better accuracy**

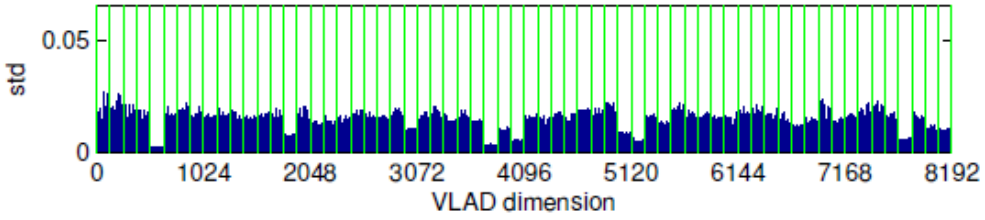
Variance



(a) Original VLAD normalization (L2)



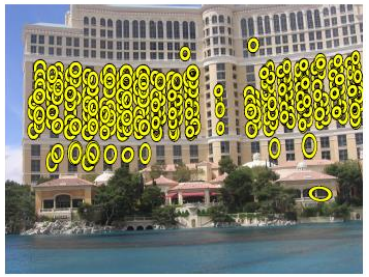
(b) Signed square rooting (SSR) followed by L2



(c) Intra-normalization (innorm) followed by L2

L2 normalization, i.e., $\frac{v}{|v|^2}$

Square rooting for burstiness

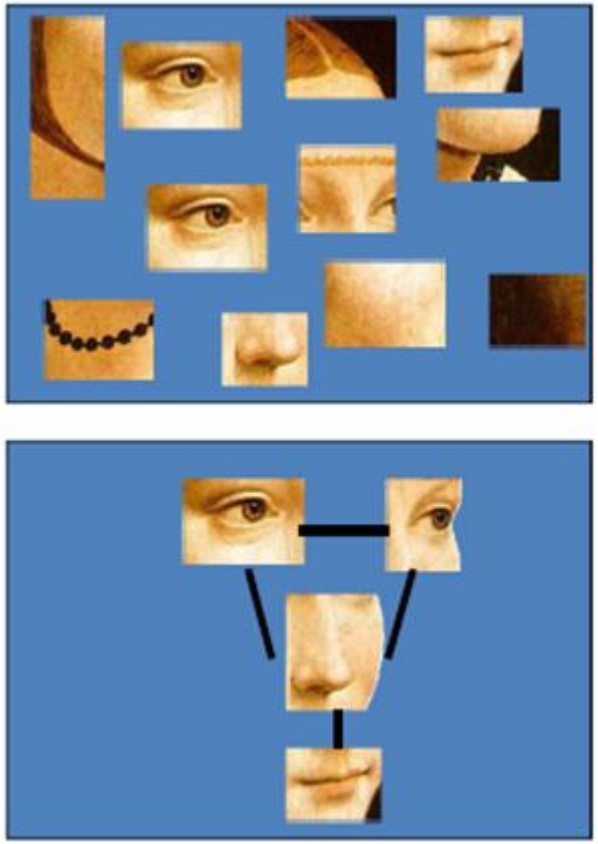


L2 normalization within each VLAD block

Ack. All about VLAD

Problems of BoW Model

- **No spatial relationship between words**
- **How can we perform segmentation and localization?**

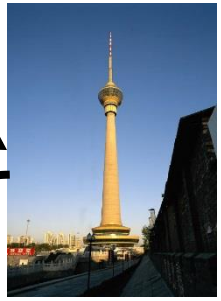


Ack.: Fei-Fei Li

Post-Processing or Reranking

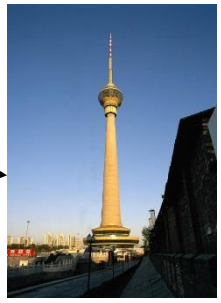
Query image

Shortlist (e.g., 100 images)



..

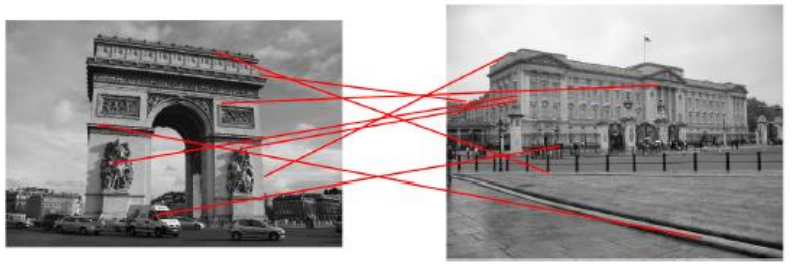
Re-ranking



..

Post-Processing

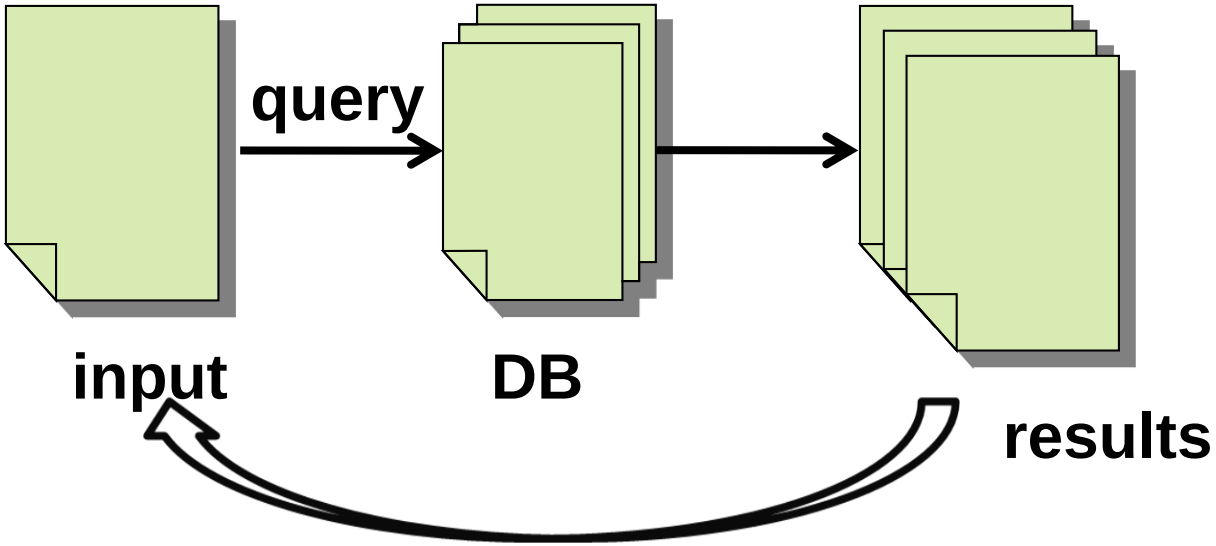
- **Geometric verification**
  - **RANSAC**



Matching w/o spatial matching

(Ack: Edward Johns et al.)

- **Query expansion**



KAIST

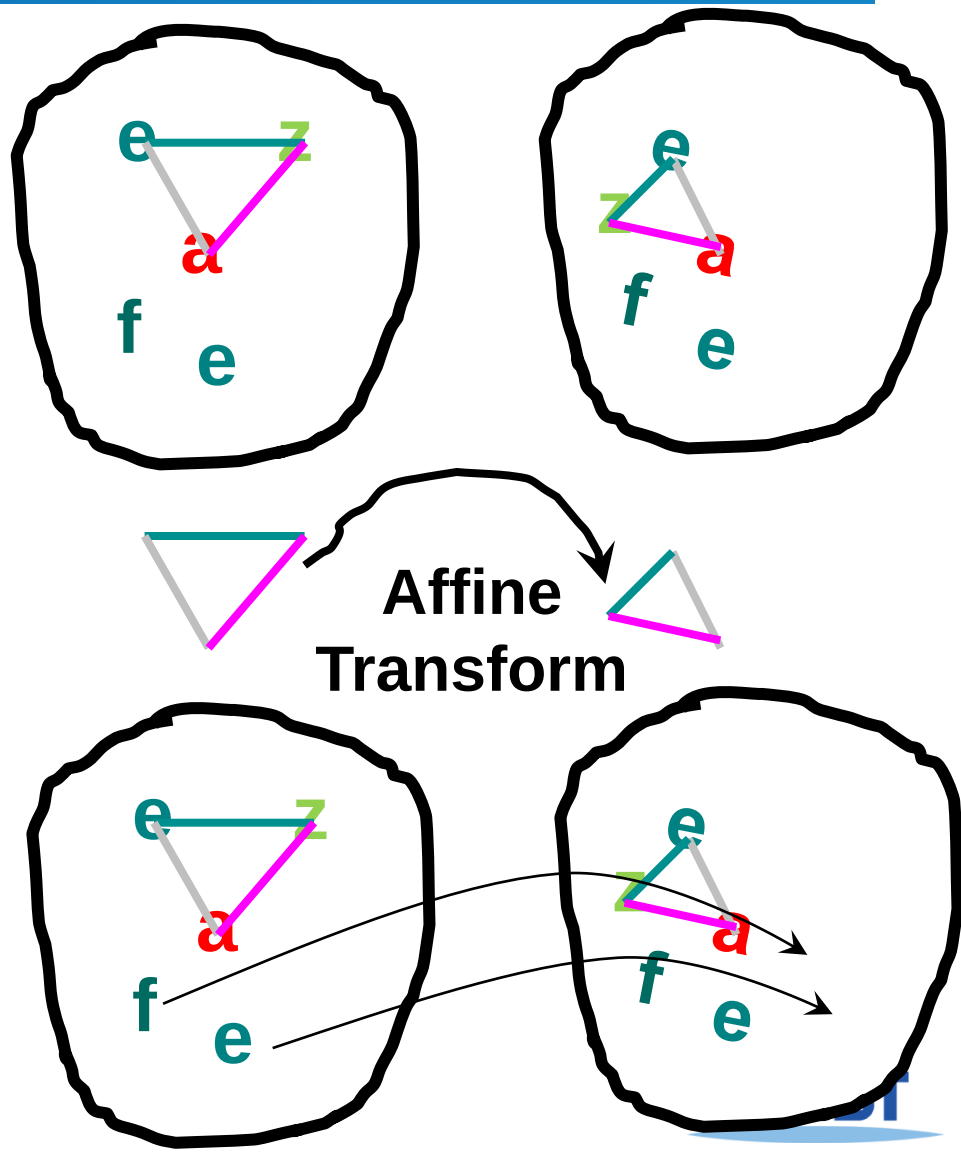
Geometric Verification using RANSAC for Affine Transform

Repeat N times:

- Randomly choose 3 matching pairs

- Estimate transformation

- Predict remaining points and count "inliers"



Ack.: Derek Hoiem (UIUC)

Query Expansion [Chum et al. 07]

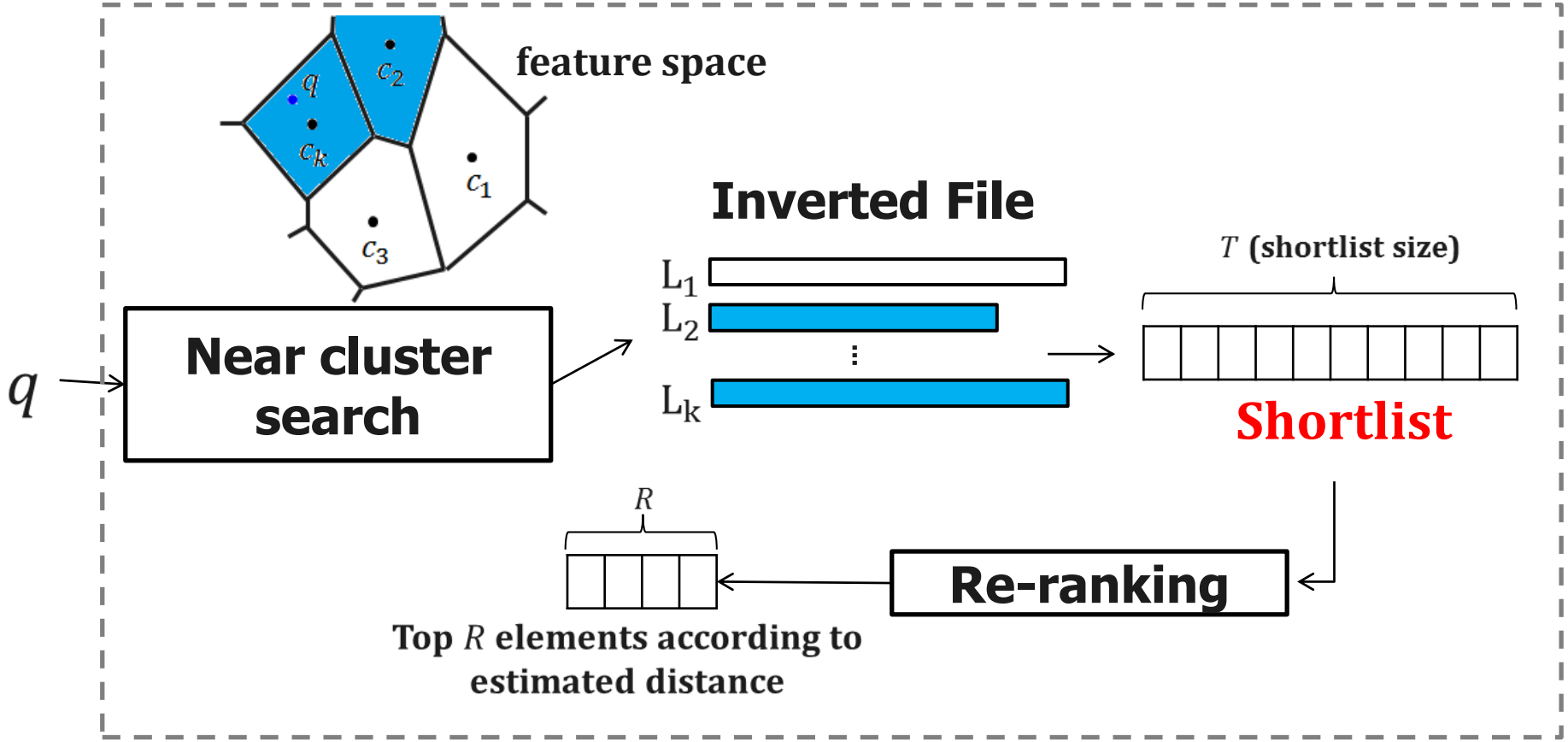
Original query

Top 4 images

Expanded results that were not identified by the original query

Inverted File or Index for Efficient Search

- **For each word, list images containing the word**



Ack.: Dr. Heo

KAIST

Inverted Index

Construction time:

- Generate a codebook by quantization
  - e.g. k-means clustering

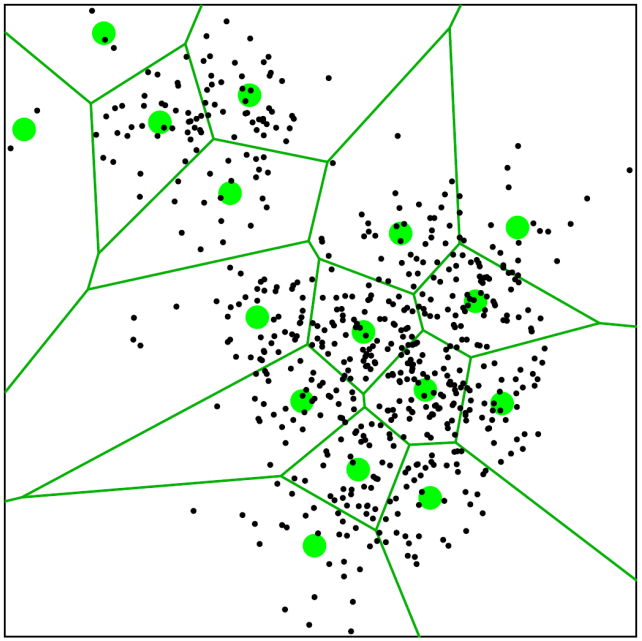
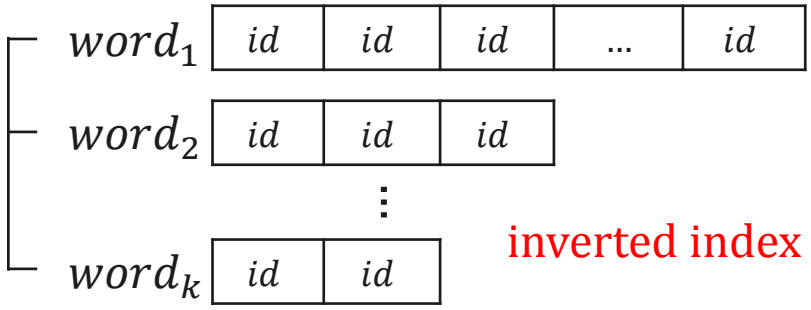


Figure from Lempitsky's slides

- Build an inverted index
  - Quantize each descriptor into the closest word
  - Organize desc. IDs in terms of words

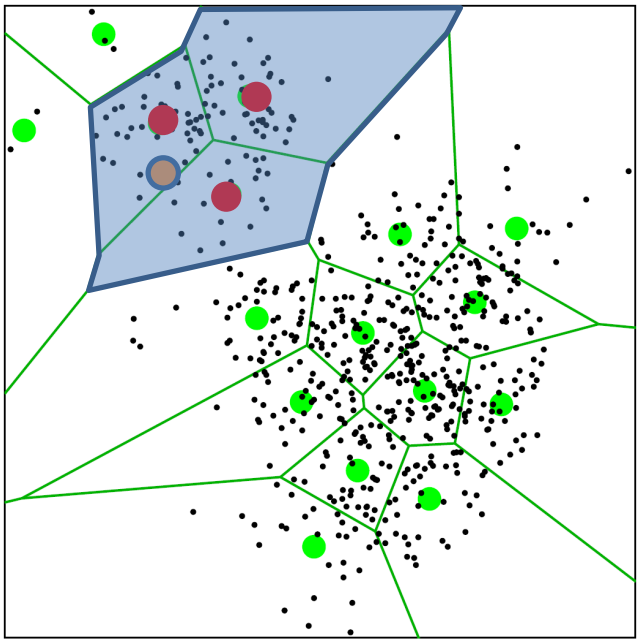


Ack.: Zhe Lin

Inverted Index

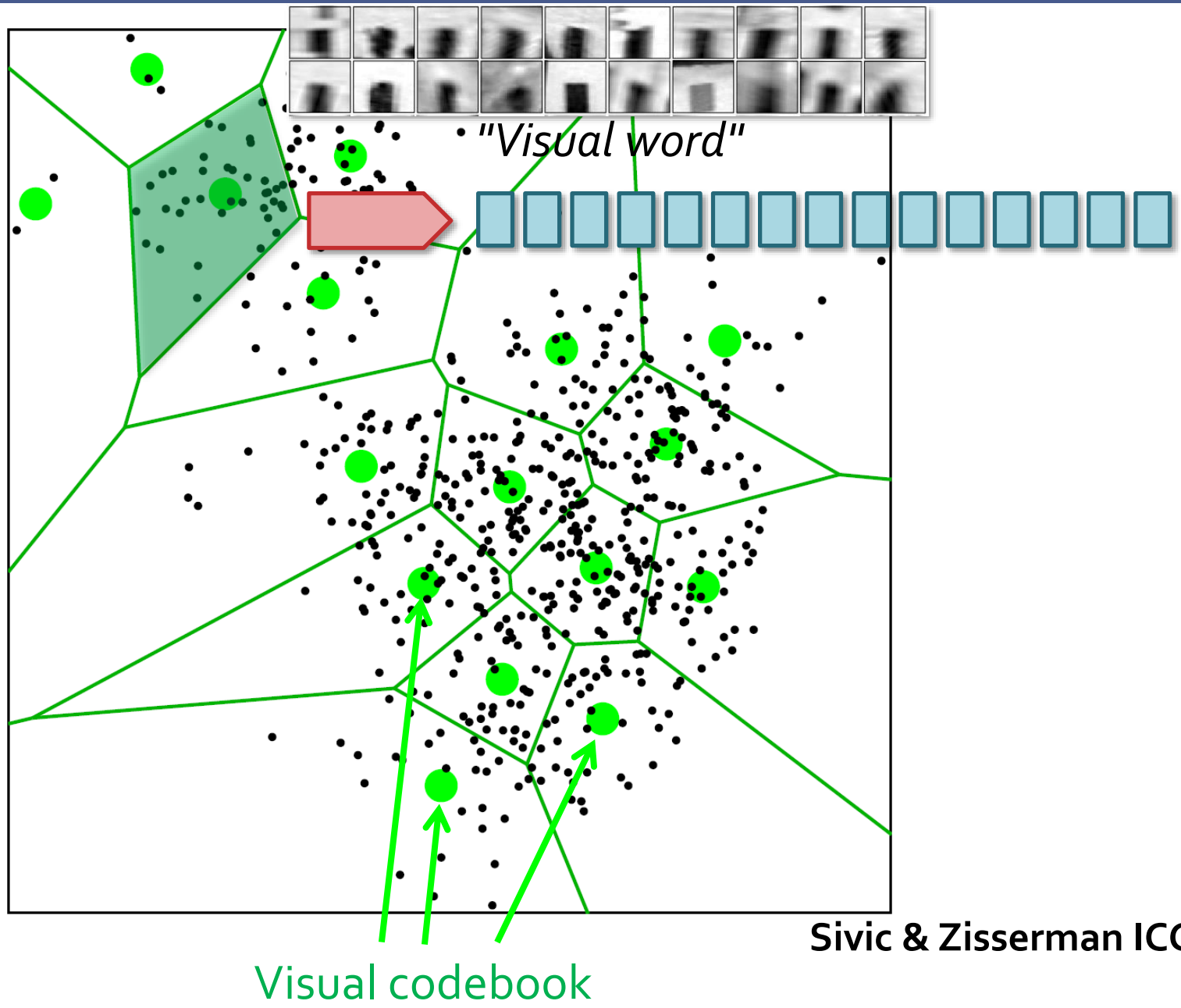
Query time:

- Given a query,
  - Find its K closest words
  - Retrieve all the data in the K lists corresponding to the words
- Large K
  - Low quantization distortion
  - Expensive to find kNN words



Ack.: Zhe Lin

The inverted index



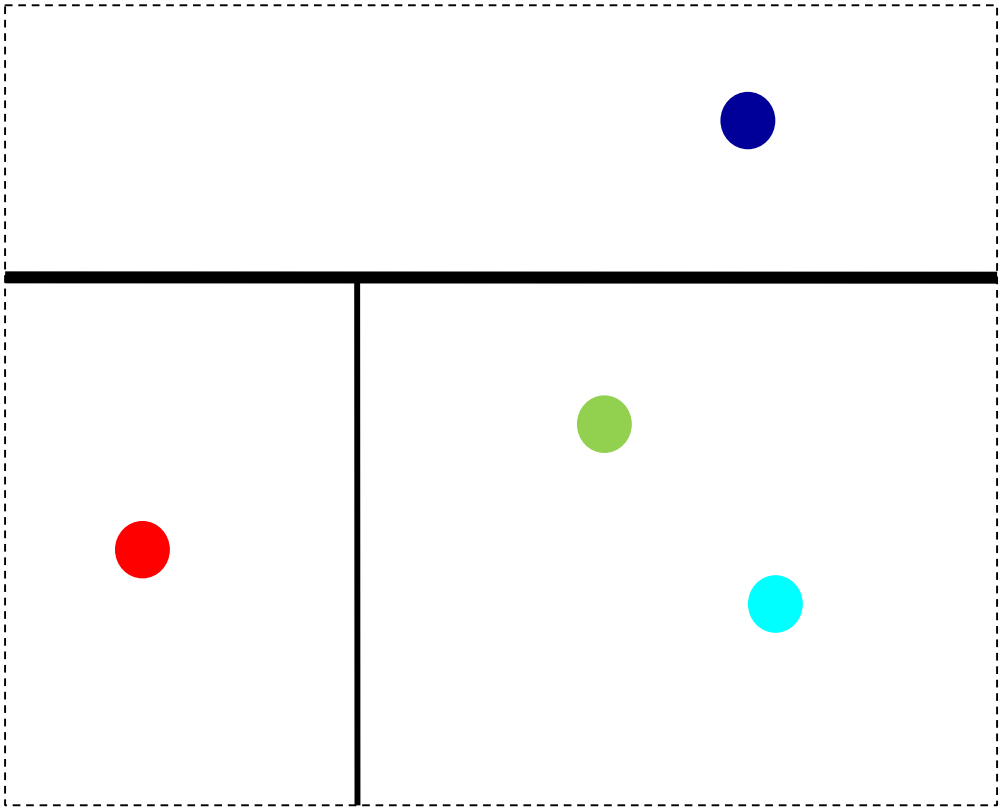
Sivic & Zisserman ICCV 2003

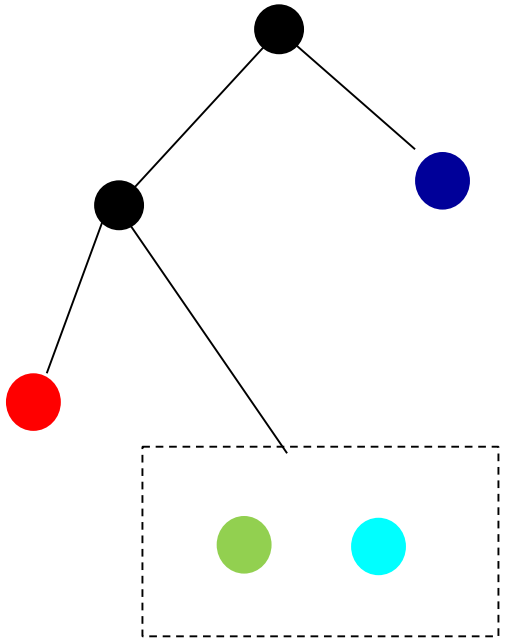
Approximate Nearest Neighbor (ANN) Search

- For large K
  - Takes time to find clusters given the query
  - Use those ANN techniques for efficiently finding near clusters

- ANN search techniques
  - kd-trees: hierarchical approaches for low-dimensional problems
  - Hashing for high dimensional problems; will be discussed later with binary code embedding
  - Quantization (k-means cluster and product quantization)

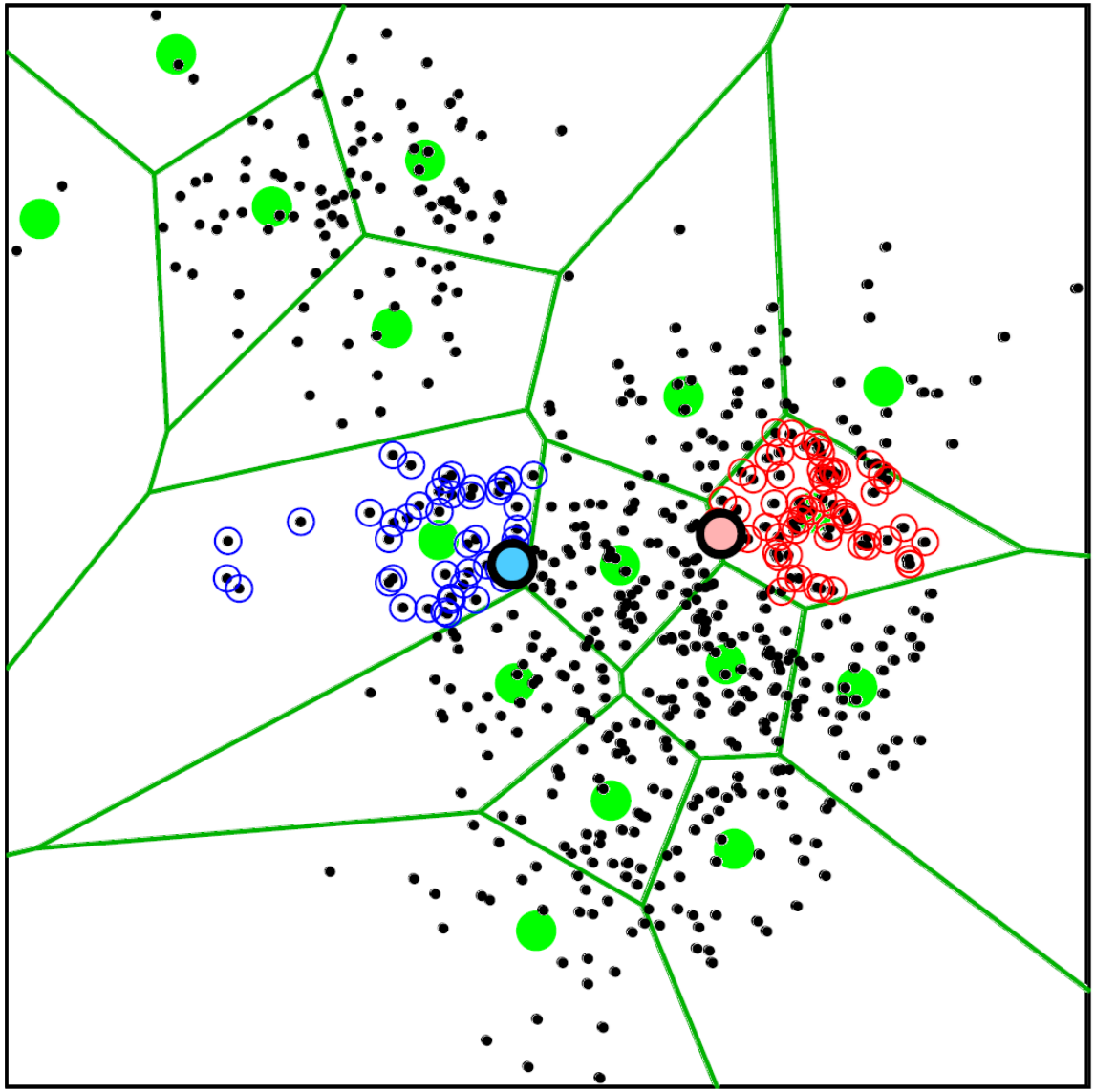
kd-tree Example



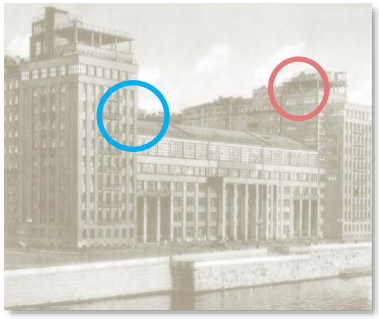


- **Many good implementations (e.g., vl-feat)**

Querying the inverted index



Query:



- Have to consider several words for best accuracy
- Want to use as big codebook as possible

conflict

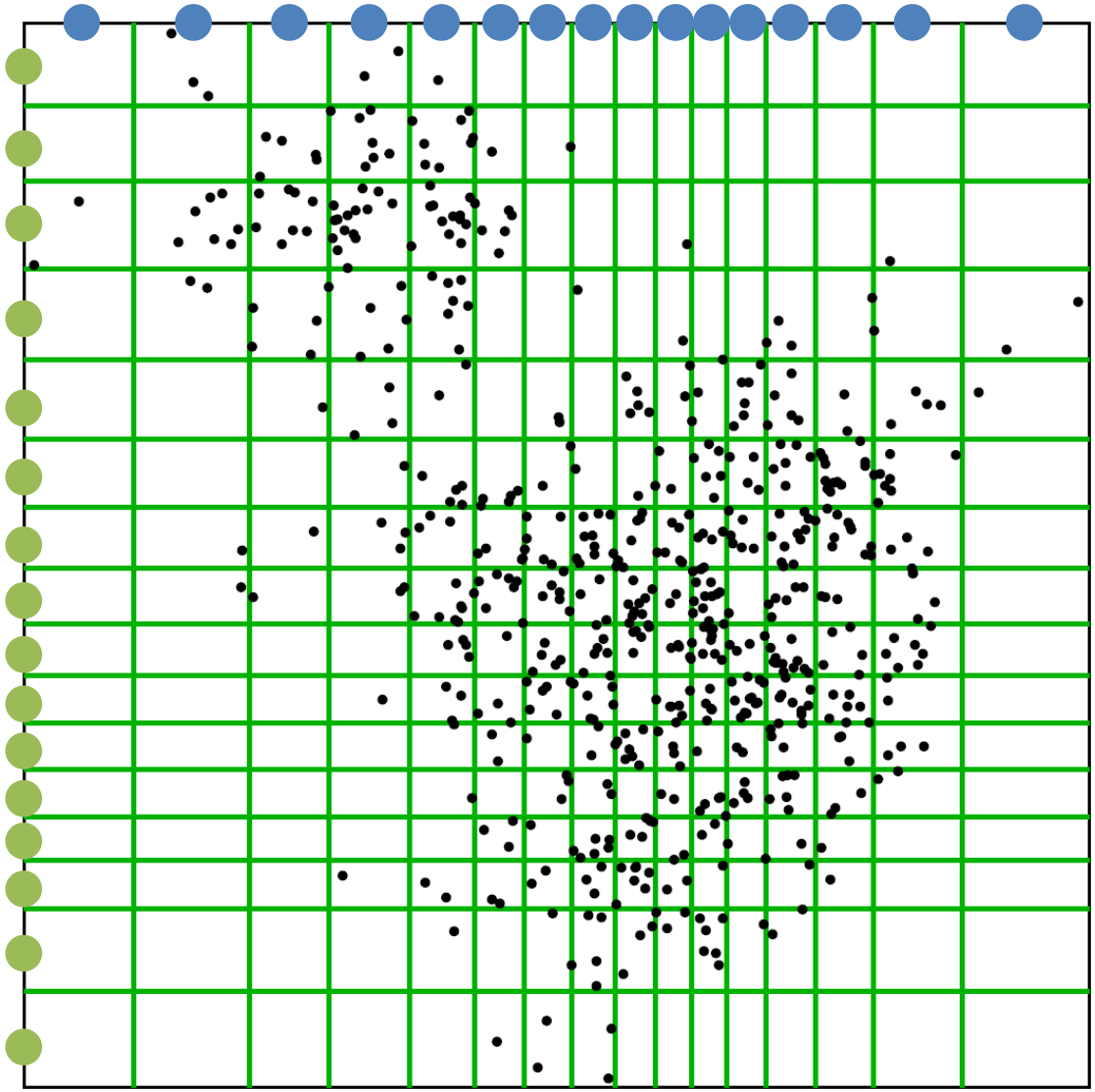
- Want to spend as little time as possible for matching to codebooks

Ack.: Lempitsky

Inverted Multi-Index [Babenko and Lempitsky, CVPR 2012]

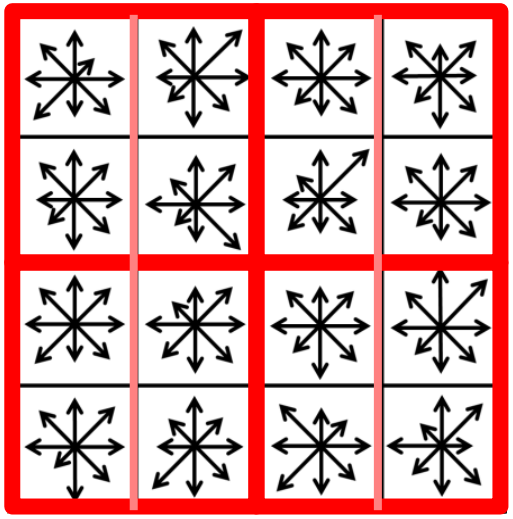
- **Product quantization for indexing**

- **Main advantage:**
  - For the same K, much finer subdivision
  - Very efficient in finding kNN codewords



Ack.: Lempitsky

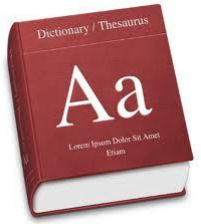
Product quantization



[Jegou, Douze, Schmid // TPAMI 2011]:

1. Split vector into correlated subvectors
2. use separate small codebook for each chunk

Quantization vs. Product quantization:

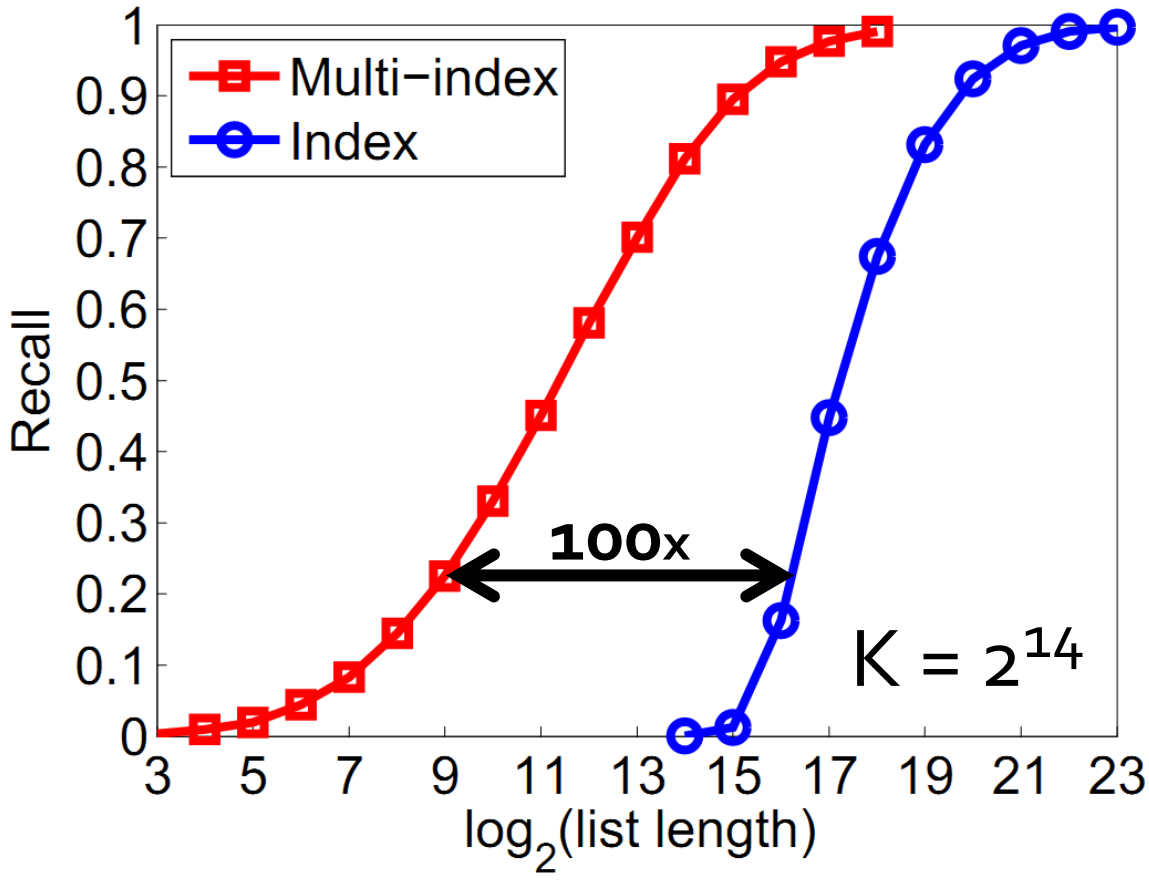


For a budget of 4 bytes per descriptor:

1. Use a single codebook with 1 billion codewords or	**many minutes**	**128GB**
2. Use 4 different codebooks with 256 codewords each	**< 1 millisecond**	**32KB**

Ack.: Lempitsky

Performance comparison on 1 B SIFT descriptors

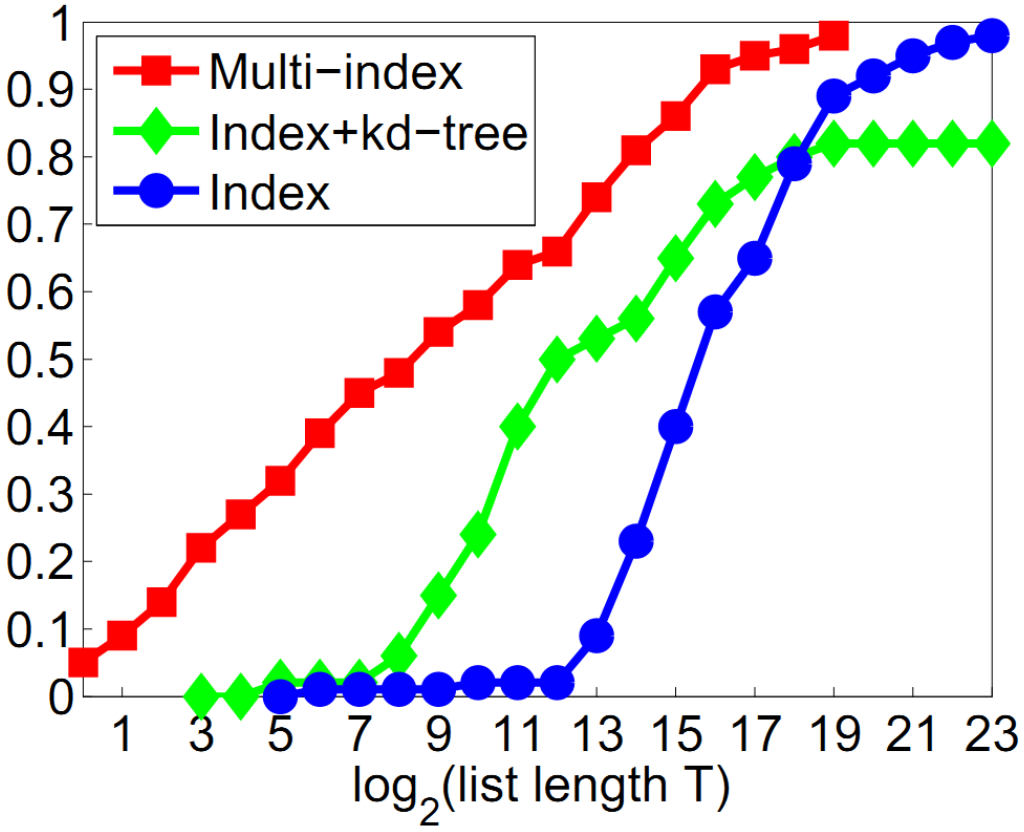


Time increase: 1.4 msec -> 2.2 msec on a single core (with BLAS instructions)

Ack.: Lempitsky

Performance on 80 million GISTs

Index vs Multi-index:



Tests on 80 million GISTs (384 dimensions) of Tiny Images [Torralba et al. PAMI'08]

Ack.: Lempitsky

Retrieval examples

Exact NN
Uncompressed GIST

Multi-D-ADC
16 bytes

Exact NN
Uncompressed GIST

Multi-D-ADC
16 bytes

Exact NN
Uncompressed GIST

Multi-D-ADC
16 bytes

Exact NN
Uncompressed GIST

Multi-D-ADC
16 bytes

Ack.: Lempitsky

Scalability

- **Issues with billions of images?**
  - **Searching speed → inverted index**
  - **Accuracy → larger codebooks, spatial verification, expansion, features**
  - **Memory → compact representations**
  - **Easy to use?**
  - **Applications?**
  - **A new aspect?**

Class Objectives were:

- **Bag-of-visual-Word (BoW) model**
- **Understand approximate nearest neighbor search**
  - **Inverted index**
  - **Inverted multi-index**

Next Time…

- **Learning techniques**

Homework for Every Class

- **Go over the next lecture slides**
- **Come up with one question on what we have discussed today**
  - **1 for typical questions (that were answered in the class)**
  - **2 for questions with thoughts or that surprised me**
- **Write questions at least 4 times**