Multimedia Technology

Lecture 8: Feature Matching and Aggregation

Lecturer: *Dr*. Wan-Lei Zhao

Autumn Semester 2022

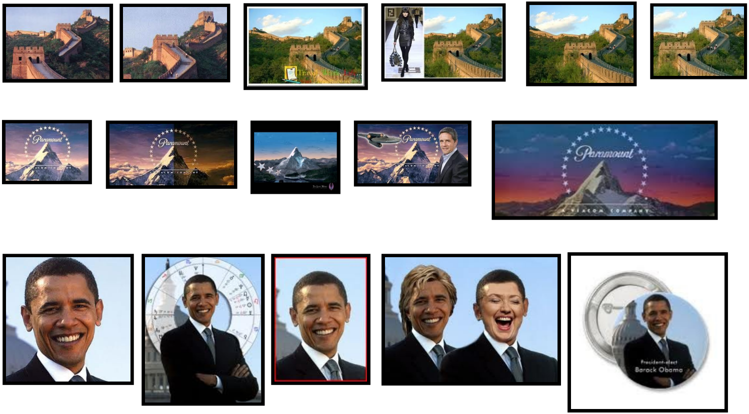
Email: wlzhao@xmu.edu.cn,

Outline

Introduction: opening discussion

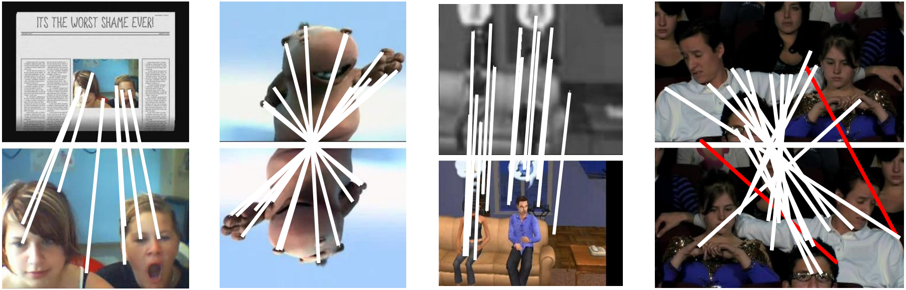
- Image features
  - Global Features: Color-Moment, Color-Histogram
  - Descriptor: SIFT and SURF
  - Deep local Features: DELF, R-MAC
- We are now ready to compare images by their features

Introduction: image near-duplicates (1)



- More than **22%** of web images have similar/near-duplicate counterparts

Advantages of image local feature



(a) scaling (b) rotation (c) blur+scaling (d) flip

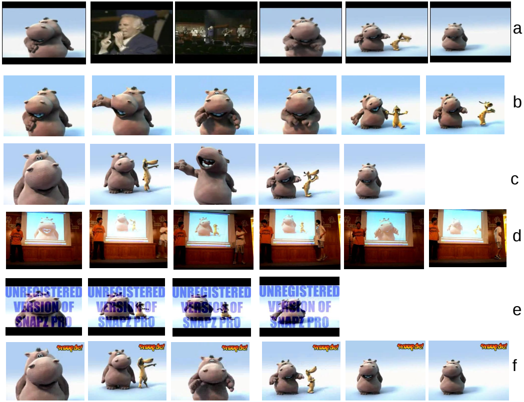
- Robust to transformations such as scaling, rotation, cropping and etc.
- Invariant to flipping
- One-to-one region correspondence between sub-regions is established

The Scale of the Problem: Image Case

- The Complexity of Feature Matching
  - Given 1,500 features are extracted from one image
  - One feature is of 128 dimensions
  - $1500 \times 128 \times 4 = 768,000$ bytes
  - Given there are 10 billions of images in the database
  - Memory cost is: 768×10^4G bytes
  - Time for one query: 0.2×10^{10}s $= 63.4$ years

Introduction: video near-duplicate (2)

- Different versions of "Lion Sleeps Tonight"

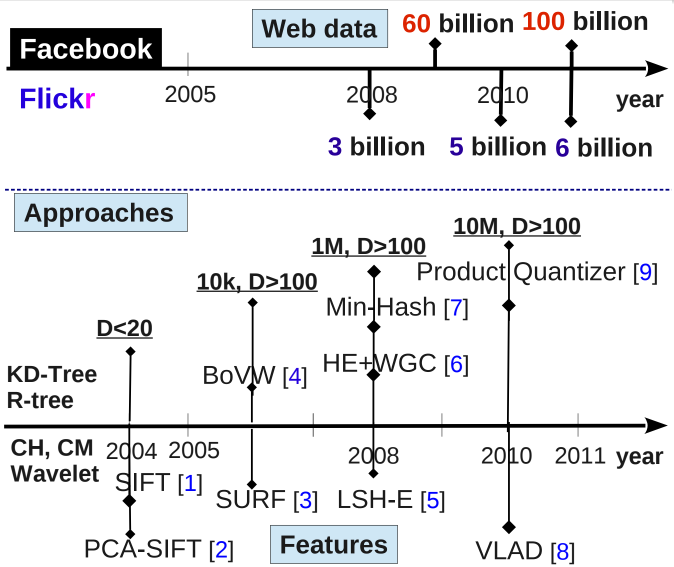


(a) a. mixture of several videos; b. color changes; c. frame dropping; d. camcoding; e. superimpose texts; f. superimpose logos

The Scale of the Problem: Video Case

- The Compleixty of Video Feature Matching
  - Given 10 minutes video
  - Two frames/second, $2\times60\times10 = 1200$
  - One feature is extracted from one frame
  - One feature is of 128 dimensions
  - $1200\times128\times4 = 614,400$ bytes
  - Given there are 100 millions of videos in the database
  - Memory cost is: 61.44T bytes
  - Time cost for one query: 0.2×10^8s = 231.48 days

Related Works: Image near-duplicate Retrieval/Detection

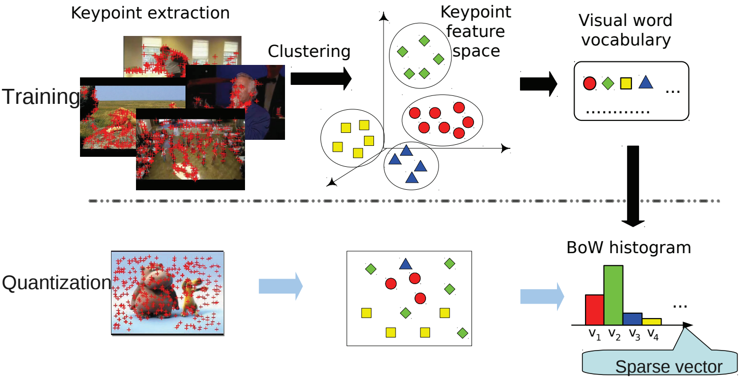


Outline

Why BoW is preferred over point-to-point matching

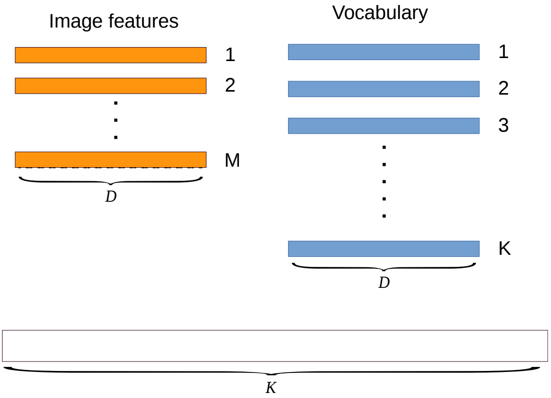
- Challenges
  - Speed efficiency: 1 day for cross-matching within 600 images
  - Memory efficiency: the size of feature $>$ the size of image
  - For **1,000** hours web-videos, more than **600,000** images are extracted, computation costs are counted in **CPU years**
- Opportunities
  - Video is composed by image sequence with certain temporal order and rate (e.g., *25*fps)
  - Approach for ND image retrieval/detection is extensible to ND video retrieval/detection
  - Bag-of visual words (BoW) framework [4]

Bag-of visual words (BoW) Framework



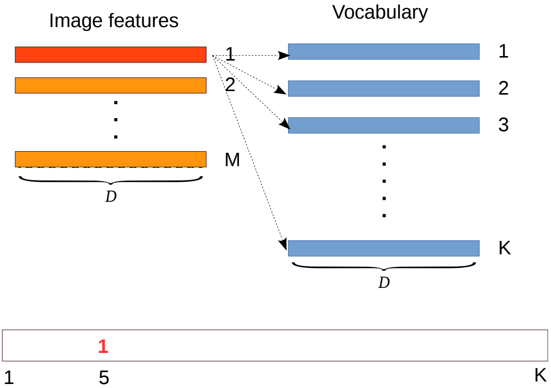
- Advantages: inverted file can be leveraged, matching becomes highly efficient
  - Only 0.62s for 1 query against 1M images, while OOS takes 139 hours
- Disadvantages: introduces many false matches, loss of correct matches

BoW Offline Quantization: explained (1)



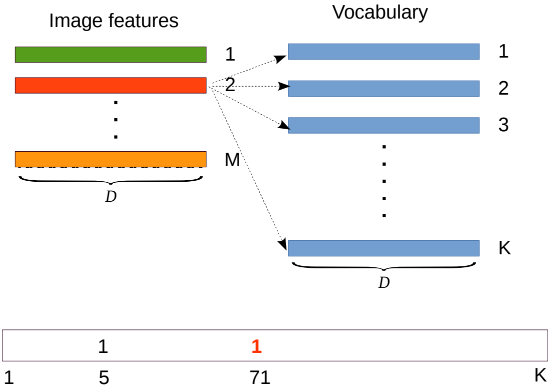
- Given image features (say SIFTs)
- Given vocabulary trained by K-means
- Quantization searches nearest neighbor for each feature in the vocabulary

BoW Offline Quantization: explained (2)



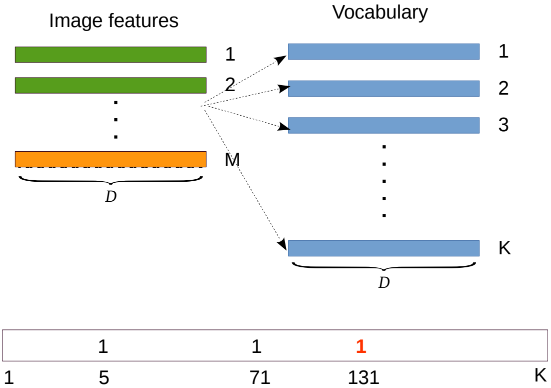
- Given image features (say SIFTs)
- Given vocabulary trained by K-means
- We count the term frequency (TF) that each word appears in the image

BoW Offline Quantization: explained (3)



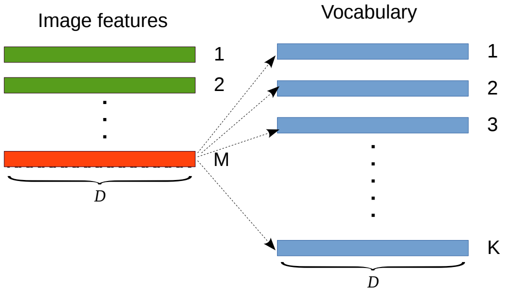
- Given image features (say SIFTs)
- Given vocabulary trained by K-means
- We count the term frequency (TF) that each word appears in the image

BoW Offline Quantization: explained (4)



- Given image features (say SIFTs)
- Given vocabulary trained by K-means
- We count the term frequency (TF) that each word appears in the image

BoW Offline Quantization: explained (5)



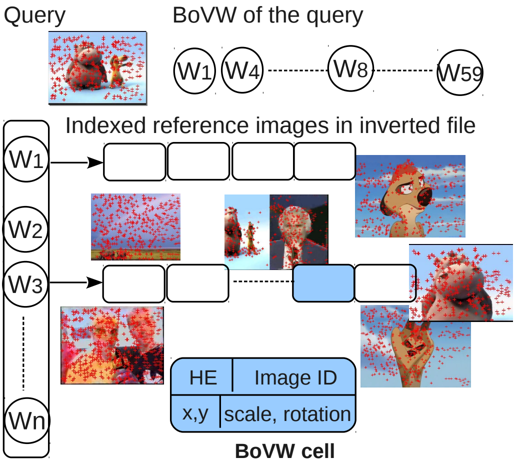
1	5	32	71	131	211	K
	1	1	5	2	2	

- Given image features (say SIFTs)
- Given vocabulary trained by K-means
- We count the term frequency (TF) that each word appears in the image

BoW Offline Quantization: comments

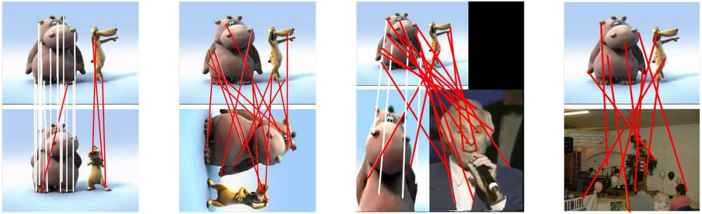
- BoW quantization is a typical vector quantization
- It maps a D-dimensional vector into an integer
- Good news:
  if the vocabulary is large enough, the resulting vector is very sparse
- Bad news: because of quantization, many details get lost

Online Retrieval with BoW: explained



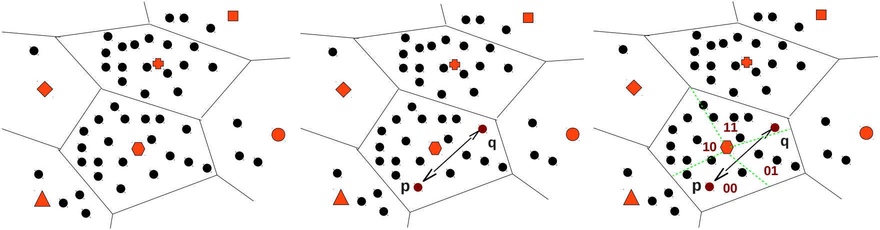
- Local features of an image are represented by TF/IDF of visual words

Noisy feature matches from BoW matching



- How to remove the false matches as many as possible?
- Principle
  1. Integratable with BoW framework
  2. As efficient as possible
- Current solution
  - Hamming embedding [6] (visual verification)
  - Weak Geometric Constraint [6] (geometric verification)

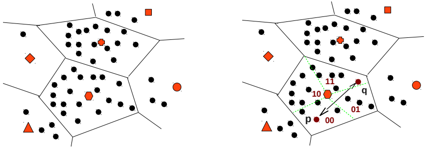
Hamming Embedding: the idea



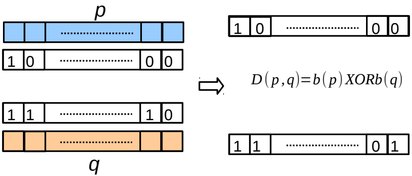
(a) BoVW (b) D(q,p)==0 (c) D(q,p)==2

- BoVW is modeled as Voronoi diagram in feature space
- Feature points in one cell are with zero distances to each other
- Hamming Embedding helps to estimate the intra-distance efficiently

Hamming Embedding: explained



(a) BoVW (b) D(q,p)==2



- Hamming Embedding helps to estimate the intra-distance efficiently
- This Hamming signature is used to prune noisy matches
- A simple idea is to prune matches hold Hamming distance larger than a threshold

Hamming Embedding: offline training

- Step 1. Produce projection matrix
  1. Draw a $D \times D$ white Gausian noise matrix $\mathbf{M}$
  2. Perform QR decomposition on $\mathbf{M}$
  3. Select first K vector from Q to form projection matrix $\mathbf{P}$
- Step 2. Train median for each visual word
  1. Foreach training SIFT f_i do
  2. Quantize f_i into visual word w_j
  3. Project f_i by $p_i = f_i^T P$
  4. Join p_i to U_j
  5. endFor
  6. Find median m_j for each U_j

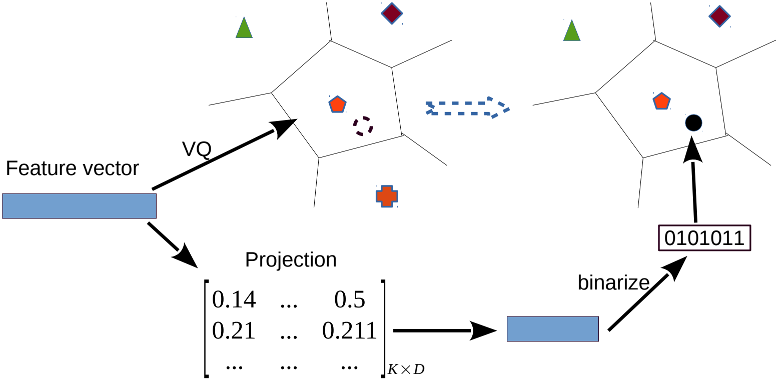
Hamming Embedding: online quantization (1)

- Online quantization for one image
  1. Foreach SIFT f_i in one image
  2. Quantize f_i into visual word w_j
  3. Project f_i by $p_i = f_i^T P$
  4. Binarize p_i based on m_j
  5. endFor

$$b(p_{ik}) = \begin{cases} 1 & p_{ik} > m_{jk} \\ 0 & p_{ik} <= m_{jk} \end{cases}$$

- Above procedure quantizes SIFT features in one image
- A binary signature with **K** bits is generated for each SIFT feature
- This signature is used for verification

Hamming Embedding: online quantization (2)

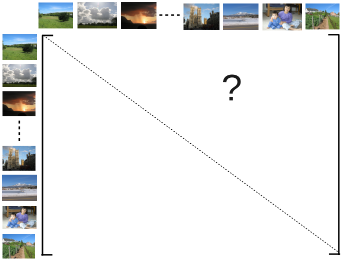


- A binary signature is attached to each quantized feature
- It is later used to verify the visual word match
- It introduces extra memory cost

Outline

Background: image-linking within big collection

- Build hyper-links between images in the web
- Find near-duplicate shots in video collections



- Compute a matrix with $N \times N$ entries
- Requires huge memory and computationally intensive!!
- Called as **image-linking** problem

Motivation: min-Hash

$X = \{A, B, C\}$

$Y = \{B, C, E\}$

$$J(X, Y) = \frac{|X \cap Y|}{|X \cup Y|}$$

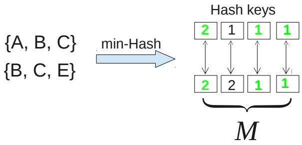
Random permutation

$\pi_j(.)$	A	B	C	D	E
$\pi_1(.)$	3	5	2	1	4
$\pi_2(.)$	1	2	5	3	4
$\pi_3(.)$	2	1	4	5	3
$\pi_4(.)$	5	1	3	4	2

Min-Hash

σ_j	ABC	BCE
$min\,\pi_1(.)$	2	2
$min\,\pi_2(.)$	1	2
$min\,\pi_3(.)$	1	1
$min\,\pi_4(.)$	1	1

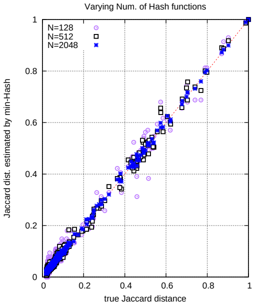
$$\approx \frac{1}{M} \sum_{j=1}^{M} \delta(\sigma_j(X) = \sigma_j(Y))$$



$$J(X, Y) \approx \frac{3}{4}$$

- The probability of key collision (equal key value) equals to J
- The complexity of computing $J(X, Y)$ is $O(nlog(n))$
- Only O(M) if min-Hash is adopted!!

How well min-Hash is??



- The more hash functions we use, the better approximation we get
- However the higher the cost it takes

min-Hash sketches

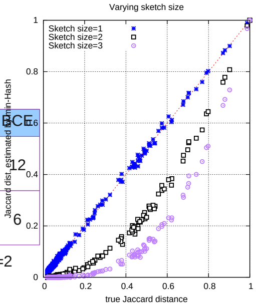
- Combine keys into sketch

σ_j	ABC	BCE
$min\,\pi_1(.)$	2	2
$min\,\pi_2(.)$	1	2
$min\,\pi_3(.)$	1	1
$min\,\pi_4(.)$	1	1

Sketch size=1

$\sigma_j * N + \sigma_{j+1}$	ABC	BCE
K1	11	12
K2	6	6

Sketch size=2



- Reduce the complexity further
- Equivalent to a co-occurrence constraint
- Degraded estimation (potential matches have been missed)
- The sketch size is set to *2* in our experiments

Sim-min-Hash: the motivation

$X=\{A,B,C\}$

$Y=\{B,C,E\}$

$$J(X,Y)=\frac{|X\cap Y|}{|X\cup Y|}$$

Random permutation

$\pi_j(.)$	A	B	C	D	E
$\pi_1(.)$	3	5	2	1	4
$\pi_2(.)$	1	2	5	3	4
$\pi_3(.)$	2	1	4	5	3
$\pi_4(.)$	5	1	3	4	2

Min-Hash

σ_j	ABC	BCE
$min\,\pi_1(.)$	2	2
$min\,\pi_2(.)$	1	2
$min\,\pi_3(.)$	1	1
$min\,\pi_4(.)$	1	1

$$\approx \frac{1}{M}\sum_{j=1}^{M}\delta(\sigma_j(X)=\sigma_j(Y))$$

min-Hash

$$\begin{cases}\delta(.)=0 & \sigma_j(X)\neq\sigma_j(Y)\\ \delta(.)=1 & \sigma_j(X)=\sigma_j(Y)\end{cases}$$

$\Downarrow$

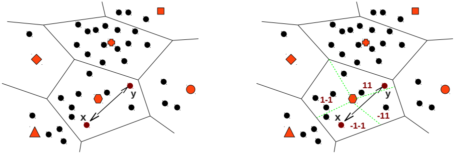
Sim-min-Hash

$$\begin{cases}\delta(.)=0 & \sigma_j(X)\neq\sigma_j(Y)\\ \delta(.)=\Omega(x,y) & \sigma_j(X)=\sigma_j(Y)\end{cases}$$

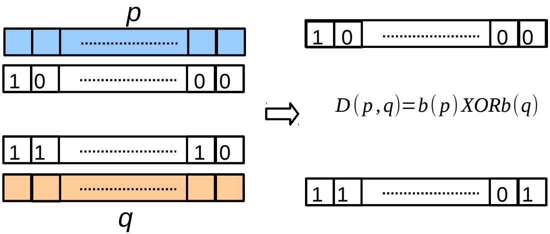
- Similarity between objects x and y is considered

Measure $\Omega(x, y)$ by Hamming embedding (HE)

- Given we are under the context of BoVW



(a) BoVW (b) HE



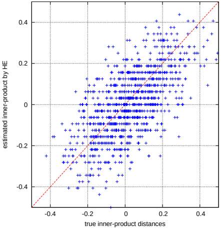
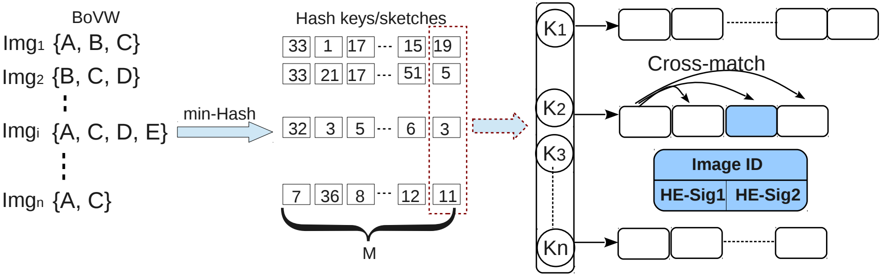
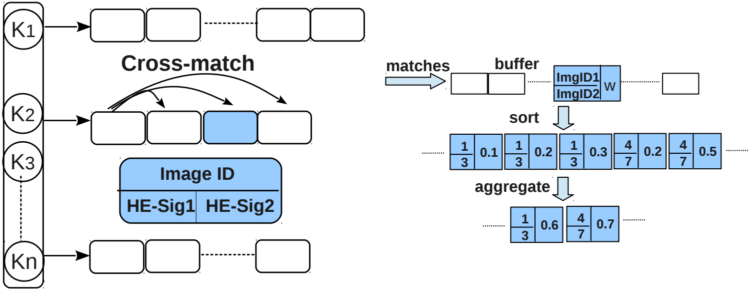


Image-linking with Sim-min-Hash



1. Load one column of sketches into one inverted file
2. Perform cross-matching on each inverted list

Image-linking with Sim-min-Hash



1. Only the matches whose $\Omega(x, y) > \tau$ are kept
2. Matches are sorted by Image IDs after matching
3. Matches belonging to the same image pair are aggregated

Results by examples

Figure: Only the links whose confidence score above 2.0 are shown.

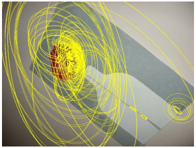
Summary over Sim-min-Hash

- Achieves much better trade-off between speed and quality
- Can be scaled up to 100 million level image collections
- Promising for web-scale data

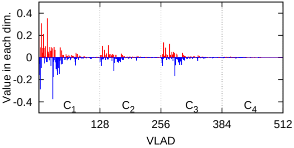
Outline

VLAD: framework

- Given vocabulary $\mathcal{W}(C_i \in \mathcal{W})$ and features $\mathrm{P}(x_i \in \mathrm{P})$ in one image

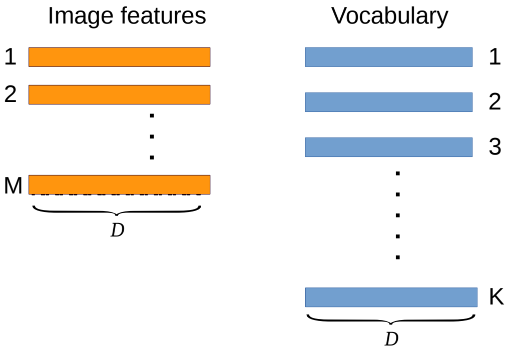


$$V_{i,j} = \sum_{x \in \chi : q(x) = C_i} x_j - C_{i,j}$$



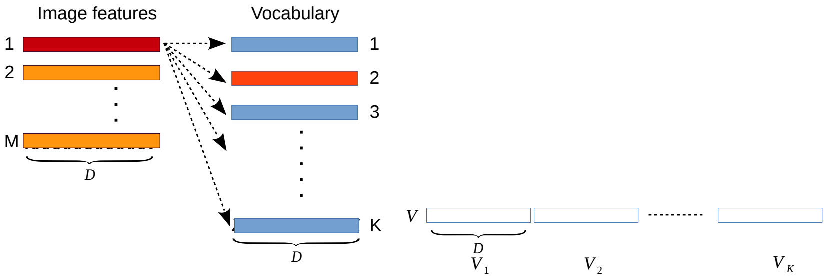
- General procedure
  - Foreach SIFT feature x_i do
  - Find the nearest neighbor k in $\mathcal{W}$ for x_i
  - Substract x_i with C_k
  - Aggregate this residue to V_k
  - endFor
- This results in a long dense vector representation for one image

VLAD: explained (1)



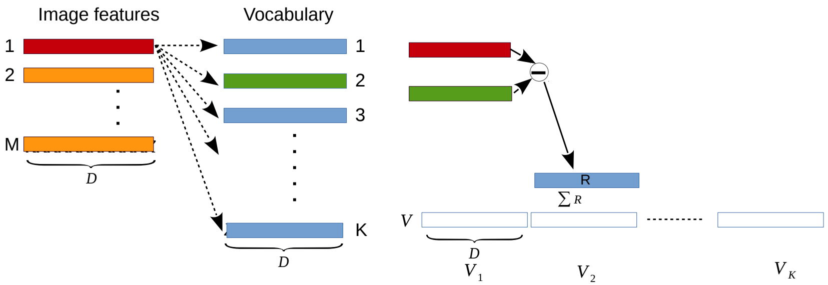
- Given image features $\{x_i\}$ and vocabuarly $\mathcal{W}\{w_j\}$

VLAD: explained (2)



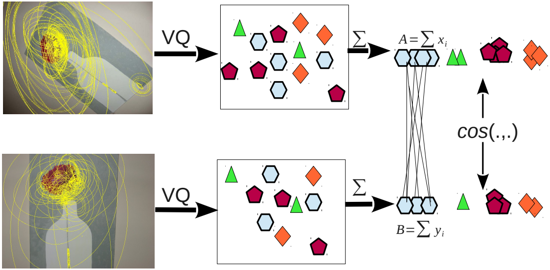
- Given image features $\{x_i\}$ and vocabuarly $\mathcal{W}\{c_j\}$
- Find the nearest neighbor k in $\mathcal{W}$ foreach x_i

VLAD: explained (3)



- Given image features $\{x_i\}$ and vocabuarly $\mathcal{W}\{c_j\}$
- Find the nearest neighbor k in $\mathcal{W}\{c_j\}$ foreach x_i
-   Substract x_i with C_k: $R = x_i - C_k$
-   Aggregate this residue R to V_k
- Output $V_1 V_2 \cdots V_K$

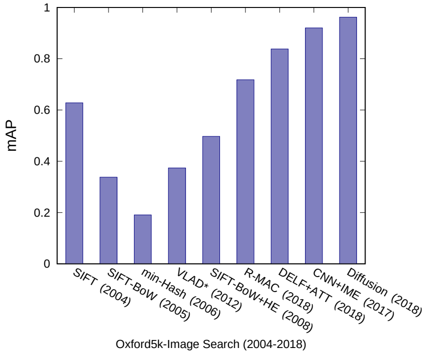
VLAD: equivalent to matching groups of features



$$A^t \cdot B = \sum \begin{bmatrix} x_1 \\ .. \\ x_n \end{bmatrix}^t \cdot \begin{bmatrix} y_1 \\ .. \\ y_n \end{bmatrix} \tag{1}$$

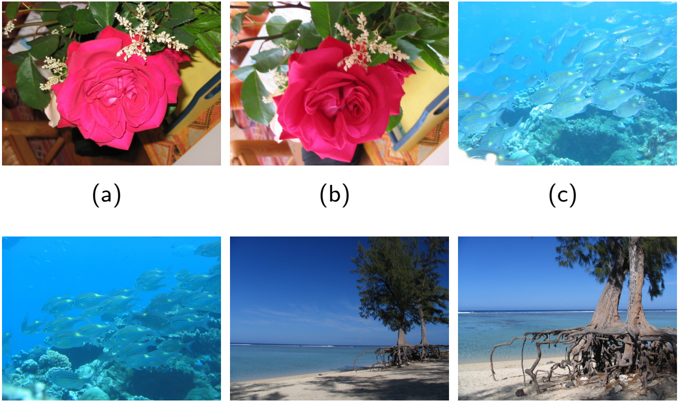
- Many-to-Many feature matching
- Problem: matching via VLAD introduces too many noises

Performance Comparison on Oxford5k



- VLAD performs pretty well with much lower memory complexity
- BoVW+HE performs well the best at the cost of much more memory

Holidays Dataset



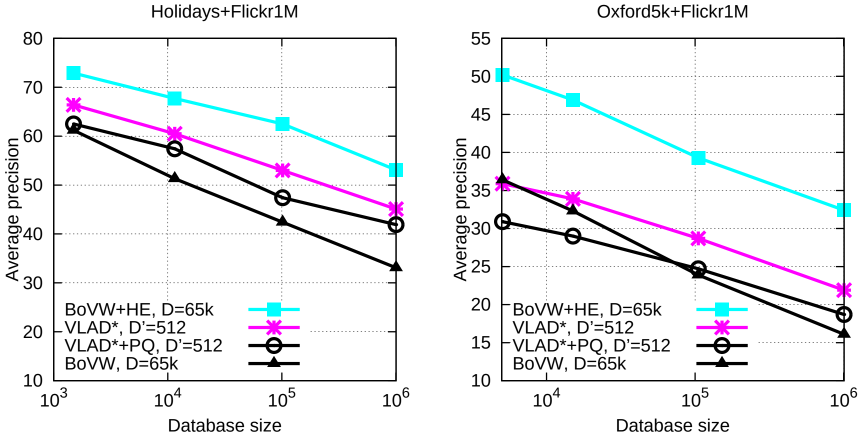
(a) (b) (c)

(d) (e) (f)

- There are *1,491* images captured from around the world[1]
- *500* images are selected as the query

[1]https://lear.inrialpes.fr/ jegou/data.php

Holidays and Oxford5k



(a) Holidays+1M (b) Oxford5k+1M

- Measured by mean Average Precision
- VLAD performs pretty well with much lower memory complexity

References

1. Distinctive Image Features from Scale-Invariant Keypoints, D. G. Lowe, *IJCV'10*
2. Near-duplicate Image and Video Detection, Wan-Lei Zhao, Chong-Wah Ngo, *Wiley Encyclopedia of Electrical and Electronics Engineering*, 2015
3. SURF: Speeded Up Robust Features, H. Bay and et al., *ECCV'06*
4. Video Google: Efficient Visual Search of Videos, J. Sivic and et al., 2006
5. Efficiently Matching Sets of Features with Random Histograms, W. Dong and et al., *MM'08*
6. Embedding and Weak Geometry Constraint on Bag-of visual Keyword, H. Jegou and et al., *ECCV'08*
7. Near Duplicate Image Detection: min-Hash and tf-idf Weighting, O. Chum and et al., *BMVC'08*
8. Aggregating local descriptors into a compact image representation, H. Jegou and et al., CVPR'10
9. Product quantization for nearest neighbor search, H. Jegou and et al., *PAMI'11*
10. Sim-Min-Hash: An efficient matching technique for linking large image collections, Wan-Lei Zhao, Herve Jegou, Guillaume Gravier, *ACM MM'13*
11. Content-based copy detection using distortion-based probabilistic similarity search, A. Joly and et al., in *TMM'07*

Q & A

Thanks for your attention!