Instance level recognition IV:
Very large databases

Cordelia Schmid
LEAR – INRIA Grenoble
Visual search

close-up change in viewing angle
Matches

22 correct matches
Image search system for large datasets

- **Issues** for very large databases
  - to reduce the query time
  - to reduce the storage requirements
  - with minimal loss in retrieval accuracy

**Large image dataset**
(one million images or more)
Large scale object/scene recognition

Image dataset: > 1 million images

- Each image described by approximately 2000 descriptors
  - $2 \times 10^9$ descriptors to index for one million images!

- Database representation in RAM:
  - Size of descriptors: 1 TB, search+memory intractable
Bag-of-features [Sivic & Zisserman’03]

- **Visual Words**
  - 1 word (index) per local descriptor
  - only images ids in inverted file
  \(\Rightarrow\) 8 GB for a million images, fits in RAM

- **Problem**
  - Matching approximation

[Chum & al. 2007]
Visual words – approximate NN search

- Map descriptors to words by quantizing the feature space
  - Quantize via k-means clustering to obtain visual words
  - Assign descriptors to closest visual words

- Bag-of-features as approximate nearest neighbor search

  Descriptor matching with $k$-nearest neighbors

  $$f_{k\text{-NN}}(x, y) = \begin{cases} 1 & \text{if } x \text{ is a } k\text{-NN of } y \\ 0 & \text{otherwise} \end{cases}$$

  Bag-of-features matching function

  $$f_q(x, y) = \delta_{q(x), q(y)}$$

  where $q(x)$ is a quantizer, i.e., assignment to a visual word and
  $\delta_{a,b}$ is the Kronecker operator ($\delta_{a,b}=1$ iff $a=b$)
Approximate nearest neighbor search evaluation

• ANN algorithms usually returns a short-list of nearest neighbors
  – this short-list is supposed to contain the NN with high probability
  – exact search may be performed to re-order this short-list

• Proposed quality evaluation of ANN search: trade-off between
  – **Accuracy**: **NN recall** = probability that *the* NN is in this list
  – **Ambiguity removal** = proportion of vectors in the short-list
    - the lower this proportion, the more information we have about the vector
    - the lower this proportion, the lower the complexity if we perform exact search on the short-list

• ANN search algorithms usually have some parameters to handle this trade-off
ANN evaluation of bag-of-features

- ANN algorithms returns a list of potential neighbors

- **Accuracy**: NN recall
  = probability that the NN is in this list

- **Ambiguity removal**: = proportion of vectors in the short-list

- In BOF, this trade-off is managed by the number of clusters $k$
Vocabulary size

• The intrinsic matching scheme performed by BOF is weak
  – for a “small” visual dictionary: too many false matches
  – for a “large” visual dictionary: complexity, true matches are missed

• No good trade-off between “small” and “large”!
  – either the Voronoi cells are too big
  – or these cells can’t absorb the descriptor noise
  → intrinsic approximate nearest neighbor search of BOF is not sufficient
20K visual word: false matches
200K visual word: good matches missed
Hamming Embedding [Jegou et al. ECCV'08]

Representation of a descriptor $x$
- Vector-quantized to $q(x)$ as in standard BOF
+ short binary vector $b(x)$ for an additional localization in the Voronoi cell

Two descriptors $x$ and $y$ match iif

$$f_{HE}(x, y) = \begin{cases} 
  (\text{tf-idf}(q(x)))^2 & \text{if } q(x) = q(y) \\
  0 & \text{and } h(b(x), b(y)) \leq h_t \\
  h(a, b) & \text{Hamming distance} 
\end{cases}$$
Term frequency – inverse document frequency

• Weighting with tf-idf score: weight visual words based on their frequency

• Tf: normalized term (word) frequency $t_i$ in a document $d_j$

$$tf_{ij} = \frac{n_{ij}}{\sum_k n_{kj}}$$

• Idf: inverse document frequency, total number of documents divided by number of documents containing the term $t_i$

$$idf_i = \log \frac{|D|}{|\{d : t_i \in d\}|}$$

Tf-Idf: $tf - idf_{ij} = tf_{ij} \cdot idf_i$
Hamming Embedding [Jegou et al. ECCV’08]

• Nearest neighbors for Hamming distance ≈ those for Euclidean distance
  → a metric in the embedded space reduces dimensionality curse effects

• Efficiency
  – Hamming distance = very few operations
  – Fewer random memory accesses: 3 x faster that BOF with same dictionary size!
Hamming Embedding

• **Off-line** (given a quantizer)
  - draw an orthogonal projection matrix $P$ of size $d_b \times d$
  - this defines $d_b$ random projection directions
  - for each Voronoi cell and projection direction, compute the median value for a learning set

• **On-line**: compute the binary signature $b(x)$ of a given descriptor
  - project $x$ onto the projection directions as $z(x) = (z_1, \ldots z_{db})$
  - $b_i(x) = 1$ if $z_i(x)$ is above the learned median value, otherwise 0
Hamming neighborhood

Trade-off between memory usage and accuracy

→ More bits yield higher accuracy

In practice, 64 bits (8 byte)
ANN evaluation of Hamming Embedding

compared to BOW: at least 10 times less points in the short-list for the same level of accuracy

Hamming Embedding provides a much better trade-off between recall and ambiguity removal
Matching points - 20k word vocabulary

201 matches

240 matches

Many matches with the non-corresponding image!
Matching points - 200k word vocabulary

69 matches  

35 matches

Still many matches with the non-corresponding one
Matching points - 20k word vocabulary + HE

83 matches

8 matches

10x more matches with the corresponding image!
Bag-of-features [Sivic & Zisserman’03]

Query image

Harris-Hessian-Laplace regions + SIFT descriptors

Set of SIFT descriptors

centroids (visual words)

Bag-of-features processing + tf-idf weighting

sparse frequency vector

Inverted file

querying

Re-ranked list

Geometric verification

ranked image short-list

[Chum & al. 2007]
Geometric verification

Use the **position** and **shape** of the underlying features to improve retrieval quality

Both images have many matches – which is correct?
Geometric verification

We can measure **spatial consistency** between the query and each result to improve retrieval quality.

Many spatially consistent matches – **correct result**

Few spatially consistent matches – **incorrect result**
Geometric verification

Gives localization of the object
Weak geometry consistency

- Re-ranking based on full geometric verification
  - works very well
  - but performed on a short-list only (typically, 100 images)
  → for very large datasets, the number of distracting images is so high that relevant images are not even short-listed!

![Graph showing the rate of relevant images short-listed versus dataset size for different short-list sizes (20 images, 100 images, 1000 images).]
Weak geometry consistency

- Weak geometric information used for all images (not only the short-list)

- Each invariant interest region detection has a scale and rotation angle associated, here characteristic scale and dominant gradient orientation

  ![Diagram of scale change and rotation angle]

  Scale change 2
  Rotation angle ca. 20 degrees

- Each matching pair results in a scale and angle difference

- For the global image scale and rotation changes are roughly consistent
WGC: orientation consistency

Max = rotation angle between images
WGC: scale consistency
Weak geometry consistency

• Integration of the geometric verification into the BOF
  – votes for an image in two quantized subspaces, i.e. for angle & scale
  – these subspace are show to be roughly independent
  – final score: filtering for each parameter (angle and scale)

• Only matches that do agree with the main difference of orientation and scale will be taken into account in the final score

• Re-ranking using full geometric transformation still adds information in a final stage
INRIA holidays dataset

• Evaluation for the INRIA holidays dataset, 1491 images
  – 500 query images + 991 annotated true positives
  – Most images are holiday photos of friends and family
• 1 million & 10 million distractor images from Flickr
• Vocabulary construction on a different Flickr set
• Almost real-time search speed

• Evaluation metric: mean average precision (in $[0,1]$, bigger = better)
  – Average over precision/recall curve
Holiday dataset – example queries
Dataset: Venice Channel
Dataset: San Marco square
Example distractors - Flickr
Experimental evaluation

- Evaluation on our holidays dataset, 500 query images, 1 million distracter images
- Metric: mean average precision (in [0,1], bigger = better)

![Graph showing mAP vs database size for different methods]

**Average query time** (4 CPU cores)

<table>
<thead>
<tr>
<th>Method</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compute descriptors</td>
<td>880 ms</td>
</tr>
<tr>
<td>Quantization</td>
<td>600 ms</td>
</tr>
<tr>
<td>Search – baseline</td>
<td>620 ms</td>
</tr>
<tr>
<td>Search – WGC</td>
<td>2110 ms</td>
</tr>
<tr>
<td>Search – HE</td>
<td>200 ms</td>
</tr>
<tr>
<td>Search – HE+WGC</td>
<td>650 ms</td>
</tr>
</tbody>
</table>
Results – Venice Channel

Query

Base 1

Flickr

Flickr

Base 4
Comparison with the state of the art: Oxford dataset [Philbin et al. CVPR'07]

Evaluation measure:
Mean average precision (mAP)
Comparison with the state of the art: Kentucky dataset [Nister et al. CVPR’06]

4 images per object

Evaluation measure: among the 4 best retrieval results how many are correct (ranges from 1 to 4)
Comparison with the state of the art

<table>
<thead>
<tr>
<th>dataset</th>
<th>Oxford</th>
<th>Kentucky</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>distractors</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>100K</td>
</tr>
<tr>
<td>soft assignment [14]</td>
<td>0.493</td>
<td>0.343</td>
</tr>
<tr>
<td>ours</td>
<td>0.615</td>
<td>0.516</td>
</tr>
<tr>
<td>soft + geometrical re-ranking [14]</td>
<td>0.598</td>
<td>0.480</td>
</tr>
<tr>
<td>ours + geometrical re-ranking</td>
<td>0.667</td>
<td>0.591</td>
</tr>
<tr>
<td>soft + query expansion [14]</td>
<td>0.718</td>
<td>0.605</td>
</tr>
<tr>
<td>ours + query expansion</td>
<td>0.747</td>
<td>0.687</td>
</tr>
<tr>
<td>hierarchical vocabulary [6]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ours</td>
<td></td>
<td>3.61</td>
</tr>
<tr>
<td>ours + geometrical re-ranking</td>
<td></td>
<td>3.42</td>
</tr>
</tbody>
</table>

On-line demonstration

Demo at http://bigimbaz.inrialpes.fr
Towards larger databases?

- BOF can handle up to ~10 M d’images
  - with a limited number of descriptors per image
  - 40 GB of RAM
  - search = 2 s

- Web-scale = billions of images
  - With 100 M per machine
    - search = 20 s, RAM = 400 GB
    - not tractable!
Recent approaches for very large scale indexing

- Each image is represented by one vector (not necessarily a BOF)
- This vector is compressed to reduce storage requirements

Query image

Hessian-Affine regions + SIFT descriptors

Set of SIFT descriptors

Bag-of-features processing + tf-idf weighting

centroids (visual words)

sparse frequency vector

Vector compression

Vector search

Re-ranked list

Geometric verification

ranked image short-list
Related work on very large scale image search

- Min-hash and geometrical min-hash [Chum et al. 07-09]
- Compressing the BoF representation (miniBof) [Jegou et al. 09]
  → these approaches require hundreds of bytes to obtain a “reasonable quality”

- GIST descriptors with Spectral Hashing [Weiss et al.’08]
  → very limited invariance to scale/rotation/crop
Global scene context – GIST descriptor

- The “gist” of a scene: Oliva & Torralba (2001)

- 5 frequency bands and 6 orientations for each image location
- Tiling of the image to describe the image
GIST descriptor + spectral hashing

- The position of the descriptor in the image is encoded in the representation

Gist

- Spectral hashing produces binary codes similar to spectral clusters

Torralba et al. (2003)
Related work on very large scale image search

- Min-hash and geometrical min-hash [Chum et al. 07-09]
- Compressing the BoF representation (miniBof) [Jegou et al. 09]
  - require hundreds of bytes are required to obtain a “reasonable quality”

- GIST descriptors with Spectral Hashing [Weiss et al.’08]
  - very limited invariance to scale/rotation/crop

- Aggregating local descriptors into a compact image representation [Jegou & al.’10]

- Efficient object category recognition using classemes [Torresani et al.’10]
Compact image representation

- **Aim:** improving the tradeoff between
  - search speed
  - memory usage
  - search quality

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

---

[H. Jegou et al., Aggregating local desc into a compact image representation, CVPR’10]
Aggregation of local descriptors

- Problem: represent an image by a single fixed-size vector:
  
  set of $n$ local descriptors $\rightarrow$ 1 vector

- Most popular idea: BoF representation [Sivic & Zisserman 03]
  - sparse vector
  - highly dimensional
  $\rightarrow$ high dimensionality reduction/compression introduces loss

- Alternative: vector of locally aggregated descriptors (VLAD)
  - non sparse vector
  - excellent results with a small vector dimensionality
VLAD: vector of locally aggregated descriptors

- Learning: a vector quantifier ($k$-means)
  - output: $k$ centroids (visual words): $c_1, \ldots, c_i, \ldots, c_k$
  - centroid $c_i$ has dimension $d$

- For a given image
  - assign each descriptor to closest center $c_i$
  - accumulate (sum) descriptors per cell
    
    $$v_i := v_i + (x - c_i)$$

- VLAD (dimension $D = k \times d$)

- The vector is L2-normalized

- Alternative: Fisher vector
VLADs for corresponding images

SIFT-like representation per centroid (+ components: blue, - 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

<table>
<thead>
<tr>
<th>Aggregator</th>
<th>k</th>
<th>D</th>
<th>D’=D (no reduction)</th>
<th>D’=128</th>
<th>D’=64</th>
</tr>
</thead>
<tbody>
<tr>
<td>BoF</td>
<td>1,000</td>
<td>1,000</td>
<td>41.4</td>
<td>44.4</td>
<td>43.4</td>
</tr>
<tr>
<td>BoF</td>
<td>20,000</td>
<td>20,000</td>
<td>44.6</td>
<td>45.2</td>
<td>44.5</td>
</tr>
<tr>
<td>BoF</td>
<td>200,000</td>
<td>200,000</td>
<td>54.9</td>
<td>43.2</td>
<td>41.6</td>
</tr>
<tr>
<td>VLAD</td>
<td>16</td>
<td>2,048</td>
<td>49.6</td>
<td>49.5</td>
<td><strong>49.4</strong></td>
</tr>
<tr>
<td>VLAD</td>
<td>64</td>
<td>8,192</td>
<td>52.6</td>
<td><strong>51.0</strong></td>
<td>47.7</td>
</tr>
<tr>
<td>VLAD</td>
<td>256</td>
<td>32,768</td>
<td><strong>57.5</strong></td>
<td>50.8</td>
<td>47.6</td>
</tr>
</tbody>
</table>

- Observations:
  - VLAD better than BoF for a given descriptor size
    → comparable to Fisher kernels for these operating points
  - Choose a small D if output dimension D’ is small
**Product quantization for nearest neighbor search**

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

- Subvectors are quantized separately by quantizers $q(y) = [q_1(y_1) | \ldots | q_m(y_m)]$ 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
  - each subvector is quantized with 256 centroids $\rightarrow$ 8 bit
  - very large codebook $256^8 \sim 1.8 \times 10^{19}$

\[\begin{array}{cccccccc}
y_1 & y_2 & y_3 & y_4 & y_5 & y_6 & y_7 & y_8 \\
\end{array}\]

256 centroids

\[\begin{array}{cccccccc}
q_1 & q_2 & q_3 & q_4 & q_5 & q_6 & q_7 & q_8 \\
\end{array}\]

8 bits

$\Rightarrow$ 8 subvectors x 8 bits = 64-bit quantization index
Joint optimization of VLAD and dimension reduction-indexing

- For VLAD
  - The larger $k$, the better the raw search performance
  - But large $k$ produce large vectors, that are harder to index

- Optimization of the vocabulary size
  - Fixed output size (in bytes)
  - $D'$ computed from $k$ via the joint optimization of reduction/indexing
  - Only $k$ has to be set

  ➔ end-to-end parameter optimization
Results on the Holidays dataset with various quantization parameters

![Graph showing mAP and number of bytes for different ADC parameters (8x8, 16x8, 32x10, 128x10) and various values of VLAD k (16, 64, 256) and miniBOF ([8]).]
Results on standard datasets

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

<table>
<thead>
<tr>
<th>Method</th>
<th>bytes</th>
<th>UKB</th>
<th>Holidays</th>
</tr>
</thead>
<tbody>
<tr>
<td>BoF, k=20,000</td>
<td>10K</td>
<td>2.92</td>
<td>44.6</td>
</tr>
<tr>
<td>BoF, k=200,000</td>
<td>12K</td>
<td>3.06</td>
<td>54.9</td>
</tr>
<tr>
<td>miniBOF</td>
<td>20</td>
<td>2.07</td>
<td>25.5</td>
</tr>
<tr>
<td>miniBOF</td>
<td>160</td>
<td>2.72</td>
<td>40.3</td>
</tr>
<tr>
<td>VLAD k=16, ADC 16 x 8</td>
<td>16</td>
<td>2.88</td>
<td>46.0</td>
</tr>
<tr>
<td>VLAD k=64, ADC 32 x10</td>
<td>40</td>
<td>3.10</td>
<td>49.5</td>
</tr>
</tbody>
</table>

\(D' = 64 \text{ for } k=16 \text{ and } D' = 96 \text{ for } k=64\)

ADC (subvectors) \(x\) (bits to encode each subvector)

\[\text{miniBOF: “Packing Bag-of-Features”, ICCV’09}\]
Large scale experiments (10 million images)

- Exhaustive search of VLADs, $D’=64$
  - 4.77s

- With the product quantizer
  - Exhaustive search with ADC: 0.29s
  - Non-exhaustive search with IVFADC: 0.014s

IVFADC   -- Combination with an inverted file
Large scale experiments (10 million images)

Timings

- ADC: 0.286s
- IVFADC: 0.014s
- SH ≈ 0.267s

Database size: Holidays+images from Flickr