Instance level recognition IV: Very large databases

Cordelia Schmid
LEAR – INRIA Grenoble
Visual search

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 scale object/scene recognition

- 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
  ⇒ 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 return 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

    against

  – **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 $k$
20K visual word: false matches
200K visual word: good matches missed
Problem with bag-of-features

- The intrinsic matching scheme performed by BOF is weak
  - for a “small” visual dictionary: too many false matches
  - for a “large” visual dictionary: many 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

- Possible solutions
  - Soft assignment [Philbin et al. CVPR’08]
  - Additional short codes [Jegou et al. ECCV’08]
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

$$f_{HE}(x, y) = \begin{cases} 
    (\text{tf-idf}(q(x)))^2 & \text{if } q(x) = q(y) \\
    0 & \text{and } h((b(x), b(y))) \leq h_t \\
    h(a,b) & \text{Hamming distance}
\end{cases}$$
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 than 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 $z(x) = (z_1, \ldots z_{d_b})$
- $b_i(x) = 1$ 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
Re-ranking based on geometric verification

- works very well
- but performed on a short-list only (typically, 1000 images)
  → for very large datasets, the number of distracting images is so high
    that relevant images are not even short-listed!
  → weak geometry
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

  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 subspaces are shown 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
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

- Query
- Base 1
- Base 2
- Base 3
- Base 4
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 with lines for baseline, WGC, HE, WGC+HE, and re-ranking.]

<table>
<thead>
<tr>
<th>Average query time (4 CPU cores)</th>
<th></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
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></th>
<th>dataset distractors</th>
<th>Oxford</th>
<th>Kentucky</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>100K</td>
</tr>
<tr>
<td>soft assignment</td>
<td>0.493</td>
<td>0.343</td>
<td></td>
</tr>
<tr>
<td>ours</td>
<td>0.615</td>
<td>0.516</td>
<td></td>
</tr>
<tr>
<td>soft + geometrical re-ranking</td>
<td>0.598</td>
<td>0.480</td>
<td></td>
</tr>
<tr>
<td>ours + geometrical re-ranking</td>
<td>0.667</td>
<td>0.591</td>
<td></td>
</tr>
<tr>
<td>soft + query expansion</td>
<td>0.718</td>
<td>0.605</td>
<td></td>
</tr>
<tr>
<td>ours + query expansion</td>
<td>0.747</td>
<td>0.687</td>
<td></td>
</tr>
<tr>
<td>hierarchical vocabulary</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CDM</td>
<td>3.61</td>
<td>2.93</td>
<td></td>
</tr>
<tr>
<td>ours</td>
<td>3.42</td>
<td>3.10</td>
<td></td>
</tr>
<tr>
<td>ours + geometrical re-ranking</td>
<td>3.55</td>
<td>3.40</td>
<td></td>
</tr>
</tbody>
</table>

On-line demonstration

Demo at http://bigimbaz.inrialpes.fr
Towards large-scale image search

- BOF+inverted file can handle up to ~10 millions images
  - with a limited number of descriptors per image → RAM: 40GB
  - search: 2 seconds

- Web-scale = billions of images
  - with 100 M per machine → search: 20 seconds, RAM: 400 GB
  - not tractable

- Solution: represent each image by one compressed vector
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

sparse frequency vector

Centroids (visual words)

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
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]
Aggregating local descriptors

• Set of n local descriptors $\rightarrow$ 1 vector

• Popular approach: bag of features, often with SIFT features

• Recently improved aggregation schemes
  – Fisher vector [Perronnin & Dance ’07]
  – VLAD descriptor [Jegou, Douze, Schmid, Perez ‘10]
  – Supervector [Zhou et al. ‘10]
  – Sparse coding [Wang et al. ’10, Boureau et al.’10]

• Use in very large-scale retrieval and classification
Aggregating local descriptors

- Most popular approach: BoF representation [Sivic & Zisserman 03]
  - sparse vector
  - highly dimensional
  → significant dimensionality reduction introduces loss

- Vector of locally aggregated descriptors (VLAD) [Jegou et al. 10]
  - non sparse vector
  - fast to compute
  - excellent results with a small vector dimensionality

- Fisher vector [Perronnin & Dance 07]
  - probabilistic version of VLAD
  - initially used for image classification
  - comparable or improved performance over VLAD for image retrieval
VLAD: vector of locally aggregated descriptors

- Determine 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 square-root + L2-normalized

- Alternative: Fisher vector

[Jegou, Douze, Schmid, Perez, CVPR’10]
VLADs for corresponding images

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

- good coincidence of energy & orientations
**Fisher vector**

- Use a Gaussian Mixture Model as vocabulary
- Statistical measure of the descriptors of the image w.r.t the GMM
- Derivative of likelihood w.r.t. GMM parameters

GMM parameters:
- $w_i$ weight
- $\mu_i$ mean
- $\sigma_i$ co-variance (diagonal)

Translated cluster $\rightarrow$
large derivative on $\mu_i$ for this component

[Translated cluster $\rightarrow$
large derivative on $\mu_i$ for this component]

[Perronnin & Dance 07]
Fisher vector

FV formulas:

\[
G_{\mu,i}^X = \frac{1}{T \sqrt{w_i}} \sum_{t=1}^{T} \gamma_t(i) \left( \frac{x_t - \mu_i}{\sigma_i} \right)
\]

\[
G_{\sigma,i}^X = \frac{1}{T \sqrt{2w_i}} \sum_{t=1}^{T} \gamma_t(i) \left[ \frac{(x_t - \mu_i)^2}{\sigma_i^2} - 1 \right]
\]

\[\gamma_t(i) = \text{soft-assignment of patch } x_t \text{ to Gaussian } i\]

Fisher Vector = concatenation of per-Gaussian gradient vectors

For image retrieval in our experiments:
- only deviation wrt mean, dim: K*D [K number of Gaussians, D dim of descriptor]
- variance does not improve for comparable vector length
We compare Fisher, VLAD and BoF on INRIA Holidays Dataset (mAP %)

Dimension is reduced to D’ dimensions with PCA

- Fisher, VLAD better than BoF for a given descriptor size
- Choose a small D if output dimension D’ is small
- Performance of GIST not competitive

[Jegou, Perronnin, Douze, Sanchez, Perez, Schmid, PAMI’12]
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

Image representation
VLAD / Fisher

PCA + PQ codes

(Non) – exhaustive search
Optimizing the dimension reduction and quantization together

- Fisher vectors undergoes two approximations
  - mean square error from PCA projection
  - mean square error from quantization

- Given $k$ and bytes/image, choose $D'$ minimizing their sum

Results on Holidays dataset:
- there exists an optimal $D'$
- 16 byte best results for $k=64$
- 320 byte best results for $k=256$
Results on the Holidays dataset with various quantization parameters
Product quantization for nearest neighbor search

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

- Subvectors are quantized separately by quantizers $q(y) = [q_1(y_1) \mid \ldots \mid q_m(y_m)]$
  
  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 -> 8 bit
  - very large codebook $256^8 \sim 1.8 \times 10^{19}$

$\Rightarrow$ 8 subvectors x 8 bits = 64-bit quantization index

[Jegou, Douze, Schmid, PAMI’11]
Comparison to the state of the art

<table>
<thead>
<tr>
<th>Method</th>
<th>bytes</th>
<th>UKB</th>
<th>Holidays</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOW, $K=20,000$</td>
<td>10364</td>
<td>2.87</td>
<td>43.7</td>
</tr>
<tr>
<td>BOW, $K=200,000$</td>
<td>12886</td>
<td>2.81</td>
<td>54.0</td>
</tr>
<tr>
<td>miniBOF [12]</td>
<td>20</td>
<td>2.07</td>
<td>25.5</td>
</tr>
<tr>
<td></td>
<td>80</td>
<td>2.72</td>
<td>40.3</td>
</tr>
<tr>
<td></td>
<td>160</td>
<td>2.83</td>
<td>42.6</td>
</tr>
<tr>
<td>FV $K=64$, spectral hashing 128 bits</td>
<td>16</td>
<td>2.57</td>
<td>39.4</td>
</tr>
<tr>
<td>VLAD, $K=16$, ADC 16×8 [23]</td>
<td>16</td>
<td>2.88</td>
<td>46.0</td>
</tr>
<tr>
<td>VLAD, $K=64$, ADC 32×10 [23]</td>
<td>40</td>
<td>3.10</td>
<td>49.5</td>
</tr>
<tr>
<td>FV $K=8$, binarized [22]</td>
<td>65</td>
<td>2.79</td>
<td>46.0</td>
</tr>
<tr>
<td>FV $K=64$, binarized [22]</td>
<td>520</td>
<td>3.21</td>
<td>57.4</td>
</tr>
<tr>
<td>FV $K=64$, ADC 16×8 ($D′=96$)</td>
<td>16</td>
<td>3.10</td>
<td>50.6</td>
</tr>
<tr>
<td>FV $K=256$, ADC 256×10 ($D′=2048$)</td>
<td>320</td>
<td>3.47</td>
<td>63.4</td>
</tr>
</tbody>
</table>

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
IVFADC: 0.02s
Conclusion

- Competitive search accuracy with a few dozen bytes per indexed image

- Tested on 220 million video frames
  - extrapolation for 1 billion images: 20GB RAM, query < 1s on 8 cores

- Code on-line available Software for Fisher computation and PQ-codes
  - http://lear.inrialpes.fr/software