From similarity to scalability in content-based image and video retrieval

Matthieu Cord
LIP6, Univ. UPMC-PARIS VI, FRANCE
matthieu.cord@lip6.fr

December 13, 2011
# Introduction

- Content based image and video retrieval systems
- Kernel design and data representation for actor retrieval
- Approximate k-NN for fast similarity approximation
- Optimization active learning on large scale databases
Content Based retrieval system:
Visual data representation

Image features
- Pixels, Points of Interest, RoI, Regions, Blobs
Indexing process

1. Feature extraction
   ⇒ Regions, points of interest, ...

2. Descriptors
   ⇒ Color, texture, SIFT, ...

3. Bags of feature $B_i = \{b_{ri}\}_r$
   ⇒ $b_{ri}$: region/poi $r$ of image $i$

4. Similarity $S(B_i, B_j)$ ?
SoA model

Bag of Visual Words (BoW)

clustering

keypoint feature space

visual word vocabulary

BoW histogram

Credit: Prof. Shih-Fu Chang
Similarity $S(B_i, B_j)$ using Visual Dictionary

- 2 steps:
  1. Explicit mapping of $B_i$ into a vector space
  2. Similarity on vectors
- Computation of the Visual Dictionary over the database
- Strategies to cluster all the feature data, like k-means

Image index: distribution on the Visual dictionary
Content-based video shot retrieval
Example of search by similarity
Example of search by similarity
Online learning

Optimisation of the ranking using $A$
Online learning

Optimisation of the ranking using $A$

- similarity function updating $f(x)$
Online learning

Optimisation of the ranking using $A$

- similarity function updating $f(x)$
- classification scheme
Online learning

Optimisation of the ranking using $A$

- similarity function updating $f(x)$
- classification scheme

$A$ Enhancement

- Show the top rank examples
Online learning

Optimisation of the ranking using $\mathcal{A}$

- similarity function updating $f(x)$
- classification scheme

$\mathcal{A}$ Enhancement

- Show the top rank examples
- or show the best ones to enhance ranking: Active learning strategy
Online learning
Online learning
Content-based Video Retrieval: Query

Figure: Query
Content-based Video Retrieval: Result

Figure: Top ranked
Content-based Video Retrieval: Result

Figure: Bottom ranked
## Introduction

One step further to track a needle in a haystack

- Dictionary-based approaches $\Rightarrow$ Vectors as index
- Other index ? more discriminant ?

### Similarity functions $S(B_i, B_j)$

- Alternatives to dictionary-based approaches:
  1. Copy detection approach:
     - Signature $= B_i$ the set of vectors $b_{ik}$
     - Similarity retrieval using NN search and voting systems
  2. Kernels on bags
Kernel as similarities

Definition of a kernel function $K : \mathcal{X} \times \mathcal{X} \to \mathcal{R}$

$K$ is a kernel iff $\exists \Phi | \forall (x, y), K(x, y) = <\Phi(x), \Phi(y)>$ with $\Phi$ an injection into a Hilbert $\mathcal{H}$ space (explicit or not)

$\Phi : \mathcal{X} \to \mathcal{H}$

$x \to \Phi(x)$

Advantages:

- Integration with Machine Learning techniques (Neural networks, SVM, ...)
- Allow to build similarities on non vector input spaces
Kernel functions for bags of vectors

Framework:

Soft maximum kernel function [Shawe-Taylor book02]:

\[
K_{\text{softmax}}(B_i, B_j) = \sum_{b_{ri} \in B_i} \sum_{b_{sj} \in B_j} k(b_{ri}, b_{sj})
\]

- Nice property:
  \( k \) is a kernel function \( \Rightarrow K_{\text{softmax}} \) is a kernel function
- Not enough discriminant?
Kernel on Bags of Features

Improvement [LyuCVPR 05, CordCIVR 07]

\[
K(B_i, B_j) \triangleq \left( \sum_r \sum_s \left( k(b_{ri}, b_{sj}) \right)^q \right)^{\frac{1}{q}}
\]

For each couple of Pol \((b_{ri}, b_{sj})\)

Minor Kernel: \(k(b_{ri}, b_{sj})\)
Classifier

Training set
- \( A = \{(x_i, y_i)_{i=1,N} \mid y_i \neq 0\} \)
- \( U = \{(x_i, y_i)_{i=1,N} \mid y_i = 0\} \)

SVM:
- Minimize \( \frac{||w||^2}{2} \) s.t. \( y_i (\langle w, \Phi(x_i) \rangle + b) \geq 1, \forall i \in [1, N] \)
- Classifier: \( f_A(x) = \langle w, \Phi(x) \rangle + b \)
Example of retrieval session
Example of retrieval session
Example of retrieval session
Example of retrieval session
Example of retrieval session
Example of retrieval session
Example of retrieval session
Any Extension?
Extension: integration of spatial constraints

Credit: Dr. S. Lazebnik
Extension: integration of spatial constraints
Extension: integration of spatial constraints
Extension: integration of spatial constraints

Kernel function on bag $\mathcal{P}_i$ of bags of pairs $P_{vi}$:

$$K_{pairs}(\mathcal{P}_i, \mathcal{P}_j) = \left( \sum_{P_{vi} \in \mathcal{P}_i} \sum_{P_{wj} \in \mathcal{P}_j} K_{single}(P_{vi}, P_{wj})^q \right)^{\frac{1}{q}}$$

For each region $b_{ri}$, we build 3 pairs with its 3 closest regions. $K_{pairs}$ may be connected to kernel on graphs [Kashima]
Evaluation

RETIN Active learning with 5 labels/feedback, 10 feedbacks.
Extension (2): application to video actor retrieval

Video object extraction and description

- RoI = face tubes
  - Frame face detection
  - Face region grouping in shots

Example of a tube:
Data representation

Temporal stability of SIFT points: Intra-tube chain tracking

SIFT points along the same chain in same color (scale and orientation of ellipses representing the scale and orientation of SIFT)
**Representation optimization**

- Intra-tube chain tracking
- Consistent chain extraction:

\[ T_i : \text{a set of chains } C_{ri} \text{ of SIFT descriptors:} \]

\[ T_i = \{ C_{1i}, \ldots, C_{ki} \} \text{ and } C_{ri} = \{ SIFT_{1ri}, \ldots, SIFT_{pri} \} \]
Kernel design for actor retrieval

The major kernel on tubes is then defined as:

$$K'_{\text{pow}}(T_i, T_j) = \left( \sum_{r} \sum_{s} \frac{|C_{ri}|}{\sqrt{|T_i|}} \frac{|C_{sj}|}{\sqrt{|T_j|}} k'(C_{ri}, C_{sj})^{q} \right)^{\frac{1}{q}}$$

(1)

with the following minor kernel on chains:

$$k'(C_{ri}, C_{sj}) = \exp \left( -\frac{1}{2\sigma^2} \chi^2 \left( \overline{C_{ri}}, \overline{C_{sj}} \right) \right) e^{-\frac{(\bar{x}_{ri} - \bar{x}_{sj})^2 + (\bar{y}_{ri} - \bar{y}_{sj})^2}{2\sigma^2}}$$

(2)
Kernel design
And so what?
Kernel design

And so what? Actually, all the work is done!
Kernel design

And so what? Actually, all the work is done!
It is now RETIN compatible: online actor retrieval
Experiments on a french movie "L’esquive"
Experiments on a french movie "L'esquive"
Experiments on a french movie "L’esquive"
Experiments for multi-class actor retrieval on videos "Buffy" [Zisserman&Sivic database]
Experiments for multi-class actor retrieval on videos “Buffy” [Zisserman & Sivic database]
Experiments for multi-class actor retrieval on videos "Buffy" [Zisserman&Sivic database]
Experiments on system robustness:
Experiments on system generalization:
Experiments on system generalization:
Experiments on system generalization:
Experiments on system generalization:
Introduction to fast retrieval scheme

Computation optimization pb

Control of search complexity when the size of the database becomes huge
Problem even more crucial when the number and the size of the descriptors increase

Computational pb of similarity functions $S(B_i, B_j)$

- All the Alternatives to dictionary-based approaches are time consuming
  - 1. Copy detection approach:
     - Signature = $B_i$ the set of vectors $b_{ik}$
     - Similarity retrieval using NN search and voting systems
  - 2. Kernels on bags

⇒ Need optimization scheme!
Copy Detection scheme [Lowe04]

Image database

Image query

Description: SIFT

Knn search

Fast selection of images of interest by VOTE

Result

Geometric consistency
Copy Detection scheme

Optimization scheme
- **Fast NN search (1)** to quickly retrieve near duplicate or most similar images (TOPN) to a given query
- Need to structure the database ⇒ indexing scheme

Database indexing schemes
- Classical indexing schemes fail with high dimensional data
- Approximate search approaches
  - Tree techniques (BBFirst Kd-tree, ...)
  - Projections (NV Tree, VA files, Space Filing Curves, **Locality Sensitive Hashing** )
Implementing Locality Sensitive Hashing

\[ f_i() \]: function of the hash table \( i \) and \( h_{a,c}() \) the hash function:
\[
 f_i(b) = (h_{a_1,c_1}^i(b), \ldots, h_{a_k,c_k}^i(b))
\]
\[
 h_{a,c}(b) = \left\lfloor \frac{a \cdot b + c}{w} \right\rfloor
\]
Implementing Locality Sensitive Hashing

[datar 2004]

\( f_i() \): function of the hash table \( i \) and \( h_{a,c}() \) the hash function:

\[
f_i(b) = (h_{a_1,c_1}^i(b), \ldots, h_{a_k,c_k}^i(b))
\]

\[
h_{a,c}(b) = \left\lfloor \frac{a \cdot b + c}{w} \right\rfloor
\]
Under conditions, the $h_{a,c}()$ family is LSH:

\[
\mathcal{H} = \{ h : S \rightarrow U^1 \} \text{ is } (R, \epsilon, p_1, p_2) \text{ Locality-Sensitive if, } \forall (A, B, Q):
\]

\[
A \in B(Q, R) \Rightarrow \Pr_{\mathcal{H}}[h(Q) = h(A)] \geq p_1, \quad (3)
\]

\[
B \notin B(Q, (1+\epsilon)R) \Rightarrow \Pr_{\mathcal{H}}[h(Q) = h(B)] \leq p_2. \quad (4)
\]
Implementing Locality Sensitive Hashing (2)

Implementation depending on the representation space

- in Hamming space $\mathcal{H}^d$ or $\mathbb{Z}^d$: LSH random permutation [Indyk98]
- in $\mathcal{R}^d$ normalized: cosine similarity [Charikar02]
- in $\mathcal{R}^d$: distance L2 or L1
  - [Gionis99] projection of $\mathcal{R}^d$ in $\mathcal{H}^d$ + [Indyk98]
  - [Datar04] splitting along 1 dimension
  - [Lv07] (multi-probe) extension of [Datar04]
  - [Andoni06] 24 lattice, [Jegou08] E8 lattice

Implementation available for a vector representation of images and distances aforementioned

Extension to other similarities and to non vector spaces?
Fast kernel on Bags Pyramid Match Hashing

[Grauman07]

- Each image is described by a bag of SIFT
- Injection with a function $\Phi$ in a space of high dimension
- The injection is explicit:
  - Projection into SIFT space
  - Multi-scale grid
  - Projection into Hamming space
  ⇒ Each image becomes a unique Vector
- An explicit induced space allows to use LSH
- The resulting kernel allows to get a similarity from the matching between PoIs (Points of Interest) of the 2 images
LSH on other kernels?

- Pyramid Match Hashing ⇒ good performances
- BUT cannot be extended to kernels where $\Phi$ is not explicit
- If the class of kernels is different:
  
  \[
  \text{ex: } K(B_i, B_j) = \left( \sum_r \sum_s (k(b_{ri}, b_{sj}))^q \right)^{\frac{1}{q}}
  \]

- Can we speed up the computation?
Approx. scheme [ICPR 2008]

- **Model:**
  - consider each image as a bag of unordered features
  - similarity: class of kernels on bags

\[
K(B_i, B_j) = \left( \sum_r \sum_s \left( k(b_{ri}, b_{sj}) \right)^q \right)^{\frac{1}{q}}
\]

- **Objective:**
  - fast computation of the topN from a ranking of the database with similarity kernel \( K \)
  - ⇒ decrease the kernel computational complexity

- **Principle (inspired from copy detection):**
  1. (2) Quick selection of database subset (LSH scheme)
  2. (3) Kernel computation only on this relevant subset
  - ⇒ resulting scheme is an approximation of the exact similarity ranking of the whole database
Principle for fast retrieval

Image query

Image database

Description: SIFT

Knn search (1)

Fast selection of images of interest by VOTE (2)

Evaluation of kernel similarity K + ranking (3)

Result
Pre-processing: Hashing of the database

For each image $B_i$

For each attribute $b_{si}$

For each hash table $k$

- selection of a bucket with hashing function: $f_k(b_{si})$
- put $b_{si}$ in the selected bucket

Locality Sensitive Hashing [datar 2004]

Notation: $f_i()$: function of the hash table $i$

$$f_i(b) = \left( h_{a_1,c_1}(b), \ldots, h_{a_k,c_k}(b) \right)$$

$h_{a,c}()$: hash function

$$h_{a,c}(b) = \left\lfloor \frac{a \cdot b + c}{w} \right\rfloor$$
Retrieval Algorithm

Database

Hashing of the database

All the images of the database are distributed into different Buckets of table 1

Hash table 1

...
Retrieval Algorithm

Database

Hashing of the database

All the images of the database are distributed into different Buckets of table 1

Hash table 1

Hash table N

Query

(1) Knn search:
selection of Buckets containing nearest Pols to query Pols
Retrieval Algorithm

Database

Hashing of the database

All the images of the database are distributed into different Buckets of table i

Hash table 1

Hash table N

Query

(2) Selection by Vote
Selection of Images containing at least v PoIs belonging to selected Buckets in (1)

(1) Knn search

s

s

s
Retrieval Algorithm

- Database

- Hashing of the database

- All the images of the database are distributed into different Buckets of Table 1

- Query

- Hash table 1

- Hash table N

- TOP N

- \( K(B_i, B_j) \triangleq \sum_{b_{ri} \in B_i} \sum_{b_{sj} \in B_j} k(b_{ri}, b_{sj}) \)

- \( k(b_{ri}, b_{sj}) = \langle \phi(b_{ri}), \phi(b_{sj}) \rangle \)

- (3) Ranking

- (2) Selection by Vote

- (1) Knn search

- Vote
Experiments

- VOC2006 database: 5,304 images
- Indexing: $\sim 100$ PoI per Image
  - MSER region detectors
  - SIFT descriptors
- Variance normalization

- E2LSH parameters
  - radii between 150 and 250 (4.0 and 6.0 after normalization)
  - $L = 50$ hash tables
  - $K = 20$ projections

- Image selection VS whole database
  - TOP100 deterioration
  - computational time reduction
Example

Fast Selection + Ranking by Vote

372 / 5304 images (7.1% of the database)
Example

Fast selection

372 / 5304 images (7.1% of the database)

Ranking of the selection by Similarity K
Selection ranking

Ground truth for K : Ranking of the whole database

96% of images of TOP100 obtained from our fast selection are identical to TOP100 on the whole database
Results

Accuracy of TOP100 for various radii of search around query points

<table>
<thead>
<tr>
<th>radius</th>
<th>4.0</th>
<th>5.0</th>
<th>5.2</th>
<th>6.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>factor</td>
<td>122.17</td>
<td>14.85</td>
<td>10.03</td>
<td>3.19</td>
</tr>
</tbody>
</table>

Percentage of selected images for various radii

Speed improvement factor regarding the true search
Discussion

Fast similarity scheme

- Fast similarity search working with non explicit kernels and with all fast knn search methods
- Good trade-off between Precision and Speed for $R=5.2$: 10 time faster and median precision 99%

But ...

TOPN not good enough for category retrieval
Discussion

Fast similarity scheme
- Fast similarity search working with non explicit kernels and with all fast knn search methods
- Good trade-off between Precision and Speed for R=5.2: 10 time faster and median precision 99%

But ...
- TOPN not good enough for category retrieval
- Is it RETIN compatible?
Discussion

**Fast similarity scheme**
- Fast similarity search working with non explicit kernels and with all fast knn search methods
- Good trade-off between Precision and Speed for R=5.2: 10 time faster and median precision 99%

**But ...**
- TOPN not good enough for category retrieval
- Is it RETIN compatible?
- Need adaptation for online category learning
Can we decrease the complexity of Active Learning using similar strategy than ICPR08?
Can we decrease the complexity of Active Learning using similar strategy than ICPR08? Not straightforward to combined fast similarity schemes with online/Active Learning.
Can we decrease the complexity of Active Learning using similar strategy than ICPR08? Not straightforward to combine fast similarity schemes with online/Active Learning. Active Learning schemes: at least a complexity linear regarding the size of the database. ⇒ impracticable for large database.
Active Learning have 2 problems of scalability. The database have to be sorted to extract:

- the relevant images for intermediate results.
- the most uncertain images for annotation strategy.
Active Learning have 2 problems of scalability. The database have to be sorted to extract:

- the relevant images for intermediate results.
- the most uncertain images for annotation strategy.
Active Learning have 2 problems of scalability. The database have to be sorted to extract:

- the relevant images for intermediate results.
- the most uncertain images for annotation strategy.
Active Learning have 2 problems of scalability. The database have to be sorted to extract:

- the relevant images for intermediate results.
- the most uncertain images for annotation strategy.
Active Learning have 2 problems of scalability. The database have to be sorted to extract:

- the relevant images for intermediate results.
- the most uncertain images for annotation strategy.

These scalability problems occur at each feedback loop.
Active Learning have 2 problems of scalability. The database have to be sorted to extract:

- the relevant images for intermediate results.
- the most uncertain images for annotation strategy.

We tackle these problems by considering only a relevant subset $S$ instead of $U$. 
Approx. scheme [icip09]

Unlabelled dataset \( \mathcal{U} = \{(x_i, y_i)_{i=1,n} \mid y_i = 0\} \)

Query image

Sort \( f_A(x_i) \)

Retrieved images
Approx. scheme [icip09]
Approx. scheme [icip09]

Unlabelled dataset
\[ \mathcal{U} = \{(x_i, y_i)_{i=1,n} \mid y_i = 0\} \]

Query image

LSH

Selection

Sort \( f_A(x_i) \)

\[ \arg \min_{i \in S} R_{\text{text}}(f_A \cup \{(x_i, s(x_i))\}) \]

Labeled dataset

Retrieved images

Annotation Strategy

User labels
Approx. scheme [icip09]

Unlabelled dataset $U = \{ (x_i, y_i) | y_i = 0 \}$

LSH $\rightarrow$ Selection

Images positively annotated

Sort $(f(x_i))$ $i \in S$

Training $\rightarrow f_{A_t}$

Labeled dataset $A_t$ $\rightarrow$ User labels

Annotation Strategy

Retrieved images $\rightarrow i^* = \arg \min_{i \in S} R_{test}(f_{A_t \cup \{ (x_i, y_i) \}})$
Each image is represented by a 192-dimension vector: 64 chrominances CIE Lab and 2 histograms of 64 textures from Gabor filters.
Experiments

Performances are evaluated with Mean Average Precision of the TOP500, i.e., the sum of the Precision/Recall curve for the first 500 images retrieved.

E2LSH parameters are $R = 16.0$ and $L = 30$ hash tables of $K = 20$ projections.
Experiments

Average time of an interactive search function of the number of iteration
Conclusion

Next-Generation Visual Search

Human Vision  Internet Vision  Computer Vision/
Machine Learning

Content-based image
video search engines

Credit: Prof. Shih-Fu Chang
People

Matthieu Cord, Nicolas Thome
LIP6, Univ. UPMC-PARIS VI
matthieu.cord@lip6.fr

Frédéric Precioso
ETIS UCP
precioso@ensea.fr

- Shuji Zhao (PhD student),
- David Gorisse (PhD student),
- David Picard (PostDoc LIP6)

http://webia.lip6.fr/~cord/
http://frederic.precioso.free.fr
<table>
<thead>
<tr>
<th>Bibliography pieces</th>
</tr>
</thead>
</table>