Module II: Multimedia Data Mining

Laurea Magistrale in Ingegneria Informatica
University of Bologna

Complex Retrieval Models and MM DBMS

Home page: http://www-db.disi.unibo.it/courses/DM/
Electronic version: 2.02.ComplexRetrievalModels.pdf

Outline

- Region-based image retrieval
  - The Windsurf framework
- “Local” feature-based retrieval
  - SIFT and SURF
- MM DBMS
Focusing on real systems

- From the previous lessons we know that the set of images representing the result of the a nearest neighbor (NN) query mainly depends on the selected
  - low-level features, similarity distance functions, and the way objects are ranked with respect to the query

- Our simplified hypothesis was to consider “global” features
  - i.e., no image segmentation, no local features

- Let’s move now to discover what happens and how things can become complex in a “real” image retrieval system… 😊

The “region-based” image retrieval approach

- DB population time:
  - Pre-process images to segment them into regions
  - Represent regions as vectors of features

- Query time:
  - Compare query regions to DB regions
  - Assess similarity between images by combining similarity between regions
Differences between RBIR and CBIR

- In RBIR:
  - an image is a complex object
    - e.g., a set of regions, each one represented by means of a feature vector
  - the “image comparison” process is a more complex task than for CBIR
- Here we need to solve the matching problem among region feature vectors and define the aggregation modality applied to the region similarity scores in order to assess the query result (i.e., the ranking model)

- The similarity between the query image and a DB image is assessed by taking into account the similarity between matched regions

Matching types

- The matching type defines which set of constraints applies when the component regions of the query image $I_q = \{R_{q1}, \ldots, R_{qn}\}$ have to be matched to the component regions of a DB image $I = \{R_1, \ldots, R_m\}$

- Two relevant cases for matching types are:
  - the one-to-one (1–1) and
  - the many-to-many (n–m)
Matching 1-1

- With 1–1 matching, each region of image $I_q$ is associated to at most one region of $I$, and vice versa
- Each matching has to be complete
  - i.e., if $n > m$ (respectively, $n < m$) then only $n - m$ (resp., $m - n$) regions of $I_q$ (resp., $I$) have to remain unmatched

Matching n-m

- With $n-m$ matching, each region of $I_q$ can be associated to many regions of $I$, and vice versa
- This, however, could lead to undesired (pathological) results
  - For example, a single region of the query could be matched to all regions of a DB image
    - This has been termed “the two tigers problem” in literature since it arises when a single region (a tiger) of the query image is very similar to multiple regions of a DB image (e.g., containing two tigers)

- A special case of $n-m$ matching that avoids this problem is the Earth Mover’s Distance (EMD) matching, where variable-sized pieces of regions are allowed to be matched (the size of each region defines the maximum amount for its matching)
  - This contrasts with the 1–1 matching, where elements of fixed size (i.e., regions) are matched individually
Ranking model (1)

- The *k-NN ranking model* requires to define the *image similarity* of a DB image $I$ with respect to a query image $I_q$, $s_I(I_q; I)$
  - by means of a *numerical scoring function* ($sf$), such as the *average*, which *aggregates the region similarity scores* into a *global similarity value*
- Among all valid matchings that satisfy the constraints of the specific matching type, the rationale is to select the one that *maximizes* the aggregated score
- This can be modeled as an *optimization problem* whose solution depends on the particular choice of the scoring function
  - For the most commonly used functions, efficient algorithms exist

- Examples:
  - when using the average function with the 1 – 1 matching type, the problem takes the form of the well-known *assignment problem*
  - while with the n–m (EMD) matching type (see above), this corresponds to the *transportation problem*

I. Bartolini

Ranking model (2)

- Given the query image $I_q$ and two DB images $I_1$ and $I_2$, image $I_1$ will be considered more similar than $I_2$ to $I_q$ iff
  \[ s_I(I_q; I_1) > s_I(I_q; I_2) \]
- In such a way it is possible to *linearly order* DB images and return to the user only the *$k$ highest scored ones*
- Limitation of the k-NN ranking model:
  - the choice of a particular scoring function influences the final result, i.e., different scoring functions will likely yield different results

- Alternative ranking models based on the notion of *preference relation* has been proposed in literature
  - e.g., “Best Match Only” (BMO) model
  - out of the scope of this course
The Windsurf case study (1)
[ABP99, BCP00, BP00, BC03, Bar09a, BCP+09, BCP10, BPS11]

- **Windsurf** ([Wavelet-Based Indexing of Images Using Regions Fragmentation](http://www-db.deis.unibo.it/Windsurf/index.html)) is a general framework for efficiently processing CBIR

- With particular emphasis to the region-based paradigm (RBIR), **Windsurf provides an environment where different alternatives of the paradigm can be implemented**
  - This allows such implementations to be compared on a fair basis, from the points of view of both effectiveness and efficiency

The Windsurf case study (2)
[ABP99, BCP00, BP00, BC03, Bar09a, BCP+09, BCP10, BPS11]

- The framework offers a number of appealing features:
  - **extensibility and personalization:**
    - different types of low-level image representation, image segmentation, feature extraction, and region comparison criteria can coexist and be compared
  - support of **different types of region matching** constraint
    - e.g., 1-1 and N-M
  - **efficient processing** of rank-based queries
  - **efficient processing** of BMO queries
  - handling of **different types of queries**
    - e.g., full vs. part-of queries
Windsurf: example

- Query requesting for the "Dome of St. Peter" and its corresponding results according to one of the implementations included in the framework,
  - i.e., the Windsurf system... will details very soon on it 😊

Windsurf: the framework (1)

- Tackles the problem of RBIR providing efficient techniques for the processing of queries

- Since the framework is extendible, one can exploit the query processing techniques in Windsurf by opportunely instantiating the relevant parametric portions of the framework

- Windsurf provides a Java library including classes that can be appropriately extended to realize the required RBIR model

  + one concrete instantiation of the framework, the Windsurf system
Windsurf: the framework (2)

- Four main modules:
  - *Query processor*, that is in charge of efficient query resolution
  - *Feature extractor*, dealing with low-level features extraction and image segmentation
  - *Persistence of raw-data and features* is guaranteed by means of abstract managers (i.e., *RD-manager* and *F-manager*):
    - concrete implementations of the managers include the use of text files and relational DBMSs
  - *Feature index* is built upon request and used for query solving:
    - the framework supports the use of different kinds of indices and includes the implementation of *M-tree*

Query processor

- Provides *sequential* and efficient *index-based* algorithms for the resolution of several types of queries:
  - full vs. part-of queries
  - k-NN vs. range vs. BMO queries

and is parametric with respect to three basic ingredients:

- the *distance function* used to compare image regions;
- the *matching type* specifying how regions of the images being compared have to be coupled (either 1-1 or n-m);
- the *aggregation function* combining distance values between matched regions into an overall image distance value

- Algorithms included in the library for RBIR are general enough to accommodate *other document types*!!
  - The only requirement is that the *document* can be spited into *sub-documents* (e.g., regions for images) and that *comparison between documents is performed using the three above described ingredients*
Feature extractor

- The Feature extractor is orthogonal to the Query processor
  - query processing techniques can be applied to any feature

- The Feature extractor is an abstract component of the framework
  - its concrete implementations are realized by properly extending framework classes dealing with
    - low-level image representation,
    - image segmentation, and
    - regions features extraction

Feature extractor: example

- The library includes a concrete instance of the Feature extractor, implementing the Windsurf system feature extractor
Windsurf: the software library

- The Java library primarily focuses on RBIR and has the ability to encompass different querying models.
- Such flexibility is provided by a number of different templates that can be appropriately instantiated in order to realize the particular retrieval model needed by the user/service/application.

- The library is released under the "QPL" license and is freely available for your own use only, for education and research purposes only.
  - For commercial purposes, it is necessary to require a differently licensed version of the library that does not contain the above restrictions.

Contributors: Ilaria Bartolini, Marco Patella, and Guido Stromei

Windsurf: the system

- The software library includes, as one of the possible instantiations of the framework, the Windsurf system.
- From the point of view of feature extraction, the Windsurf system is characterized by:
  - image representation based on the Discrete Wavelet Transform (DWT) filter in the HSV color space.
  - region segmentation based on the K-means algorithm, clustering wavelet coefficients according to the Mahalanobis distance.
  - region features include the centroid and the covariance matrix of the wavelet coefficients.
  - region comparison using the Bhattacharyya distance function on the regions representations.

- Windsurf system provides concrete implementations of all the query processing models and techniques included in the Windsurf framework.
The Windsurf system: detailed analysis

- **Windsurf**: Wavelet-Based Indexing of Images Using Regions Fragmentation
  - Discrete Wavelet Transform (DWT): extracts a set of features representing the image in the color-texture space
  - Clustering: fragments the image into a set of regions using wavelet coefficients
  - Similarity Features: used to compare regions

---

Discrete Wavelet Transform (DWT)

- **Haar** wavelet: simple and quick
- Each coefficient is defined by:
  - level DWT \( l \)
  - frequency sub-band \( B \)
  - color channels \( (H, S, V) \)

\[
\begin{align*}
  w_{l,B}^{j} & = (w_{0,B}^{j}, w_{1,B}^{j}, w_{2,B}^{j}) \\
  B & \in \{LL, LH, HL, HH\}
\end{align*}
\]
Clustering (1)

- **K-means algorithm** (3rd level and low frequency info)
  - Choose \( k \) initial centroids;
  - Associate each point to its nearest centroid;
  - Recompute centroids and repeat previous step;
  - Stop when solution does not change.

- **Mahalanobis distance**:

\[
\delta(w_i^{3,LL}, w_j^{3,LL})^2 = (w_i^{3,LL} - w_j^{3,LL})^T (C^{3,LL})^{-1} (w_i^{3,LL} - w_j^{3,LL})
\]

- Correlation between wavelet coefficients takes into account variations in color, i.e. texture
Clustering (2)

- Optimal value for $k$?
- Minimization of a validity function
  - Intra-cluster distance
  - Clusters’ size
  - Inter-cluster distance

Similarity features

- Region similarity with Bhattacharyya distance
  - Regions are ellipsoids in 37-D feature space (all frequencies info is used)
    - (3-D centroid + 6-D covariance matrix + 1-D region size)
  - Distance between regions’ centroids (color info)
  - Covariance matrices (texture info)

\[
 d_B(R_i, R_j)^2 = \frac{1}{2} \ln \left( \frac{C_{R_i}^{3:B} + C_{R_j}^{3:B}}{2 \left| \frac{1}{2} C_{R_i}^{3:B} \right| \cdot \left| \frac{1}{2} C_{R_j}^{3:B} \right|} \right) + \frac{1}{8} \left( \mu_{R_i}^B - \mu_{R_j}^B \right)^T \left( C_{R_i}^{3:B} + C_{R_j}^{3:B} \right)^{-1} \left( \mu_{R_i}^B - \mu_{R_j}^B \right)
\]
Image similarity

- Similarity between images is a function of similarities among "matched" regions
- How regions are "matched" can therefore strongly influence the result of a query:
  - 1-1 match - formulated as Assignment Problem
  - n-m match - formulated as Transportation Problem

- After computing image similarities between DB images and the image query, DB objects are ranked in decreasing order of similarities and returned as final result of the nearest neighbor (NN) query

Assignment problem

- Goal: “Find the optimal match where unit elements of fixed size are matched individually”

- Implemented with the Hungarian algorithm, maximizing a function that is monotonic in the similarity scores (e.g. average)

\[
\begin{array}{cccccc}
 & r_1 & r_2 & r_3 & r_4 & r_5 \\
q_1 & .52 & .17 & .41 & .16 & .29 \\
q_2 & .27 & .19 & .81 & .35 & .49 \\
q_3 & 1.0 & .11 & .27 & .24 & .29 \\
\end{array}
\]

\[
(0.52 + 0.17 + 1.0)/3 = 0.77
\]

\[
\begin{array}{cccccc}
 & r_1 & r_2 & r_3 & r_4 & r_5 \\
q_1 & .52 & .17 & .41 & .16 & .29 \\
q_2 & .27 & .19 & .81 & .35 & .49 \\
q_3 & 1.0 & .11 & .27 & .24 & .29 \\
\end{array}
\]

\[
(0.29 + 0.81 + 1.0)/3 = 0.7
\]
Transportation problem

- Goal: “Find the least expensive flow where variable-size pieces of “mass” are allowed to be moved together”

- Implemented with the Earth Mover’s Distance (EMD):
  - “Given two distributions (let’s see one as piles of earth and the second as a collection of holes), EMD measures the least amount of work needed to fill the holes with earth. A unit of work corresponds to transporting a unit of earth by a unit of ground distance”

Ranking images based on image similarities
Effectiveness comparison example

"Flowers" query  
Windsurf clusters  
Blobworld [CTB+99] clusters

Windsurf  
Blobworld

I. Bartolini  
Data Mining M  
31

Local feature-based retrieval

- Relevant implementations of the BoW model are
  - SIFT [Low99] and SURF [BET08]
- In both cases, the image is represented by means of a (large) set of local salient (or key) points such as corners, blobs, and T-conjunction

- Similarity between images is assessed by matching visual characteristics of their salient points based on Euclidean or quadratic distances
Scale-invariant feature transform (SIFT) [Low99]

- SIFT features are *local* and based on the appearance of the object at particular interest points.
- Interest points (also named *keypoints*) are detected as follows:
  - The image is convolved with Gaussian filters at different scales, and then the difference of successive Gaussian-blurred images are taken; *keypoints are then taken as maxima/minima of the difference of Gaussians that occur at multiple scales*.
    - Difference of Gaussian (DoG) filter
    - *Low contrast* keypoints and those located on edges are discarded.

SIFT: keypoint descriptor

- For each of the so extracted keypoints, information on *local oriented gradients* (using an histogram of 8-bins) based on blocks of 4x4 pixels are maintained.
- Each keypoint is so represented by a vector of 128 (4x4x8) elements.
SIFT: properties

SIFT features are:
- invariant to image scale and rotation
- robust to changes in illumination, noise, and minor changes in viewpoint
- highly distinctive, relatively easy to extract and allow for correct object identification with low probability of mismatch
- relatively easy to match against a (large) database of local features

**BUT!**
- the high dimensionality can be an issue
  - E.g., 800/1000 keypoints, each represented by a 128-D vector
  - approximate solutions to the Nearest Neighbor search problem in high-dimensional spaces are applied
    - E.g., based on main memory k-d tree structure with “best bin first” matching and using the Euclidean distance as similarity criterion

---

Speeded up robust features (SURF) [BET08]

- Partially inspired by SIFT descriptors
- SURF approximates and outperforms SIFT with respect to *repeatability*, *distinctiveness*, and *robustness*, yet can be *computed* and *compared* much faster
- This is achieved by using an approximation of Hessian-matrix based on *integral images* (salient points *detection*)

- **Blob-like structures are detected at locations where determinant is maximum**
SURF: salient point descriptor

- **Point descriptor** represents the *distribution of the intensity content* based on Haar-wavelet (1st order) responses within the interest point neighborhood
  - computed based on blocks of 4x4 pixels and using 4-D descriptors
  - Each silent point is represented by a 64-D feature vectors (4x4x4)

![SURF illustration](image)

SURF: properties

- With respect to SIFT, SURF is:
  - More robust to scale and rotation transformations
  - Very sensitive to changes in illumination

  **Matching basic idea:**
  "if the contrast between two interest points is different (dark on light background vs. light on dark background), the candidate is not considered a valuable match"

- Shares same problem with respect to *high dimensionality*
Windsurf framework: an alternative instantiation

- To provide an example of the generality of the Windsurf framework, here we show a pic of the WritingSimilaritySearch system built on top of the Windsurf library.

- Let's instantiate the key points of the Windsurf framework within the new context:
  - Regions correspond to local features (i.e., SURF)
  - Region distance function is the Euclidean distance
  - The matching problem is solved by means of an “approximation” of the 1-1 matching
    - “best bin first” match
    - high dimensionality!!

![WritingSimilaritySearch: an example]

**WritingSimilaritySearch: an example**

query

Top-k results (k=9)
Why MM data are a problem for DBMSs?

- Databases promise:
  - well structured data organization
  - efficient storage of large amounts of structured data querying
  - transactional support for concurrent users
- If we include MM data
  - MM is large and may swamp other data
  - MM data structures are completely different from standard database structures
  - MM data structures do not easily lend themselves to content-based searching

How to represent MM data in DBMSs (1)

- Every commercial DBMS offers traditional functionalities supporting above all numerical and string data types
- Additional user-defined data types, like images, audio, video, etc. need an extensible DBMS
- Extensible DBMSs offer facility to provide new data types, along with functions that operate (e.g., “display”, “comparing”, etc.) on them
- The definition of new data types and the associated functions for them are typically implemented by a specialist
- After the definition of such new data type (e.g., ‘image’), we could create tables that can hold MM employee records, e.g.:

```sql
create table EMPLOYEE (
    empN integer PRIMARY KEY;
    name char(50);
    age integer;
    salary float;
    face image );
```
How to represent MM data in DBMSs (2)

- Assuming that the predicate `similar` has been appropriately defined for the ‘image’ data type, we can look for employees that look like a given person as follows:

  ```sql
  Select name
  From EMPLOYEE
  Where EMPLOYEE.face similar desirableFace
  ```

  where ‘desirableFace’ is the Object-identifier (Obj-ID) of the desirable JPEG image

- Providing the ability to efficiently answer this type of query is our next step!! 😊

Domain types of MM data

- Most current DBMSs provide three different kinds of domain for MM data:
  1. Large object (LOB) data types used to store sequences of unstructured data up to 4GB; two types:
     - Binary Large Objects (BLOBs) which are an unstructured sequence of bytes
     - Character Large Objects (CLOBs) which are an unstructured sequence of characters
  2. File references, instead of holding the data, a file reference contains a link to the data (OLE in Access)
  3. Genuine multimedia data types (e.g., Oracle, IBM DB2, and Jasmine)

- There is an important difference between 3. and 1./2.:
  - multimedia data types present the possibility of exploiting the structure of the data for querying and manipulation
  - provided by means of “Extenders”
  - SQL3 standard is the support for extensible type systems
  - large objects at best allow you to extract sections or to concatenate them
  - file references mean that the DBMS has no access the data at all
MM Extenders

- Extenders are *software packages* that help in the use of large objects (LOBs)
- They define special data types and functions for many types of large objects, including
  - Images
  - Audios and videos
  - Text
  - XML documents
  - Spatial objects (maps)
- Helps make these objects less cumbersome to manipulate with SQL

Example: IBM DB2 Image Extender

- *DB2 Image Extender* allows you to bring together images and related business data in one SQL query, for example, a photograph of a product and some description about it
- *The DB2 Image Extender* supports a variety of *image formats*, such as GIF, JPEG, BMP, and TIFF
- Some *user-defined functions* are provided for *storing, accessing* and *manipulating* images
- You can also define *your own data types and functions* for image data using DB2’s built-in support for *user-defined types* and *user-defined functions*
Some things Image Extender can do

- **Import and export** images and their attributes into and out of a database
  - When you import an image, Image Extender stores and maintains image attributes such as *size in bytes, format, height, width, and number of colors*
- **Control access** to images with the same level of protection as traditional business data
- **Change the formats** of images
  - You have the flexibility of importing or exporting an image in its source format, or converting to a different format. You can also scale an image, rotate it, do black-white image inversion, or change representational characteristics, such as bits per sample and compression type
- **Backup and recover images**
  - Images and their attributes that you store in the database have the same security and recovery protection as traditional business data

MM data and DBMSs: which conclusions?

- At the moment, there is no reason for **putting MM data into a relational DB**
  - you can do something
  - but everything will be very slow
- However, there are still 3 main reasons for **integrating MM data with a relational DB**:
  1. **Cataloguing the data**
     - a column for file names is good enough
  2. **Decorating Reports**
     - The OLE approach works well here otherwise a file name column and a simple application for generating the reports would do
  3. **Web Applications**
     - Again a file name column is good enough
A generic architecture of a MMDBMS

- **MM data extraction and organization**: extract and organize the MM features for retrieval purposes
  - i.e., indexing features with effective structures

- **MM data query processing**: exploit above index structures in order to provide efficient retrieval algorithms based on similarity functions