MM data retrieval

- From the previous lesson we know that features are a smarter way to represent MM data content than their original format
  - e.g., color and texture for an image
- Today we focus on which are the most suitable models for
  - representing, interpreting, describing and
  - comparing such features
    - E.g., color histograms for images by using the Euclidean distance as similarity measure
- ...with the final goal to be able to retrieve from MM collections those objects which are most interesting for us!! 

Content-based search

- First approach to search for MM objects relies on standard text-based techniques, provided objects come with a precise textual description of what they represent/describe, i.e., of their semantics
  - However, the "annotation" of MM objects is a subjective, time consuming, and tedious process (completely manual!!!)
- A more convenient approach, suitable to manage large DBs, is to automatically extract from MM objects a set of (low-level) relevant numerical features that, at least partially, convey some of the semantics of the objects
- Clearly, which are the "best" features to extract depend on the specific medium and on the application at hand (i.e., what we are looking for)

- Look for cheetahs?
  - This is fine; but, how to find it?
Content-based similarity search

- Once we have feature values, we can search objects by using them
- Assume a database (DB) with \( N \) MM objects (e.g., images) and, for each of the \( N \) objects, we have extracted the “relevant features”
- E.g., we could extract some color information from images
- We can now search for objects whose feature values are “similar” (in some sense to be defined) to the feature values of our query \([SWS+00, LSD+07, LZL+07, DJL+08]\)
- In general, this approach, much alike as it happens in text-retrieval, cannot guarantee that all and only relevant results are returned as result of a query

The general scenario

- In general, we have a 2-levels scenario:

Representing color

- In a digital image, the color space that encodes the color content of each pixel of the image is necessarily discretized
  - This depends on how many bits per pixel (bpp) are used
    - Example: if one represents images in the RGB space by using \( 8 \times 3 = 24 \) bpp, the number of possible distinct colors is \( 2^{24} \approx 16,777,216 \)
    - With 8 bits per channel, we have 256 possible values on each channel
- Although discrete, the possible color values are still too many if one wants to compactly represent the color content of an image
  - This also aims at achieving some robustness in the matching process (e.g., the two RGB values (123,078,226) and (121,080,230) are almost indistinguishable)
- In practice, a common approach to represent color is to make use of histograms...

Color histograms

- A color histogram \( h \) is a \( D \)-dimensional vector, which is obtained by quantizing the color space into \( D \) distinct colors
  - Typical values of \( D \) are 32, 64, 256, 1024, ...
    - Example: the HSV color space can be quantized into \( D=32 \) colors: \( H \) is divided into \( 8 \) intervals, and \( S \) into 4
    - \( V = 0 \) guarantees invariance to light intensity
  - The \( i-th \) component (also called bin) of \( h \) stores the percentage (number) of pixels in the image whose color is mapped to the \( i-th \) color
  - Although conceptually simple, color histograms are widely used since they are relatively invariant to translation, rotation, scale changes and partial occlusions
Comparing color histograms

- Since histograms are vectors, we can use any \( L_p \)-norm to measure the distance (dissimilarity) of two color histograms.
- However, doing so we are not taking into account colors' correlation.
- Depending on the query and the dataset, we might therefore obtain low-quality results.
- Weighted \( L_p \)-norms and relevance feedback can partially alleviate the problem.

The problem is that \( L_p \)-norms just consider the difference of corresponding bins, i.e., they perform a 1-1 comparison.
- With color histograms, our "coordinates" are not unrelated ("cross-talk" effect).

Sample queries based on color (1)

Sample queries based on color (2)

Quadratic distance

- Consider two histograms \( h \) and \( q \), both with \( D \) bins.
- Their quadratic distance is defined as:

\[
L_A(h, q; A) = \sqrt{\sum_{i=1}^{D} \sum_{j=1}^{D} a_{i,j} (h_i - q_i)(h_j - q_j)}
\]

where \( A = (a_{i,j}) \) is called the (color)-similarity matrix.
- The value of \( a_{i,j} \) is the "similarity" of the \( i \)-th and the \( j \)-th colors (\( a_{i,i} = 1 \)).
- Note that:
  - when \( A \) is a diagonal matrix we are back to the weighted Euclidean distance,
  - when \( A = I \) (the identity matrix) we obtain the \( L_2 \) distance.
- In order to guarantee that \( L_A \) is indeed a distance (\( L_A(h, q; A) \geq 0 \forall h, q \)), it is sufficient that \( A \) is a symmetric positive definite matrix.
Quadratic distance vs. Euclidean distance

- As a simple example, let D = 3, with colors red, orange, and blue.
- Consider 3 pure-color images and the corresponding histograms:
  
  \[
  h_1 = (1,0,0), \quad h_2 = (0,1,0), \quad h_3 = (0,0,1)
  \]

- Using \( L_2 \), the distance between two different images is always \( \sqrt{2} \).
- On the other hand, let the color-similarity matrix be defined as:

  \[
  \begin{array}{ccc}
  1 & 0.8 & 0 \\
  0.8 & 1 & 0 \\
  0 & 0 & 1 \\
  \end{array}
  \]

- Now we have \( L_A(h_1,h_2) = 0.4 \), whereas \( L_A(h_1,h_3) = L_A(h_2,h_3) = \sqrt{2} \).

Representing texture

- Tamura features correspond to properties of a texture which are readily perceived, that is coarseness, contrast and directionality (3-D feature vector).
  - Coarseness - coarse vs. fine: it provides information about the “granularity” of the pattern.
  - Contrast - high vs. low contrast: it measures the amount of local changes in brightness.
  - Directionality - directional vs. non-directional: it’s a global property of the image.

Texture extraction with Gabor filters

- A Gabor filter is a Gaussian modulated by a sinusoid, which can reveal the presence of a pattern along a certain direction and at a certain scale (frequency).
- To extract texture information, one chooses a number of directions/orientations (e.g., 6) and scales (e.g., 5) according to which the image has to be analyzed.
- For each orientation and scale, the average and the variance (standard deviation) of the filter output are computed.
  - This leads to, say, 2·6·5 = 60-D feature vectors, which are usually compared using the Manhattan (L_1) distance.

Gabor filter

- Let I be an image, with \( I(x,y) \) being the gray-scale value of the pixel in position (x,y).
- A Gabor function is written as
  \[
  G(x,y) = \frac{1}{2\pi\sigma_x\sigma_y} \exp\left(-\frac{x^2}{2\sigma_x^2} - \frac{y^2}{2\sigma_y^2}\right) \cos(2\pi f x y)
  \]
  and is completely determined by its frequency \( (\omega_0) \) and bandwidth \( (\omega_0,\omega_d) \).
- The Gabor filter \( G_{mn}(x,y) \) for scale m and orientation n is then defined as
  \[
  G_{mn}(x,y) = a^{-n}G(x',y') \quad x' = a^{n}(-\cos\theta_n + y\sin\theta_n), \quad y' = a^{-n}(-x\sin\theta_n + y\cos\theta_n)
  \]
  where \( K \) is the total number of orientations.
- Finally, the image is analyzed by convolution with the filter:
  \[
  w_{mn}(x,y) = \sum_{xy} G_{mn}(x',y') I(x',y')
  \]
Representing shape

- Once one has succeeded in extracting an object’s contour, the next step is how to represent/encode it.
- A common approach is to navigate the contour, which leads to an ordering of the pixels in the contour:
  \[ \{ (x(t), y(t)) : t = 1 \ldots M \} \]
- A 2nd step is to represent the resulting curve in a parametric form.
- For instance, a possibility is to resort to complex values, by setting \( z(t) = x(t) + j y(t) \).
- Thus, now we have vectors of complex values...
- The problem is that each vector has a different length (i.e., \( M \) depends on the specific image).

Representative points

- The idea is to keep only the \( D \) most “interesting” points.
- Some methods are:
  - Equally-spaced sampling (a)
  - Grid-based sampling (b)
  - Maximum curvature points (c)
  - Fourier-based methods, which first compute the DFT of the contour, and then keep only the first \( D \) coefficients.
- Working in the frequency domain has several advantages:
  - It can be proved that by properly modifying Fourier coefficients one can achieve invariance to scale, translation and rotation.
  - Further, by viewing shape as a “signal”, one can adopt distance measures that have been developed for the comparison of time series and that are somewhat insensitive to signals’ modifications.

Comparing shapes

- The commonest way to measure the (dis-)similarity of two shape vectors of equal length \( D \) is based on Euclidean distance (\( L_2 \)).
- However, with Euclidean distance we have to face a basic problem:
  - Sensitivity to “alignment of values”.
- Intuitively, we would need a distance measure that is able to “match” a point of time series \( s \) even with “surrounding” points of time series \( q \).
- Alternatively, we may view the time axis as a “stretchable” one.
- A distance like this exists, and is called “Dynamic Time Warping” (DTW).

Sample queries based on shape ([BCP02])

- In a sample query, \( R \) = relevant (same type of fish)

1100 objects’ contours
This is not the whole story…

- ...of course, many other features models (and correspondent distance functions) have been defined for MM data
- This was just a way to provide some concrete examples of features and modalities to comparing them!
- Let’s go now into the details of what happens and how things can become complex in a “real” image retrieval system…

The region-based image retrieval approach

- DB population time:
  - Preprocess 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

Windsurf case study [ABP99, BCP00, BP00, BC03, Bar09a, BCP+09, BCP10]

- 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)

\[
B \in \{L, LH, HL, HH\}
\]

\[
B_{w}^{j} = \sum_{l=1}^{3} w_{l}^{j} B_{l}
\]

\[
B_{w}^{j} = \left[ B_{1}^{j}, B_{2}^{j}, B_{3}^{j} \right]
\]
**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**:
  \[
  d_i = \sqrt{(\mathbf{w}_i^T \mathbf{C}^{-1} \mathbf{w}_j)^2 - 2\mathbf{w}_i^T \mathbf{C}^{-1} \mathbf{w}_j}
  \]

  - 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)

\[
 d_{ij} = \frac{1}{2} \ln \left( \frac{\mathbf{C}_i^{1/2} \mathbf{C}_j^{1/2}}{\mathbf{C}_i^{1/2} + \mathbf{C}_j^{1/2}} \right) + \frac{1}{8} \left( \mathbf{w}_i - \mu_i \right)^T \mathbf{C}_i^{-1} \left( \mathbf{w}_i - \mu_i \right)
\]
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:
  - "one-to-one" match (formulated as Assignment Problem)
  - "many-to-many" match (formulated as Transportation Problem)

Assignment problem

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

Transportation problem

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

Sample query
Traditional databases

- Before entering into the details of MM databases (MMDBs), we need to provide the minimal set of concepts from relational database management systems (RDBMS or just DBMS) for students that have no background on this topic!

- Intuitively, a database (DB) can be seen as a collection of objects representing some information of interest

- A DBMS is a software system able to "manage" collections of objects which can be very large (Giga-Tera byte and more) and shared by different applications in a persistent way (even in presence of faults)
  - "manage" = obtain, elaborate, maintain, produce, distribute

- Examples of DBMSs: Oracle, IBM (DB2 UDB), Microsoft (SQL Server), Sybase, mySQL, PostgreSQL, InterBase

- Proceeds of global sales of the DBMS market increase of more than 10% every year!

Relations as tables

- DBMSs use the relational model (Codd, 1970) to describe the data, that is the information is organized in tables ("relations")

- The rows of table corresponds to records, while the columns correspond to attributes

- The language to store/retrieve information from such tables is the Structured Query Language (SQL)

- Example: if we want to create a table with employees records, so that we can store their employee number, name, age and salary, we can use the following SQL statement:

```sql
create table EMPLOYEE (
    empN integer PRIMARY KEY;
    name char(50);
    age integer;
    salary float );
```

Populating and querying tables

- Tables can be populated with the SQL insert command, e.g.:

```sql
insert into EMPLOYEE values (123, 'Smith, John', 30, 38000.00);
insert into EMPLOYEE values (456, 'Johnson, Tom', 25, 55000.00);
```

- We can retrieve information using the select command. E.g., if we want to find all the employees with salary less that 50000, we use the query:

```sql
Select * 
From EMPLOYEE 
Where salary <= 50000.00
```

Query execution

- In absence of access methods (e.g., an index), the DBMS will perform a sequential scanning, checking the salary of each and every employee record against the desired threshold of 50000!!!

- To accelerate queries execution, we can create an index (usually a B-tree index, as we will see in few minutes) with the command create index

```sql
create index salIdx on EMPLOYEE(salary)
```

- E.g., to build an index on the employee's salary, we would issue the SQL statement:

```sql
create index salIdx on EMPLOYEE(salary)
```

- In general the DBMS relies on an "optimizer" component to decide which is the more efficient way to execute a given query
  - sequential vs. index-based evaluation
  - which index is the most appropriate
  - ...
Storage hierarchies

- First level is typically main memory or core or RAM
  - Fast (access time of micro-seconds or faster), small, expensive
- Second level (secondary store) is typically magnetic disk
  - Much slower (5-10 msec. access time), but much larger and cheaper
- Database researchers has focused on "large databases" that do not fit in main memory and thus have to be stored in secondary memory
- Secondary store is organized into block (= pages)
  - The reason is that, accessing data from the disk involves the mechanical move of the read/write head of the disk above the appropriate track on the disk
  - Because these moves ("seeks") are slow and expensive, every time we do a disk read we bring into main memory a whole disk block (of the order of 1KB - 8 KB)
  - So, it makes a huge difference of performance if we mange to group similar data in the same disk blocks!!

B*-tree

- B-tree variant
  - more commonly used than B-tree
- Internal node
  \[ \begin{array}{c}
  P_1 \quad K_1 \quad \ldots \quad K_i \quad P_i \quad K \quad \ldots \quad K_q \quad P_q \\
  x \quad \ldots \quad x \quad \ldots \quad x \\
  X \leq K_i \quad X \leq K_i \quad K_i < X \\
  \end{array} \]
- Leaf node
  \[ \begin{array}{c}
  K_1 \quad P_1 \quad \ldots \quad K_i \quad P_i \quad \ldots \quad K_q \quad P_q \\
  \text{data pointer} \quad \text{data pointer} \quad \text{data pointer} \\
  \text{pointer to next leaf node in tree} \\
  \end{array} \]

B-tree

- Access methods, like B-tree, try exactly to achieve good clustering of data in order to minimize the number of disk-reads

What else...

- The relational model and SQL provide a large number of additional features, such as:
  - the ability to retrieve information from several tables ("joins"); the matching is based on values!
  - the ability to perform aggregate operations (e.g., sums, averages, etc.)
- However we restrict the discussion to the above few features which are the essential ones for MM data retrieval
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 all mentioned functionalities supporting above all numerical and string datatypes
- Additional user-defined datatypes, 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 datatypes and the associated functions for them are typically implemented by a specialist
- After the definition of such new datatype (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’ datatype, 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 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 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