Multimedia Data Indexing

Paolo Ciaccia
DEIS - University of Bologna, Italy
paolo.ciaccia@unibo.it
http://www-db.deis.unibo.it/~pciaccia

SYNONYMS

MM indexing

DEFINITION

Multimedia (MM) data indexing refers to the problem of preprocessing a database of MM objects so that they can be efficiently searched for on the basis of their content. Due to the very nature of MM data, indexing solutions are needed to efficiently support *similarity queries*, where the similarity of two objects is usually defined by some expert of the domain and can vary depending on the specific application. Peculiar features of MM indexing are the intrinsic high-dimensional nature of the data to be organized, and the complexity of similarity criteria that are used to compare objects. Both aspects are therefore to be considered for designing efficient indexing solutions.

HISTORICAL BACKGROUND

Earlier approaches to the problem of MM data indexing date back to the beginning of 90's, when it became apparent the need of efficiently supporting queries on large collections of non-standard data types, such as images and time series. Representing the content of such data is typically done by automatically extracting some low-level features (e.g., the color distribution of a still image), so that the problem of finding objects similar to a given reference one is transformed into the one of looking for similar features. Although at that time many solutions from the Pattern Recognition field were available for this problem, they were mainly concerned with the *effectiveness* issue (which features to consider and how to compare them), thus almost disregarding *efficiency* aspects.

The issue of making similarity query processing scalable to large databases was first considered in systems like QBIC [6] for the indexing of color images and by more focused approaches such as the one described by Jagadish in [8] for indexing shapes. Not surprisingly, these solution adopted index methods available at that time that had been developed for the case of low-dimensional Spatial Databases, such as R-trees and Grid files. The peculiarity of MM data then originated a flourishing brand new stream of research, which resulted in many indexes explicitly addressing the problems of high-dimensional features and complex similarity criteria.

SCIENTIFIC FUNDAMENTALS

Figure 1 illustrates the typical scenario to be dealt with for indexing multimedia data. The 1st step, *feature extraction*, is concerned with the problem of highlighting those relevant features, f_i, of an object o_i on which content-based search wants to be performed. In the figure, this is the shape of the image subject (a cheetah). The 2nd step, *feature approximation*, is optional and aims to obtain a more compact representation, af_i, of f_i that can be inserted into a suitable index structure (3rd step). It has to be remarked that, while feature extraction is needed to define which are the relevant aspects of objects on which the search has to focus on, feature approximation is mainly motivated by feasibility and efficiency reasons. This is because it might not be possible to directly index non-approximate features and/or indexing approximate features might result in a better performance of

the search algorithms.

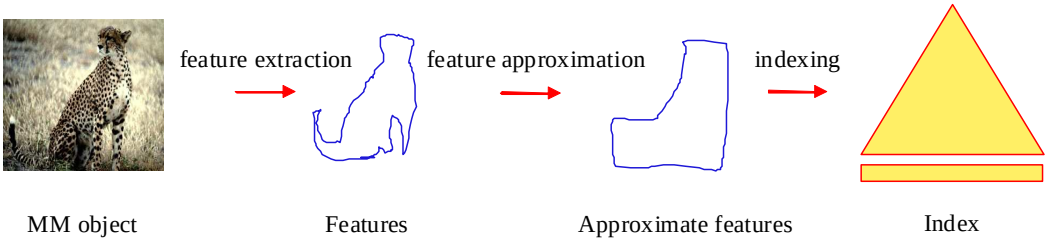

Figure 1: The multimedia data indexing scenario.

Consider a collection $O = \{o_1, o_2, \ldots, o_n\}$ of MM objects with corresponding features $F = \{f_1, f_2, \ldots f_n\}$ and approximate features $AF = \{af_1, af_2, \ldots af_n\}$. In order to compare features, a *distance function* d is typically set up, where $d(f_i, f_j)$ measures how dissimilar are the feature values of objects o_i and o_j. Given a reference object q (the *query point*), a range query with radius ϵ, also called an ϵ-similarity query, will return all the objects $o_i \in O$ such that $d(f_i, f(q)) \leq \epsilon$, whereas a k-nearest neighbor query (k-NN) will return the k objects in O whose features are closest to those of q.
A simple yet remarkable result due to Agrawal, Faloutsos, and Swami [1], and now popularly known as the lower-bounding lemma, provides the basis for exactly solving queries by means of an index that organizes approximate features:
The lower-bounding lemma: Let I be an index that organizes the set of approximate features $AF = \{af_1, af_2, \ldots af_n\}$ and that compares such features using an *approximate distance* d_{appr}. If, for any pair of objects, it is $d_{appr}(af_i, af_j) \leq d(f_i, f_j)$, then the result of a range query obtained from I is guaranteed to contain the exact result, i.e., no false dismissals are present.
The result easily follows from the observation that, since d_{appr} lower bounds d by hypothesis, $d(f_i, f(q)) \leq \epsilon$ implies $d_{appr}(af_i, af(q)) \leq \epsilon$. The lower-bounding lemma guarantees that querying the index with a search radius equal to ϵ will return a result set that contains all the objects whose non-approximate features satisfy the query constraint.

Filter & Refine. When indexing is based on an approximate distance, a two-step *filter & refine* process is therefore needed, in which the role of the index is to filter out many irrelevant objects. The so-resulting candidate objects then need to be verified by using the actual distance d. The lower-bounding lemma is also the key for solving k-NN queries using a multi-step query processing approach.
The effectiveness of the filter & refine approach depends on two contrasting requirements:
1) The approximate distance function d_{appr} should be a tight approximation of d, in order to minimize the number of false hits, i.e., those objects that do not satisfy the query constraint yet the index is not able to discard them. These are exactly those objects o_i for which both $d_{appr}(af_i, af(q)) \leq \epsilon$ and $d(f_i, f(q)) > \epsilon$ hold.
2) At the same time, d_{appr} should be relatively cheap to compute as compared to d, in order to avoid wasting much time in the filter phase.
The literature on MM indexing abounds of examples showing how to derive effective approximations for complex distance functions. For instance, the QBIC system compares color images using a quadratic form distance function, $d_A^2(f_i, f_j) = (f_i - f_j)A(f_i - f_j)^T = \sum_{k=1}^{D}\sum_{l=1}^{D} a_{k,l}(f_{i,k} - f_{j,k})(f_{i,l} - f_{j,l})$, where $A = (a_{k,l})$ is a color-to-color similarity matrix and features are color histograms. Evaluating d_A has complexity $O(D^2)$, which becomes too costly even for moderately large values of D, the number of bins in the color histograms. In [6] it is demonstrated that using as approximate features the average RGB color of an image, which is a 3-dimensional vector, and comparing average colors using the Euclidean distance, i.e., $d_{avg}^2(af_i, af_j) = \sum_{k\in\{R,G,B\}}(af_{i,k} - af_{j,k})^2$, leads to derive that $d_{avg}^2 \leq d_A^2/\lambda_1$, where λ_1 is the smallest eigenvalue of matrix A. Then, the lower-bounding lemma guarantees that querying an index built on average colors with a range query of radius $\epsilon/\sqrt{\lambda_1}$ will not lead to any false dismissal.
As another relevant example, consider the problem of comparing feature vectors that represent time-varying signals. A distance function more robust than the Euclidean one to misalignments on the time domain is the

Dynamic Time Warping (DTW) distance. However, evaluating DTW has a complexity $O(D^2)$, which is untenable for long time series. In [9] Keogh introduces an effective lower-bounding function for the DTW distance. In essence, the idea is to construct an *envelope*, $Env(q)$, around the query time series q, after that an Euclidean-like distance between $Env(q)$ and a stored sequence f_i can be computed in $O(D)$ time.

The need for approximate features. As anticipated, there are several reasons for which approximate features might have to be considered. First, in many relevant cases the features of a MM object are represented through a high-dimensional vector, $f_i = (f_{i,1}, f_{i,2}, \ldots, f_{i,D})$, with D of the order of the hundreds or even thousands. At such high dimensions it is known that the performance of multidimensional indexes rapidly deteriorates, becoming either comparable to, or even worse than, that of a sequential scan. This phenomenon, known as the dimensionality curse, inhibits any approach based on a direct indexing of feature values. Besides ad hoc solutions, such as those above described, one might consider using some dimensionality reduction technique that projects feature vectors onto a (much) lower D'-dimensional space, $D' \ll D$, and then indexing the so-obtained D'-dimensional feature vectors. The effectiveness of such techniques is however highly variable, it being dependent on the actual data distribution.
Another practical reason that could motivate the use of approximate features is the mismatch between the type of the features and the one natively supported by the index. As a simple example, consider an index implementation that only manages entries of an arbitrary, but fixed, size, and that objects to be indexed are regions of pixels described by their boundaries. Clearly, boundary descriptions have different sizes, depending on the shape of the region. In this case a possible solution would be to use a *conservative approximation* of boundaries, like minimum bounding rectangles.
Finally, it might also be the case that, although in principle actual feature values could be stored in the index, the distance function to be used on them cannot be supported by the index organization. A remarkable example is the DTW distance: since DTW is *not* a true metric, in that it does not satisfy the triangle inequality, no multidimensional index can directly process queries with such a distance function.

Metric indexing. When features are not vectors and/or the distance function is not the Euclidean distance or some other (possibly weighted) L_p norm, coordinate-based spatial indexes cannot be used. There are many cases in which this situation shows up. For instance, in region-based image retrieval (RBIR), each image is first automatically segmented into a set of homogeneous regions, each of them being represented by a vector of low-level features (usually encoding color and texture information). Thus, each f_i is a *set of vectors* and as such cannot be indexed by a spatial index. As a further example, *graphs* representing, say, spatially located objects with their relationships cannot be directly supported by a coordinate-based index. In cases like these one could consider using a metric index, such as the M-tree [5]. A metric index just requires the distance function d used to compare feature values to be a metric, i.e., a positive and symmetric function that also satisfies the triangle inequality: $d(f_i, f_j) \leq d(f_i, f_k) + d(f_k, f_j)\ \forall f_i, f_j, f_k$. Although there is nowadays a large number of metric indexes available [14], as demonstrated in [3] all of them are based on the common principle of organizing the indexed features into a set of equivalence classes and then discarding some of these classes by exploiting the triangle inequality. For instance, in the case of the M-tree each class corresponds to the set of feature values stored into a same leaf of the tree. Triangle inequality can also be applied to save some distance computations while searching the index, which turns out to be particularly relevant in the case of computationally demanding distance functions (a common case with MM data). This was first shown for the M-tree, in which distances between each feature value and its parent in the index tree are precomputed and stored in the tree. The idea is quite general and effective, an obvious tradeoff existing between the amount of extra information stored in the tree and the benefit this has on pruning the search space. Along this direction, Skopal and Hoksza propose the M*-tree [12], a variant of the M-tree in which each entry in a node also includes its NN in that node, i.e., the *NN-graph* of the features in each node is maintained.
A common objection to metric indexes is that they are bound to use only a specific distance function, namely the one with which the index is built. Along the direction of increasing flexibility, Ciaccia and Patella [4] introduce the QIC-M-tree, which is an extension of the M-tree able to support queries with any distance function d_Q from the same "family" of the distance d_I used to build the tree. On the condition that there exists a *scaling factor* $S_{d_I \to d_Q}$ such that $d_I(f_i, f_j) \leq S_{d_I \to d_Q} d_Q(f_i, f_j)$ holds (i.e., d_I lower bounds d_Q up to a constant factor), the lower-bounding lemma applies, and the index can answer queries based on d_Q. A similar idea allows the QIC-M-tree

to use also a "cheap" approximate distance d_C as a filter before computing the "costly" d_I and d_Q functions.

Ad hoc solutions. The availability of general purpose metric indexes does not rule out the possibility of deriving better, more specialized, solutions for the problem at hand. For instance, the STRG-Index [10] is a specialized structure for indexing spatio-temporal graphs arising from the modelling of video sequences. Consider a video segment with N frames. Each frame is first segmented into a set of homogeneous color regions, each of which becomes a node in the Region Adjacency Graph (RAG) of that frame, with edges connecting spatially adjacent regions. Node attributes (such as size, color, and location) are then defined, and the same is done for edges (in which case attributes such as the distance and the orientation between the centroids of connected regions can be used). Since a node representing a region can span multiple frames, nodes in consecutive RAGs can be connected to represent temporal aspects. The resulting graph is called Spatio-Temporal Region Graph (STRG). The STRG is then decomposed into a set of Object Graphs (OGs) and Background Graphs (BGs), and clusters of OGs are obtained for the purpose of indexing. Since the distance function used for comparing OGs (the so-called Extended Graph Edit Distance (EGED)) is a metric, any metric index could be used. The ad hoc STRG-Index proposed in [10] is a 3-level metric tree, where the root node contains entries for the BGs, the intermediate level stores clusters of OGs, and individual OGs are inserted into the leaf level.
Extensions of available indexes might be also required as a consequence of feature approximation. An example is found in [13], where the problem of providing rotation-invariant retrieval of shapes under the Euclidean (L_2) distance is considered. After converting a shape boundary into a time series $f_i = (f_{i,1}, f_{i,2}, \ldots, f_{i,D})$ (this is quite a common way to represent shapes, see e.g. [2]), a Discrete Fourier Transform (DFT) is applied to obtain a representation of f_i in the frequency domain. Due to Parseval's theorem, the DFT transformation preserves the Euclidean distance [1]. To obtain invariance to rotation, only the magnitude of DFT coefficient is retained. The so-resulting vectors F_i are then compressed by keeping only the k ($k \ll D$) coefficients with the highest magnitude (together with their position in the original vector) plus an error term ϵF_i given by the square root of the sum of the squares of dropped coefficients. This information allows a tight lower bound to be derived on the actual rotation-invariant Euclidean distance between f_i and a query shape q. For indexing, a variation of the VP-tree is introduced, which allows compressed features to be stored and searched.

KEY APPLICATIONS

Any application dealing with massive amounts of multimedia data requires effective indexing solutions for efficiently supporting similarity queries. This is further motivated by the complexity of distance functions that are of interest for multimedia data.

FUTURE DIRECTIONS

All the above indexing techniques and methods assume (at least) that the distance function is a metric. An interesting problem is to devise indexing methods for non-metric distance functions that do not rely on the lower-bounding lemma. The work of Skopal [11] on *semimetrics* appears to be a relevant step on this direction. In the same spirit, Goial, Lifshits and Schütse [7] study how to avoid turning the *similarity* search problem into a *distance*-based one, which in several cases might not yield a metric. Working directly with similarities is however more complex, since there is no analogue of the triangle inequality property for similarity values. Let $rank_y(x)$ be the rank of object x with respect to object y (i.e., x is the NN of y if $rank_y(x) = 1$). Then, [7] introduces the concept of *disorder constant* DC, the smallest value for which the *disorder inequality* $rank_y(x) \leq DC(rank_z(x) + rank_z(y))$ holds $\forall x, y, z$ in the given dataset, and describes algorithms for NN search based on this idea. Making this approach practical for large MM databases remains an open problem.

CROSS REFERENCES

Multimedia Data Querying
Indexing Metric Spaces
Spatial Indexing Techniques
Indexing and Similarity Search
High Dimensional Indexing
Dimensionality Curse
Dimensionality Reduction Techniques

RECOMMENDED READING

[1] Rakesh Agrawal, Christos Faloutsos, and Arun Swami. Efficient similarity search in sequence databases. In *Proceedings of the 4th International Conference on Foundations of Data Organizations and Algorithms, (FODO'93)*, pages 69–84, Chicago, IL, USA, October 1993. Springer, LNCS, Vol. 730.

[2] Ilaria Bartolini, Paolo Ciaccia, and Marco Patella. WARP: Accurate retrieval of shapes using phase of Fourier descriptors and time warping distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(1):142–147, 2005.

[3] Edgar Chávez, Gonzalo Navarro, Ricardo Baeza-Yates, and José Luis Marroquín. Proximity searching in metric spaces. *ACM Computing Surveys*, 33(3):273–321, September 2001.

[4] Paolo Ciaccia and Marco Patella. Searching in metric spaces with user-defined and approximate distances. *ACM Transactions on Database Systems*, 27(4):398–437, December 2002.

[5] Paolo Ciaccia, Marco Patella, and Pavel Zezula. M-tree: An efficient access method for similarity search in metric spaces. In *Proceedings of the 23rd International Conference on Very Large Data Bases (VLDB'97)*, pages 426–435, Athens, Greece, August 1997. Morgan Kaufmann.

[6] Christos Faloutsos, Ron Barber, Myron Flickner, Jim Hafner, Wayne Niblack, Dragutin Petkovic, and William Equitz. Efficient and effective querying by image content. *Journal of Intelligent Information Systems*, 3(3/4):231–262, July 1994.

[7] Navin Goyal, Yuri Lifshits, and Hinrich Schütse. Disorder inequality: A combinatorial approach to nearest neighbor search. In *Proceedings of the 1st ACM International Conference on Web Search and Data Mining (WSDM'08)*, Stanford, CA, USA, February 2008. ACM Press.

[8] H. V. Jagadish. A retrieval technique for similar shapes. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 208–217, Denver, CO, USA, May 1991. ACM Press.

[9] Eamonn Keogh. Exact indexing of dynamic time warping. In *Proceedings of the 28th International Conference on Very Large Data Bases (VLDB'02)*, pages 406–417, Hong Kong, China, September 2002. Morgan Kaufmann.

[10] JeongKyu Lee, Jung-Hwan Oh, and Sae Hwang. STRG-index: Spatio-temporal region graph indexing for large video databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 718–729, Baltimore, MD, USA, June 2005. ACM Press.

[11] Tomáš Skopal. On fast non-metric similarity search by metric access methods. In *Proceedings of the 10th International Conference on Extending Database Technology (EDBT'06)*, pages 718–736, Munich, Germany, March 2006. Springer LNCS, Vol. 3896.

[12] Tomáš Skopal and David Hoksza. Improving the performance of M-tree family by nearest-neighbor graphs. In *Proceedings of the 11th East European Conference on Advances in Databases and Information Systems (ADBIS'07)*, pages 172–188, Varna, Bulgaria, October 2007. Springer LNCS, Vol. 4690.

[13] Michail Vlachos, Zografoula Vagena, Philip S. Yu, and Vassilis Athitsos. Rotation invariant indexing of shapes and line drawings. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management (CIKM'05)*, pages 131–138, Bremen, Germany, November 2005. ACM Press.

[14] Pavel Zezula, Giuseppe Amato, Vlastislav Dohnal, and Michal Batko. *Similarity Search: The Metric Space Approach (Advances in Database Systems)*. Springer, 2005.