V

Valid Time

Christian S. Jensen[1], Richard T. Snodgrass[2]
[1]Aalborg University, Aalborg, Denmark
[2]University of Arizona, Tucson, AZ, USA

Synonyms

Real-world time; Intrinsic time; Logical time; Data time

Definition

The *valid time* of a fact is the time when the fact is true in the modeled reality. Any subset of the time domain may be associated with a fact. Thus, valid timestamps may be sets of time instants and time intervals, with single instants and intervals being important special cases. Valid times are usually supplied by the user.

Key Points

While other temporal aspects have been proposed, such as "decision time" and "event time," closer analysis indicates that the decision and event time of a fact can be captured as either a valid time or a transaction time of some related fact.

For example, consider a valid-time `Faculty` table with attributes `Name` and `Position` (that is, either Assistant Professor, Associate Professor, or Professor) that captures the positions of faculty members. The valid time then captures when a faculty member held a specific position.

The "decision time" of a faculty member holding a specific position would be the time the decision was made to hire or promote the faculty member into that position. In fact, there are generally several decision times for a promotion: recommendation by departmental promotion and tenure committee, recommendation by department chair, recommendation by college promotion and tenure committee, recommendation by Dean, and finally decision by Provost. Each of these decisions may be modeled as a separate fact, with an (event) valid time that captures when that decision was made.

Oracle explicitly supports valid time in its Workspace Manager. The support includes sequenced primary keys, sequenced uniqueness, sequenced referential integrity, and sequenced selection and projection.

The term "valid time" is widely accepted; it is short and easily spelled and pronounced. Most importantly, it is intuitive.

Concerning the alternatives, the term "real-world time" derives from the common identification of the modeled reality (opposed to the reality of the model) as the real world. This term is less frequently used. "Intrinsic time" is the opposite of extrinsic time. Choosing intrinsic time for valid time would require one to choose extrinsic time for transaction time. The terms are appropriate: The time when a fact is true is intrinsic to the fact; when it happened to be stored in a database is clearly an extrinsic property. However, "intrinsic" is rarely used. "Logical time" has been used for valid time in conjunction with "physical time" for transaction time. As the discussion of intrinsic time had to include extrinsic time, discussing logical time also requires the consideration of physical time. Both terms are used much less frequently than valid and transaction time, and they do not posses clear advantages over these. The term "data time" is probably the most rarely used alternative. While it is clearly brief and easily spelled and pronounced, it is not intuitively clear that the data time of a fact refers to the valid time as defined above.

Cross-references

▶ Nonsequenced Semantics
▶ Sequenced Semantics
▶ Temporal Database
▶ Temporal Generalization
▶ Time Instant
▶ Time Interval
▶ Time Period
▶ Transaction Time
▶ User-Defined Time
▶ Valid-Time Indexing

Recommended Reading

1. Jensen C.S. and Dyreson C.E. (eds.). A consensus glossary of temporal database concepts – February 1998 version. In Temporal Databases: Research and Practice, O. Etzion, S. Jajodia, S. Sripada (eds.). LNCS 1399, Springer-Verlag, Berlin, 1998, pp. 367–405.
2. Snodgrass R.T. and Ahn I. A taxonomy of time in databases. In Proc. ACM-SIGMOD Int. Conf. on Management of Data, 1985, pp. 236–246.
3. Snodgrass R.T. and Ahn I. Temporal databases. IEEE Comput., 19(9):35–42, September 1986.

Validity (Satisfiability)

► Certain (and Possible) Answers

Valid-Time Access Methods

► Valid-Time Indexing

Valid-Time Algebras

► Temporal Algebras

Valid-Time and Transaction-Time Relation

► Bitemporal Relation

Valid-Time Data Model

► Temporal Data Models

Valid-Time Indexing

Mirella M. Moro[1], Vassilis J. Tsotras[2]
[1]Federal University of Rio Grande do Sol, Porto Alegre, Brazil
[2]University of California-Riverside, Riverside, CA, USA

Synonyms

Valid-time access methods

Definition

A valid-time index is a temporal index that enables fast access to valid-time datasets. In a traditional database, an index is used for selection queries. When accessing valid-time databases such selections also involve the valid-time dimension (the time when a fact becomes valid in reality). The characteristics of the valid-time dimension imply various properties that the temporal index should have in order to be efficient. As traditional indices, the performance of a temporal index is described by three costs: (i) storage cost (i.e., the number of pages the index occupies on the disk), (ii) update cost (the number of pages accessed to perform an update on the index; for example when adding, deleting or updating a record), and (iii) query cost (the number of pages accessed for the index to answer a query).

Historical Background

A valid-time database maintains the entire temporal behavior of an enterprise as best known now [13]. It stores the current knowledge about the enterprise's past, current or even future behavior. If errors are discovered about this temporal behavior, they are corrected by modifying the database. If the knowledge about the enterprise is updated, the new knowledge modifies the existing one. When a correction or an update is applied, previous values are not retained. It is thus not possible to view the database as it was before the correction/update. This is in contrast to a transaction-time database, which maintains the database activity (rather than the real world history) and can thus rollback to a past state. Hence, in a valid-time database the past can change, while in a transaction-time database it cannot.

The problem of indexing valid-time databases can be reduced to indexing dynamic collections of intervals, where an interval represents the temporal validity of a record. Note that the term "interval" is used here to mean a "convex subset of the time domain" (and not a "directed duration"). This concept has also been named a "period"; in this discussion, however, only the term "interval" is used.

To index a dynamic collection of intervals, one could use R-trees or related dynamic access methods. Relatively fewer approaches have been proposed for indexing valid-time databases. There have been various approaches proposed for the related problem of managing intervals in secondary storage. Given the problem difficulty, the majority of these approaches have focused on achieving good worst case behavior; as a result, they are mainly of theoretical importance.

For a more complete discussion the reader is referred to a comprehensive survey [12].

Foundations

The following scenario exemplifies the distinct properties of the valid time dimension. Consider a dynamic collection of "interval-objects." The term interval-object is used to emphasize that the object carries a valid-time interval to represent the temporal validity of some object property. (In contrast, and to emphasize that transaction-time represents the database activity rather than reality, note that intervals stored in a transaction time database correspond to when a record was inserted/updated in the database.)

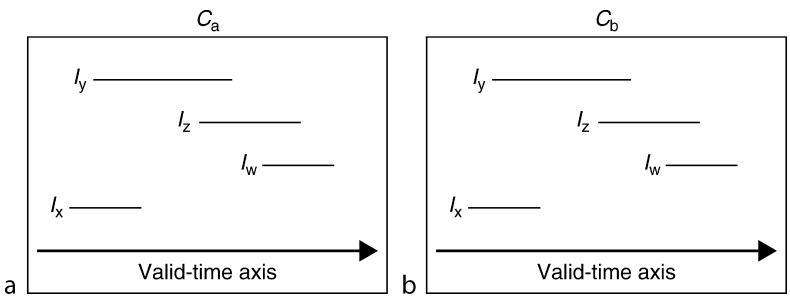
The allowable changes in this environment are the addition/deletion/modification of an interval-object. A difference with the transaction-time abstraction (the reader is referred to the Transaction-Time Indexing entry for more details) is that the collection's evolution (past states) is *not* kept. An example of a dynamic collection of interval-objects appears in Fig. 1. Assume that collection C_a has been recorded in some erasable medium and a change happens, namely object I_z is deleted. This change is applied on the recorded data and physically deletes object I_z. The medium now stores collection C_b, i.e., collection C_a is *not* retained. Note that when considering the valid time dimension, changes do not necessarily come in increasing time order (as is the case in transaction-time databases); rather they can affect *any* object in the collection. This implies that a valid-time database can correct errors in previously recorded data. However, only a single data state is kept, the one resulting after the correction is applied.

As a real-life example, consider the collection of contracts in a company. Each contract has an identity (*contract_no*) and an interval representing the contract's duration or validity. In collection C_a, there were four contracts in the company. But assume an error was discovered: contract I_z was never a company contract (maybe it was mistakenly entered). Then, this information is permanently deleted from the collection, which now is collection C_b.

The notion of time is now related to the valid-time axis. Given a valid-time instant, interval-objects can be classified as past, future or current as related to this instant, if their valid-time interval is before, after or contains the given instant. Valid-time databases can be used to correct errors anywhere in the valid-time domain (past, current or future) because the record of any interval-object in the collection can be changed, independently of its position on the valid-time axis. Note that a valid-time database may store records with the same surrogate but with non-intersecting valid-time intervals. For example, another object with identity I_x could be added in the collection at C_b as long as its valid-time interval does not intersect with the valid-time interval of the existing I_x object; the new collection will contain both I_x objects, each representing object I_x at different times in the valid-time dimension.

From the above discussion, an index used for a valid-time database should: (i) store the latest collection of interval-objects, (ii) support addition/deletion/modification changes to this collection, and (iii) efficiently query the interval-objects contained in the collection when the query is asked. Hence, a valid-time index should manage a *dynamic* collection of intervals.

Thus related is research on the interval management problem. Early work in main memory data-structures has proposed three separate approaches into managing intervals, namely, the Interval Tree, the Segment Tree and the Priority Search Tree [9]. Such approaches are focused on the "stabbing query" [6], i.e., given a collection of intervals, find all intervals



Valid-Time Indexing. Figure 1. Two valid-time databases.

that contain a given query point q. The stabbing query is one of the simpler queries for a valid-time database environment, since it does not involve any object key attribute (rather, only the object intervals).

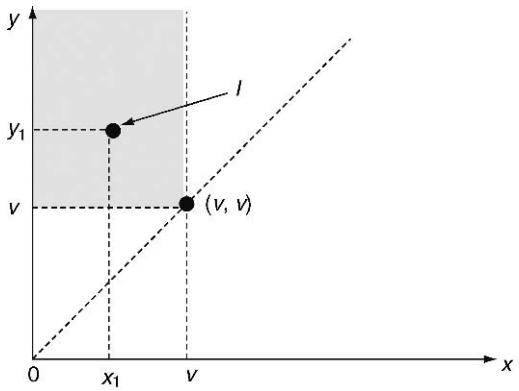
Various methods have been proposed for making these main-memory structures disk-resident. Among them are: the External Memory Interval Tree [1], the Binary-Blocked Interval Tree [4], the External Segment Tree [2], the Segment Tree with Path Caching [11] and the External Memory Segment Tree [1], the Metablock Tree [6], the External Priority Search Tree [5], and the Priority Search Tree with Path Caching [11]. Many of these approaches are mainly of theoretical interest as they concentrate on achieving a worst-case optimal external solution to the 1-dimensional stabbing query.

Recently, [8] presented the Relational Interval Tree (RI-tree) which is based on the original Interval Tree. In particular, the Interval Tree uses a backbone balanced binary tree structure that organizes the values of all the bounding points of the intervals. An interval (l, r) is registered in the highest node w of the backbone tree that it overlaps (that is, while descending the backbone tree, the first node w for which $l \leq w \leq r$). Each inner node w contains two sorted lists, one with the lower bounding points and one with the upper bounding points of the intervals assigned to this node. A "stabbing" query is answered by traversing one path of the backbone structure and accessing the lists in the nodes of this path. The RI-tree uses plain B+-trees to index these sorted lists, thus leading to fast query time. For n intervals stored in the RI-tree, the space used is linear $O(n/B)$ where B is the page capacity in records. If h corresponds to the height of the backbone structure, answering a stabbing query that returns s intervals takes $O(h \log Bn + s/B)$ page accesses.

Since intervals are 2-dimensional objects, a dynamic multi-dimensional index like an R-tree may be used. Moreover, the R-tree can also index other attributes of the valid-time objects, thus enabling queries involving non-temporal attributes as well. For example, "find contracts that were active on time t and had contract-id in the range $(r1,r2)$." While simple, the traditional R-tree approach may not always be very efficient. The R-tree will attempt to cluster intervals according to their length, thus creating pages with possibly large overlapping. It was observed in [7] that for data with non-uniform interval lengths (i.e., a large proportion of "short" intervals and a small proportion of "long" intervals), this overlapping is clearly increased, affecting the query and update performance of the index. This in turn decreases query and update efficiency.

Another straightforward approach is to transform intervals into 2-dimensional points and then use a Point Access Method (quad-tree, grid file, hB-tree, etc.). In Fig. 2, interval $I = (x_1,y_1)$ corresponds to a single point in the 2-dimensional space. Since an interval's end-time is always greater or equal than its start-time, all intervals are represented by points above the diagonal $x = y$. Note that an interval (x,y) contains a query time v if and only if its start-time x is less than or equal to v and its end-time y is greater than or equal to v. Then an interval contains query v if and only if its corresponding 2-dimensional point lies inside the box generated by lines $x = 0$, $x = v$, $y = v$, and $y = \infty$ (the shaded area in Fig. 2). This approach avoids the overlapping mentioned above. Long intervals will tend to be stored together in pages with other long intervals (similarly, for the short intervals). However, no worst case guarantees for good clustering are possible.

A related approach (MAP21) is proposed in [11]. An interval (l, r) is mapped to a point $z = (l \times 10^s) + r$, where s is the maximum number of digits needed to represent any point in the interval range. This is enough to map each interval to a separate point. A regular B^+-tree is then used to index these points. An advantage of this approach is that interval insertions/deletions are easy using the B^+-tree. To answer a stabbing query about q, the point closer but less than q is found among the points indexed in the B^+-tree, and then a sequential search for all intervals before q is performed. At worse, many intervals that do not intersect q can be found (this approach assumes that in

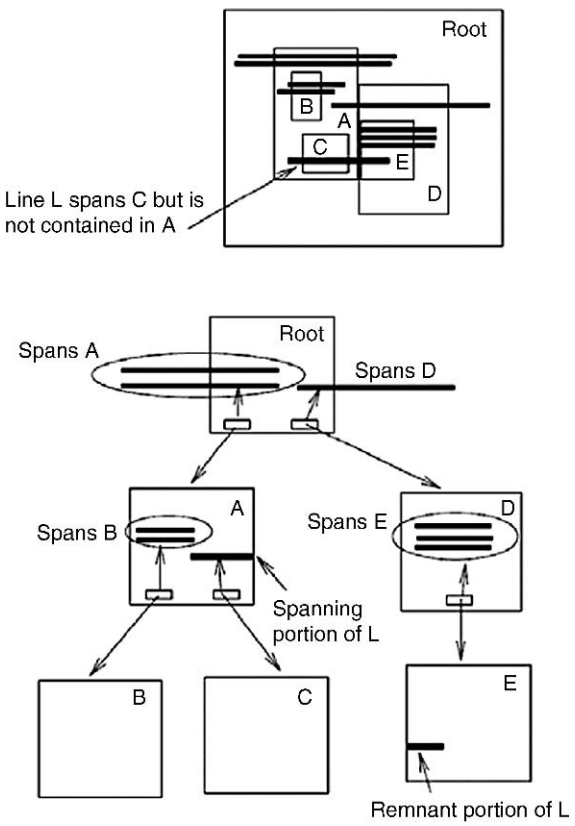


Valid-Time Indexing. Figure 2. An interval $I = (x_1,y_1)$ corresponds to a point in a 2-dimensional space.

practice the maximal interval length is known, which limits how far back the sequential search continues from q).

Another approach is to use a combination of an R-tree with Segment Tree properties. The Segment R-tree (SR-tree) was proposed in [11]. The SR-tree is an R-tree where intervals can be stored in both leaf and non-leaf nodes. An interval I is placed to the highest level node X of the tree such that I spans at least one of the intervals represented by X's child nodes. If I does not span X, it spans at least one of its children but is not fully contained in X, then I is fragmented. Figure 3 shows an example of the SR-tree approach. The top rectangle depicts the R-tree nodes (root, A, B, C, D and E) as well as the stored intervals. Interval L spans the rectangle of node C, but is not contained in node A. It is thus fragmented between nodes A and E.

Using this idea, long intervals will be placed in higher levels of the R-tree. Hence, the SR-tree tends to decrease the overlapping in leaf nodes (in the regular R-tree, a long interval stored in a leaf node will "elongate" the area of this node thus exacerbating the overlap problem). However, having large numbers of spanning records or fragments of spanning records stored high up in the tree decreases the fan-out of the index as there is less room for pointers to children. It is suggested to vary the size of the nodes in the tree, making higher-up nodes larger. "Varying the size" of a node means that several pages are used for one node. This adds some page accesses to the search cost.



Valid-Time Indexing. Figure 3. An example of the SR-tree approach.

As with the R-tree, if the interval is inserted at a leaf (because it did not span anything) the boundaries of the space covered by the leaf node in which it is placed may be expanded. Expansions may be needed on all nodes on the path to the leaf, which contains the new record. This may change the spanning relationships since existing intervals may no longer span children, which have been expanded. In this case, such intervals are reinserted in the tree, possibly being demoted to occupants of nodes they previously spanned. Splitting nodes may also cause changes in spanning relationships as they make children smaller – former occupants of a node may be promoted to spanning records in the parent.

In contrast to the traditional R-tree, the space used by the SR-tree is no longer linear. An interval may be stored in more than one non-leaf nodes (in the *spanning* and *remnant* portions of this interval). Due to the use of the segment-tree property, the space can be as much as $O(n \log_B n)$. Inserting an interval still takes logarithmic time. However, due to possible promotions, demotions and fragmentation, insertion is slower than in the R-tree. Even though the segment property tends to reduce the overlapping problem, the (pathological) worst case performance for the deletion and query time remains the same as for the R-tree organization (that is, at worst, the whole R-tree may have to searched for an interval). The average case behavior is, however, logarithmic. Deletion is a bit more complex, as all the remnants of the deleted interval have to be deleted too. The original SR-tree proposal thus assumed that deletions of intervals are not that frequent.

The SR-tree search algorithm is similar to that of the original R-tree. It descends the index depth-first, descending only those branches that contain the given query point q. In addition, at each node encountered during the search, all spanning intervals stored at the node are added to the answer. To improve the performance of the structure, the *Skeleton SR-tree* has also been proposed [7], which is an SR-tree that pre-partitions the entire domain into some number of

regions. This pre-partition is based on some initial assumption on the distribution of data and the number of intervals to be inserted. Then the Skeleton SR-tree is populated with data; if the data distribution is changed, the structure of the Skeleton SR-tree can be changed too.

When indexing valid-time intervals, overlapping may also incur if the valid-time intervals extend to the ever-increasing *now*. One approach could be to use the largest possible valid-time to represent the variable *now*. In [3] the problem of addressing both the *now* and transaction-time Until-Changed (*UC*) variables is addressed by using bounding rectangles/regions that increase as the time proceeds. A variation of the R-tree, the GR-tree is presented. More details appear in [3].

Key Applications

The importance of temporal indexing emanates from the many applications that maintain temporal data. The ever increasing nature of time imposes the need for many applications to store large amounts of temporal data. Specialized indexing techniques are needed to access such data. Temporal indexing has offered many such solutions that enable fast access.

Cross-references

- ▶ Bi-temporal Indexing
- ▶ B+-Tree
- ▶ Temporal Database
- ▶ Transaction Time
- ▶ Transaction-Time Indexing
- ▶ Valid Time

Recommended Reading

1. Arge L. and Vitter J.S. Optimal dynamic interval management in external memory. In Proc. 37th Annual Symp. on Foundations of Computer Science, 1996, pp. 560–569.
2. Blankenagel G. and Gueting R.H. External segment trees. Algorithmica, 12(6):498–532, 1994.
3. Bliujute R., Jensen C.S., Saltenis S., and Slivinskas G. R-tree based indexing of now-relative bi-temporal data. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1998, pp. 345–356.
4. Chiang Y.-J. and Silva C.T. External memory techniques for isosurface extraction in scientific visualization, external memory algorithms and visualization. In DIMACS Series in Discrete Mathematics and Theoretical Computer Science, J. Abello, J.S. Vitter (eds.). AMS, vol. 50, 1999, pp. 247–277.
5. Icking C., Klein R., and Ottmann T. Priority search trees in secondary memory. In Proc. Int. Workshop on Graph Theoretic Concepts in Computer Science, 1988, pp. 84–93.
6. Kanellakis P., Ramaswamy S., Vengroff D., and Vitter J.S. Indexing for data models with constraint and classes. In Proc. 12th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, 1993, pp. 233–243.
7. Kolovson C. and Stonebraker M. Segment indexes: dynamic indexing techniques for multi-dimensional interval data. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1991, pp. 138–147.
8. Kriegel H-P., Potke M., Seidl T. Managing intervals efficiently in object-relational databases. In Proc. 26th Int. Conf. on Very Large Data Bases, 2000.
9. Mehlhorn K. Data Structures and Efficient Algorithms, Vol. 3: Multi-dimensional Searching and Computational Geometry. EATCS Monographs, Springer, 1984.
10. Nascimento M.A. and Dunham M.H. Indexing valid time databases via B^+-Trees. IEEE Trans. Knowl. Data Eng., 11(6):929–947, 1999.
11. Ramaswamy S. and Subramanian S. Path caching: a technique for optimal external searching. In Proc. 13th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, 1994, pp. 25–35.
12. Salzberg B. and Tsotras V.J. A comparison of access methods for time-evolving data. ACM Comput. Surv., 31(2):158–221, 1999.
13. Snodgrass R.T. and Ahn I. Temporal databases. IEEE Comput., 19(9):35–42, 1986.

Value Equivalence

Nikos A. Lorentzos
Agricultural University of Athens, Athens, Greece

Definition

In temporal databases, the scheme of a temporal relation S has the form S(**E, I**) where **E** and **I** represent two disjoint sets of attributes, termed by some authors *explicit* and *implicit*, respectively. *Explicit* are the attributes in which ordinary data are recorded, such as Employee_Id, Name, Salary etc. *Implicit* are the attributes in which either *valid time* (one or more) or *transaction time* is recorded. Given two tuples $(\mathbf{e_1}, \mathbf{i_1})$ and $(\mathbf{e_2}, \mathbf{i_2})$ with the scheme of S, it is said that there is a *value equivalence* between them if and only if $\mathbf{e_1} = \mathbf{e_2}$. Equivalently, it is said that $(\mathbf{e_1}, \mathbf{i_1})$ and $(\mathbf{e_2}, \mathbf{i_2})$ are *value equivalent*.

Example: Consider one tuple, (Alex, 100, d400, d799), with scheme SALARY(Name, Amount, ValidTime), indicating that Alex's salary was 100 on each of the dates d400, d401,...,d799. Let also another tuple of the same scheme be (Alex, 100, d500, d899). For the given scheme, **E** = {Name, Amount} and

I = {ValidTime}. Given that both tuples have the same value for Name and Amount, they are value equivalent.

Key Points

The term has been introduced in the context of temporal databases, mainly in order to ease discussion on issues such as *temporal coalescing*.

Cross-references

- ▶ Period-Stamped Temporal Models
- ▶ Temporal Coalescing

Variable Time Span

Christian S. Jensen[1], Richard T. Snodgrass[2]
[1]Aalborg University, Aalborg, Denmark
[2]University of Arizona, Tucson, AZ, USA

Synonyms

Moving span

Definition

A span is *variable* if its duration is dependent on the assumed context.

Key Points

Given a specific setting, any span is either a fixed span or a variable span. An obvious example of a variable span is "1 month" in the Gregorian calendar. Its duration may be any of 28, 29, 30, and 31 days, depending on which particular month is intended. The span "1 h" is fixed because it always has a duration of 60 min.

Cross-references

- ▶ Fixed Time Span
- ▶ Temporal Database
- ▶ Temporal Granularity
- ▶ Time Interval
- ▶ Time Span

Recommended Reading

1. Bettini C., Dyreson C.E., Evans W.S., Snodgrass R.T., and Wang X.S. A glossary of time granularity concepts. In Temporal Databases: Research and Practice, O. Etzion, S. Jajodia, S. Sripada (eds.). LNCS 1399, Springer, Berlin, 1998, pp. 406–413.
2. Jensen C.S. and Dyreson C.E. (eds.). A consensus glossary of temporal database concepts – February 1998 version. In Temporal Databases: Research and Practice, O. Etzion, S. Jajodia, S. Sripada (eds.). LNCS 1399, Springer-Verlag, Berlin, 1998, pp. 367–405.

VDM

- ▶ Visual Data Mining

Vector-Space Model

Massimo Melucci
University of Padua, Padua, Italy

Synonyms

VSM

Definition

The Vector-Space Model (VSM) for Information Retrieval represents documents and queries as vectors of weights. Each weight is a measure of the importance of an index term in a document or a query, respectively. The index term weights are computed on the basis of the frequency of the index terms in the document, the query or the collection. At retrieval time, the documents are ranked by the cosine of the angle between the document vectors and the query vector. For each document and query, the cosine of the angle is calculated as the ratio between the inner product between the document vector and the query vector, and the product of the norm of the document vector by the norm of the query vector. The documents are then returned by the system by decreasing cosine.

Historical Background

The use of vectors for modeling IR systems dates back to the early days of IR, especially as a tool for describing how a system should be designed and implemented. The popularity of the VSM is due to the intuitive yet formal view of indexing and retrieval – it should not come as a surprise that the VSM has attracted many researchers and newcomers since it was mentioned in [5], employed for indexing and clustering in [9], used as a framework for deciding whether a term should be stored in an index [10], and formulated in a comprehensive way in [6]. At the experimental level, the VSM has proved

an effective and sound framework in retrieving documents in different languages, on different subjects, of different sizes, and of different media, thanks to a number of proposed and tested weighting schemes and applications (see [8] for a comprehensive evaluation). A perspective on the history of the VSM was illustrated in [2].

Despite its apparent simplicity, the mathematical properties of vector spaces can be used for advancing IR modeling. The hypothesis that the term-document frequency matrix contains information about the correlation among terms and among documents was cited in [7], stated in [13], developed in [6] and was further exploited in [1] in defining Latent Semantic Indexing (LSI). The latter is a technique based on Singular Value Decomposition (SVD) which aims at decomposing the term correlation matrix and at disclosing the principal components used to represent fewer independent concepts than many inter-dependent variables. A reconsideration of the potential of the vector spaces was presented in [12]. In that book, Hilbert's vector spaces, which is the mathematical tool of Quantum Mechanics, are used to see documents as vectors, relevance as a Hermitian operator, relevance statuses as the eigenvalues of this operator, and the computation of the probability of relevance of a document as the projection of the document vector onto the subspace of the eigenvalue of relevance. The idea of using the notion of basis of a vector space for representing context was proposed in [4].

Foundations

Basic Definitions

The VSM represents documents and queries as vectors of weights. A document vector is then

$$\mathbf{x} = (x_1,\ldots,x_k)$$

where x_i is the weight of index term i in the document and k is the number of unique index terms in the collection. Each weight is a measure of the importance of an index term in a document. A great deal of attention was paid in the past to the definition of the index term weights to be used in document vectors. An important example is

$$x_i = \mathrm{TF}_i$$

that is, the term frequency of i in the document. Of course, TF depends on the document. Another example is the binary weight,

$$x_i = \begin{cases} 1 & \text{if index term i occurs in the document} \\ 0 & \text{otherwise} \end{cases}$$

For an illustration of the term weighting schemes, see [8]. A query is represented as a vector of weights, too, for instance,

$$\mathbf{y} = (y_1,\ldots,y_k)$$

where y_i is the weight of index term i in the query. Each weight is a measure of the importance of an index term in a query – note that no Boolean operators are assumed. Also for queries, a great deal of attention has been paid in the past to the definition of the index term weights. An important example is

$$y_i = \mathrm{IDF}_i$$

where IDF stands for Inverse Document Frequency which is defined as

$$\mathrm{IDF}_i = \begin{cases} \log \frac{N}{n_i} & \text{if } i \text{ occurs in the query} \\ 0 & \text{otherwise} \end{cases}$$

where n_i is the number of documents indexed by i and N is the total number of documents in the collection. The binary weighting scheme can also be applied to queries.

At retrieval time, the documents are ranked by the cosine of the angle between the document vectors and the query vector. For each document and query, the cosine of the angle is calculated as the ratio between the inner product between the document vector and the query vector, and the product of the norm of the document vector by the norm of the query vector. The documents are then returned by the system by decreasing cosine. In a formal way,

$$\cos = \frac{\mathbf{x} \cdot \mathbf{y}}{||\mathbf{x}||\ ||\mathbf{y}||}$$

where

$$\mathbf{x} \cdot \mathbf{y} = \sum_{i=1}^{k} x_i\ y_i \qquad ||\mathbf{x}|| = \sqrt{\sum_{i=1}^{k} x_i^2}$$

$$||\mathbf{y}|| = \sqrt{\sum_{i=1}^{k} y_i^2}$$

When the weights are binary, i.e., $x_i, y_i \in \{0, 1\}$, the vectors are representations of term sets and an

alternative ranking function is Jaccard's coefficient defined as

$$\frac{\sum_{i=1}^{k} x_i \; y_i}{\sum_{i=1}^{k} x_i + \sum_{i=1}^{k} y_i - \sum_{i=1}^{k} x_i \; y_i}$$

When term frequency and inverse document frequency are used for weighting terms in the document and in the query, respectively the inner product between document and query vectors is

$$\sum_{i=1}^{k} \mathrm{TF}_i \; \mathrm{IDF}_i$$

where

$$\mathrm{TF}_i \; \mathrm{IDF}_i$$

is called TF-IDF weighting scheme. When different documents are considered, the notations $\mathbf{x}_n = (x_{1,m},\ldots, x_{k,n})$ and $\mathrm{TF}_{i,n}$ IDF_i are used for distinguishing documents $n = 1,\ldots,N$.

Mathematical Development

The VSM for IR considers a document collection as a vector space defined over the real field. The definition of a vector is given with respect to a basis, namely, a set of linearly independent vectors, called basis vectors – a set of vectors is independent when no vector of this set can be defined as a linear combination of the other vectors in the same set. The dimension of the space is the number of basis vectors. It is a fact from Linear Algebra that every vector of a space of dimension k is a linear combination of k basis vectors – for an introduction to the vector spaces see, for example, [3].

The VSM represents a collection index as a basis of a real vector space and a distinct index term is a basis vector. Moreover, every object managed by an IR system, e.g., document, query, sentence, cluster centroids, or terms, is a vector of this space. As every vector is a linear combination of a basis of the vector space, in the VSM the vector of any object is a linear combination of the basis vectors that represent the index terms. When non-textual media are to be represented, the basis vectors are a representation of content descriptors, yet the philosophy of the model remains unchanged. A basis plays an important role in expressing term correlations and in Latent Semantic Indexing.

Since there are infinite bases in a space, every vector has infinite numerical representations. To be precise, let k be the dimension of the space and $\{\mathbf{t}_1,\ldots,\mathbf{t}_k\}$ be a basis for this space. The representation of a document or a query is then a vector $\mathbf{x}$ such that

$$\mathbf{x} = \sum_{i=1}^{k} a_i \; \mathbf{t}_i.$$

where the a_i are real numbers. The numerical representations are provided by the a_i's with respect to the basis $\{\mathbf{t}_1,\ldots,\mathbf{t}_k\}$. This entails that a change of basis determines a change of the numerical representation of $\mathbf{x}$; for example,

$$\mathbf{x} = \sum_{i=1}^{k} b_i \; \mathbf{u}_i,$$

where $\{\mathbf{u}_1,\ldots,\mathbf{u}_k\}$ is a different basis and the b_i's provide a different representation of $\mathbf{x}$ – it follows that a vector can be assigned infinite representations, all being equivalent.

According to the basic definitions, the a_i's are the weights used for measuring the importance of the index terms in documents and queries. Thus, TF-IDF is a possible scheme for defining the a_i's.

The ranking function is based on the cosine of the angle between the query vector $\mathbf{y}$ and a document vector $\mathbf{x}$. The documents are then ranked by (decreasing) cosine with respect to the query. Formally, the cosine of the angle between the two aforementioned vectors is:

$$\frac{\mathbf{x}^\top \cdot \mathbf{y}}{||\mathbf{x}|| \; ||\mathbf{y}||}$$

where (The vectors are assumed to be column vectors and the symbol $^\top$ denotes transposition from a column (row) vector to a row (column) vector.):

$$\mathbf{x} = \sum_{i=1}^{k} a_i \; \mathbf{t}_i \qquad \mathbf{y} = \sum_{j=1}^{k} c_j \; \mathbf{t}_j \qquad \mathbf{x}^\top \cdot \mathbf{y} = \sum_{l=1}^{k} x_l \; y_l$$

After a few simple passages,

$$\mathbf{x}^\top \cdot \mathbf{y} = \sum_{i=1}^{k} \sum_{j=1}^{k} a_i \; c_j \; \mathbf{t}_i^\top \cdot \mathbf{t}_j$$

where $\mathbf{t}_i^\top \cdot \; \mathbf{t}_j$ is a measure of correlation between the basis vectors. The cosine is used as ranking function instead of the inner product at the numerator

because the norms at the denominator of the formula make document ranking independent of document size. In contrast, the inner product would place long documents at the top ranks, while relevant, short documents would be penalized; however, see [11]. When the index terms are assumed to be uncorrelated, the basis vectors are orthogonal, namely,

$$\mathbf{t}_i^\top \cdot \mathbf{t}_j = \begin{cases} ||\mathbf{t}_i||^2 & i = j \\ 0 & i \neq j \end{cases}$$

It is customary to assume that the basis vectors are orthonormal, therefore,

$$\mathbf{t}_i^\top \cdot \mathbf{t}_j = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$$

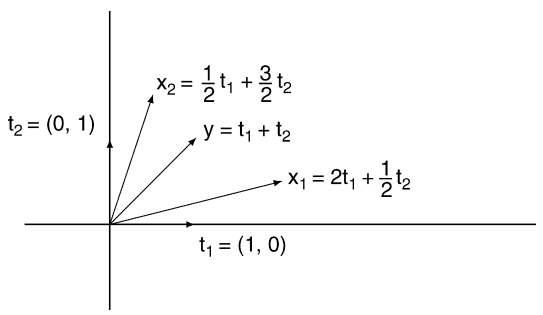
In practice, the basis is the canonical basis of a vector space, that is,

$$t_{i,j} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$$

where $t_{i,j}$ is the jth element of $\mathbf{t}_i$. In other words, the ith canonical basis vector has null elements except the ith element, which is 1. It follows that

$$\mathbf{x} = (a_1,\ldots,a_k)^\top \quad \mathbf{y} = (c_1,\ldots,c_k)^\top \quad \mathbf{x}^\top \cdot \mathbf{y} = \sum_{l=1}^{k} a_l\, c_l$$

when the canonical basis is used. Figure 1 gives a pictorial description of the VSM. The orthonormal basis ($\mathbf{t}_1$,$\mathbf{t}_2$) spans a two-dimensional spaces and every vector is thena linear combination of the basis vectors. In particular, $\mathbf{x}_1$ represents a document, $\mathbf{x}_2$ another document and $\mathbf{y}$ represents a query. After computing the cosine of the angle between the $\mathbf{x}_i$'s and $\mathbf{y}$,



Vector-Space Model. Figure 1. A pictorial description of the VSM.

$$\frac{\mathbf{y}^\top \cdot \mathbf{x}_1}{||\mathbf{y}||\ ||\mathbf{x}_1||} = 0.86 \qquad \frac{\mathbf{y}^\top \cdot \mathbf{x}_2}{||\mathbf{y}||\ ||\mathbf{x}_2||} = 0.89$$

If the basis is not orthogonal, for example, $\mathbf{t}_1 = (1, 0)^\top$ and $\mathbf{t}_2 = \left(\frac{1}{\sqrt{5}}, \frac{2}{\sqrt{5}}\right)^\top$, then, the numerical representation of the vectors and the cosines change; for example, the numerical representation of $\mathbf{x}_1$ is $\left(-\frac{1}{4}, \frac{5}{3}\right)^\top$.

Non-orthogonality has been employed for modeling relationships between index terms; for example, $\mathbf{t}_i^\top \cdot \mathbf{t}_j > 0$ for modeling that index term i is somehow related to index term j such that when i occurs, j also tends to occur – a negative value would model an inverse relationship. As a special, important case, non-orthogonality provides a principled way for modeling query expansion. Indeed, the cosine of the angle between $\mathbf{x}$ and $\mathbf{y}$ is in general affected by index term i when $a_i \neq 0$ or $c_i \neq 0$ – when the basis is orthogonal, the cosine is affected by i when $a_i \neq 0$ or $c_i \neq 0$ that is, when the index term occurs both in the document and in the query. Suppose, for example, k = 2 and the query contains term 1 only, that is, $c_1 = 1$ and $c_2 = 0$. Therefore,

$$\mathbf{x}^\top \cdot \mathbf{y} = a_1\ \mathbf{t}_1^\top \cdot \mathbf{t}_1\ + a_2\ \mathbf{t}_2^\top \cdot \mathbf{t}_1$$

If $\mathbf{t}_2^\top \cdot \mathbf{t}_1 = 0.5$, that is, the terms are partially correlated, the distance between the document and the query is also affected by term 2, even though it does not occur in the query.

Latent Semantic Indexing (LSI) is a technique based on Singular Value Decomposition (SVD) which aims at finding a basis alternative to the canonical basis. The documents and the queries are then re-expressed with respect to this new basis. The potential of LSI is twofold. First, the document and the queries are vectors of a new space whose dimension is greatly reduced. Second, the new basis vectors no longer represent index terms, but sets of index terms that may be called "concepts," "clusters" or "aggregates" depending on the application domain.

Key Applications

The VSM has been subjected to an extensive evaluation and SMART is the best known experimental IR system. Thanks to the excellent results confirmed by several tests carried out within the Text Retrieval Conference (TREC), this model is presently the reference model for many applications whenever a best-match and weighted retrieval is required. The applications ranges in many tasks, including clustering, cross-language

retrieval, summarization and personalization to name but a few. Salton and his colleagues' bibliography gives an idea of the broad range of applications.

Future Directions

The most challenging problem of Information Retrieval is to define a model that governs the complex interactions between the various components of a systems and the end users. An attempt in this direction has been made in [12], which has paved the way toward a new conception of the vector spaces and opened up some connections to other sciences.

Cross-references

- ▶ Boolean Model
- ▶ Clustering for Post Hoc Retrieval
- ▶ Indexing Units
- ▶ Probabilistic Retrieval Models and Binary Independence Retrieval (BIR) Model
- ▶ Query Expansion Models
- ▶ Relevance Feedback
- ▶ Singular Value Decomposition
- ▶ Term Weighting

Recommended Reading

1. Deerwester S., Dumais S., Furnas G., Landauer T., and Harshman R. Indexing by latent semantic analysis. J. Am. Soc. Inform. Sci., 41(6):391–407, 1990.
2. Dubin D. The most influential paper Gerard Salton never wrote. Libr. Trends, 52(4):748–764, 2004.
3. Halmos P. Finite-Dimensional Vector Spaces. Undergraduate Texts in Mathematics, Springer, 1987.
4. Melucci M. A basis for information retrieval in context. ACM Trans. Inform. Syst., 26(3), 2008.
5. Salton G. Associative document retrieval techniques using bibliographic information. J. ACM, 10440–457, 1963.
6. Salton G. Automatic Text Processing. Addison-Wesley, 1989.
7. Salton G. Mathematics and information retrieval. J. Doc., 35(1):1–29, 1979.
8. Salton G. and Buckley C. Term Weighting Approaches in Automatic Text Retrieval. Inform. Process. Manage., 24(5):513–523, 1988.
9. Salton G., Wong A., and Yang C. A vector space model for automatic indexing. Commun. ACM, 18(11):613–620, 1975.
10. Salton G., Yang C., and Yu C. A theory of term importance in automatic text analysis. J. Am. Soc. Inform. Sci., 26(1):33–44, 1975.
11. Singhal A., Buckley C., and Mitra M. Pivoted Document Length Normalization. In Proc. 19th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 1996. pp. 21–29.
12. van Rijsbergen C. The Geometry of Information Retrieval. Cambridge University Press, UK, 2004.
13. Wong S. and Raghavan V. Vector space model of information retrieval – a reevaluation. In Proc. 7th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 1984, pp. 167–185.

Vertical Fragmentation

▶ Distributed Database Design

Vertically Partitioned Data

Jaideep Vaidya
Rutgers University, Newark, NJ, USA

Synonyms

Heterogeneously distributed data

Definition

Data is said to be vertically partitioned when several organizations own different attributes of information for the same set of entities. Thus, vertical partitioning of data can formally be defined as follows: First, define a dataset D in terms of the entities for whom the data are collected and the information that is collected for each entity. Thus, $D \equiv (E, I)$, where E is the entity set for whom information is collected and I is the feature set that is collected. Assume that there are k different sites, $P_1,...,P_k$ collecting datasets $D_1 \equiv (E_1, I_1),...,D_k \equiv (E_k,I_k)$ respectively. Therefore, data is said to be vertically partitioned if $E = \bigcap_i E_i = E_1 \bigcap ... \bigcap E_k$, and $I = \bigcup_i I_i = I_1 \bigcup ... \bigcup I_k$. In general, distributed data can be arbitrarily partitioned. Vertical partitioning can also be defined as a special case of arbitrary partitioning, where all of the partitions consist of information about the same set of entities.

Key Points

In general, data can be distributed in an arbitrary fashion. This means that different parties may own partial information about different sets of entities. While such arbitrary partitioning is possible, in practice, it rarely happens. Data is said to be vertically partitioned when different sites collect different

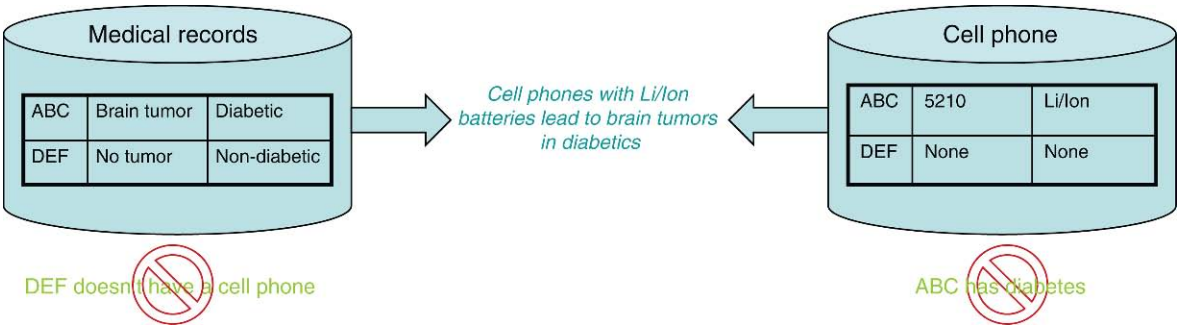
features of data for the same set of entities. Integrating the local datasets gives the global dataset. Vertically partitioned data occurs naturally in many situations, and mining it can lead to unexpected insights. For example, consider the case of Ford and Firestone. Ford collects information about vehicles manufactured. Firestone collects information about tires manufactured. Vehicles can be linked to tires. This linking information can be used to join the databases. Why bring this up? In 2001, numerous accidents due to tread separation were reported. Initially both companies blamed each other. It turned out that it was only Ford Explorers with Firestone tires from the Decatur, Illinois plant, in specific situations that had these problems. If found out earlier, much loss could have been avoided. While both companies individually collect a lot of pertinent testing data, this was not shared due to commercial concerns. Mining of the global database had the potential to reveal this, which was otherwise impossible only with the local data.

Figure 1 demonstrates another example of vertical partitioning of data. There are two data sources – a hypothetical hospital/insurance company and a wireless service provider. The hospital collects medical records such as the type of brain tumor and diabetes (none if the person does not suffer from the condition). On the other hand, the wireless provider might collect other information such as the approximate amount of airtime used everyday, the model of the cellphone and the kind of battery used. Together, merging this information for common customers and running data mining algorithms might give completely unexpected correlations (for example, a person with Type I diabetes using a cell phone with Li/Ion batteries for more than 3 hours per day is very likely to suffer from primary brain tumors.). It would be impossible to get such information by considering either database in isolation. While this example is hypothetical, similar situations abound in practice.

In general, when data is vertically partitioned, more data significantly improves the quality of the models built from the dataset. Overall, the data analysis results are significantly more accurate and real. While this is also true of horizontally partitioned data (more data is always good), but it has a more critical impact with vertically partitioned data. This is because data from different parties give significantly different additional information about the entities. This is especially true with higher dimensional data.

The complexity of privacy-preserving data mining is also significantly increased due to the vertical partitioning of data. In contrast to horizontal partitioning of data, vertical partitioning of data raises several unique questions with respect to the way data is processed, results are obtained and shared. For example, consider the task of creating a decision tree classifier from some given vertically partitioned dataset. An immediate question is how is the final tree shared by the different parties? Since each party has knowledge of only some of the attributes, knowing the structure of the tree, and especially, knowledge of an unknown attribute and its breakpoints for testing – constitutes a violation of the privacy of the individual parties. Thus, completely revealing the tree is a bad idea from the security standpoint.

Indeed, in the best case, for no leakage of information, even the structure of the tree should be hidden, with an oblivious protocol for classifying a new instance. However, the cost associated with this is typically unacceptable. A compromise may be to



Vertically Partitioned Data. Figure 1. A vertically partitioned dataset with hypothetical knowledge inferred from it.

cloak the attribute tests used in the tree while still revealing the basic structure of the tree. This way each site only need know the branch values for the decisions it makes, and the classification of a new instance would also have to be carried out in a distributed fashion by jumping from site to site. In general, it is necessary to carefully evaluate what may be the intermediate and final results, and only reveal as appropriate. Indeed, constructing the form of results is one of the major challenges, especially when one considers that multiple analyses may need to be carried out. [1] provides more details on problems, solutions and challenges in this area.

Cross-references

- ► Horizontally Partitioned Data
- ► Privacy-Preserving Data Mining
- ► Secure Multiparty Computation Methods

Recommended Reading

1. Vaidya J., Clifton C., and Zhu M. Privacy-Preserving Data Mining, 1st edn. Advances in Information Security 19, Springer-Verlag, Berlin, 2005.

Video

Ying Li
IBM T.J. Watson Research Center, Hawthorne, NY, USA

Synonyms

Rich media; Multimedia

Definition

Video, which means "I see" in Latin, is an electronic representation of a sequence of images or frames, put together to simulate motion and interactivity. From the producer's perspective, a video delivers information created from the recording of real events to be processed simultaneously by a viewer's eyes and ears. For most of time, a video also contains other forms of media such as audio.

Video is also referred to as a storage format for moving pictures as compared to image, audio, graphics and animation.

Historical Background

Video technology was first developed for television systems, but it has been further developed in many formats to allow for consumer video recordings. Generally speaking, there are two main types of video: analog video and digital video. Analog videos are usually recorded as PAL (Phase Alternating Line) or NTSC (National Television System Committee) electric signals following the VHS (Video Home System) standard, and stored in magnetic tapes. Digital videos, on the contrary, are usually captured by digital cameras and stored in digital video formats such as DVD (Digital Versatile Disc), QuickTime and MPEG-4 (Moving Picture Experts Group).

Launched in September 1976, VHS became a standard format for consumer recording and viewing by the 1990s. Since then, it has dominated both home and commercial video markets. In March 1997, the DVD format was introduced to American consumers, which gradually pulled consumers away from VHS in following years due to its much better quality. In June 2003, the DVD's market share exceeded that of the VHS for the first time. Since then, it has been steadily expanding its consumer market, and by July 2006, most major film studios have stopped releasing new movie titles in VHS format, opting for DVD-only releases. Now, VHS is gradually disappearing from both rental and retail stores, and DVD has dominated the whole commercial market. Nevertheless, VHS is still popular for home recording of television programs, due to the large installed base and the lower cost of VHS recorders and tape.

For the last few decades, as video technology quickly advances and the cost of storage devices rapidly decreases, digital videos have become widely available in diverse application areas such as medicine, remote sensing, entertainment, education and online information services. This has thus led to very active researches in various video-related areas.

Foundations

The last three decades have witnessed a significant amount of research efforts on various aspects of video technologies. Roughly speaking, they fall into the following three general categories: video representation, video content analysis, and video application. Specifically, video representation deals with the way a video is represented, in another word, the file format. Video content analysis, on the other hand, aims to automatically structurize and ultimately understand the video by analyzing

its underlying content. Due to the difficult nature of this problem, such process usually involves the analysis of multiple media modalities including visual, audio and text information. Finally, video application applies what's learned from the analysis engine, and facilitates various types of content access including video browsing, summarization and retrieval. A brief discussion on each of these three research domains is given below.

Video Representation

A video sequence with accompanying sound track can occupy a vast amount of storage space when represented in digital format. As estimated in [6], a 1-min video clip could possibly occupy up to 448 MB. Consequently, compression has been playing an important role in modern schemes for video representation.

A wide variety of methods have been proposed to compress video stream. Nevertheless, almost all of them build their approaches upon the fact that video data contains both spatial and temporal redundancy. Specifically, to reduce the spatial redundancy, an intra-frame compression is applied which registers differences between parts of a single frame. Such a task is more closely related to image compression. Likewise, to reduce the temporal redundancy, an inter-frame compression is exploited which registers differences between neighboring frames. This involves discrete Cosine transform (DCT), motion compensation and other techniques.

Some popular video compression mechanisms include H.261, H.263, H.264, MPEG-1, MPEG-2, MPEG-4 and MJPEG (Motion-Joint Photographic Experts Group). Specifically, H.261 is a 1990 ITU-T (Telecommunication Standardization Sector of International Telecommunication Union) video coding standard originally designed for transmission over ISDN lines. Later on, H.263 and H.264 which provide more capabilities and mainly target at video-conferencing applications, were standardized in 1995 and 2003, respectively.

In 1998, the Moving Picture Experts Group (MPEG) was formed to establish an international standard for the coded representation of moving pictures and associated audio on digital storage media. Currently, there have been three established MPEG standards from this effort: MPEG-1, MPEG-2, and MPEG-4. Each of them targets at different commercial applications. For instance, MPEG-1 is usually used as the Video CD (VCD) format, MPEG-2 for High Definition Television (HDTV), and MPEG-4 for streaming video applications.

Finally, to facilitate mobile appliances such as digital cameras, MJPEG was developed in 1990s which uses intra-frame coding technology that is very similar to those used in MPEG-1 or MPEG-2. However, it does not use inter-frame prediction, which on one hand, results in a loss of compression capability, yet on the other hand, it makes the degree of compression capability independent of the amount of motion in the scene. Moreover, it also eases video editing as simple edits can now be performed at any frame.

Video Content Analysis

Video is a type of rich media as it often consists of other media types such as audio and text. Consequently, research on video content analysis can be grouped into three classes: visual content analysis, audio content analysis, and audiovisual content analysis. A general goal of video content analysis is to extract the underlying video structure so as to facilitate convenient and nonlinear content access. Yet a more aggressive goal is to automatically understand video semantics so as to support applications such as video summarization and retrieval that require an in-depth understanding of the video content.

Visual Content Analysis

As the name implies, visual content analysis concentrates on processing the visual part of the video signal. Most existing work in this area falls into two directions, with one focusing on analyzing the image or frame content in the spatial domain, and the other on exploiting the temporal relationships between frames.

Visual feature extraction at different semantic levels are usually the major focus of those work in the first category. Popular features such as color, texture and motion, are low-level features, which while easy to extract, are not able to capture the video semantics. Therefore, most of recent work focuses on extracting mid to high level visual features, which can not only be derived from lower level features, but also capable of bridging the semantic gap by revealing the underlying content semantics to a certain degree. Such features include various types of video objects and semantic concepts such as sports, military, explosion and sky. Machine learning and content modeling are two popularly applied approaches to achieve such goal.

On the other hand, there is also a significant amount of work that attempts to capture both video syntax and semantics by exploring its temporal structure. Specifically, video shot change detection is usually the very first step towards this goal, where the entire video sequence is segmented into a series of cascaded shots. Shot forms the building block of a video sequence, and contains a set of frames that are continuously taken. For a comprehensive survey on various shot detection algorithms, readers can refer to [3].

Based on the syntactic shot structure, higher-level visual content analysis such as scene detection or extraction, could be subsequently carried out. Figure 1 shows a typical video content structure, which is represented by a hierarchical tree. As shown, for an incoming video stream, it is first segmented into a sequence of *shots*. Then, one or more of its frames (called *keyframes*) can be extracted to represent its underlying content. The next step is to build semantic video scenes upon the extracted shot structure. Specifically, a video *scene* is defined as a collection of semantically related and temporally adjacent shots that depicts and conveys a high-level concept or story. Consequently, a common solution to scene extraction is to group semantically related shots into a scene [14]. This process is similar to grouping words into sentences, where a sentence starts conveying meanings to readers. Finally, there could be one or more steps further from scene detection. One such effort is on event detection, where an *event* is considered as important scenes that contain particular thematic topics of interest such as dialogs, actions or something abnormal [3,5].

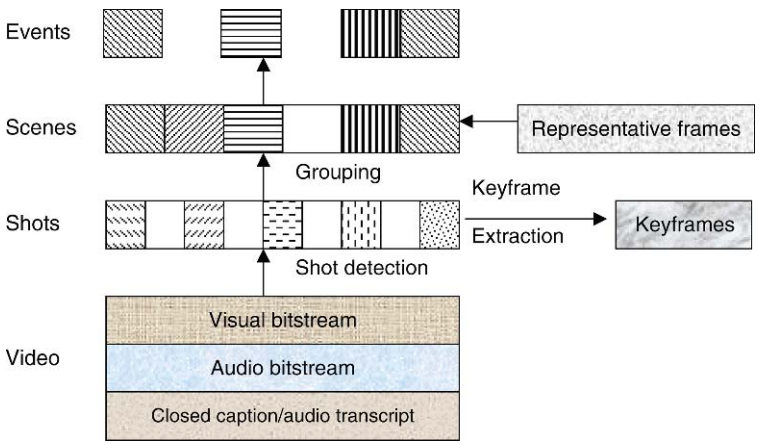
Audio Content Analysis

Audio, which includes voice, music and various kinds of environmental sounds, is an important type of media. A general approach to audio content analysis is to either classify or segment an audio clip into different homogeneous fragments with each fragment containing one particular audio type such as speech, music or silence [2]. Audio features such as short-time energy, short-time zero-crossing rate and short-time fundamental frequency, are usually extracted for this analysis purpose [15].

As a key ingredient of a video stream, the audio source could represent the story in a much simpler fashion than its pictorial counterpart. For instance, if people are only allowed to listen to a TV program without watching it, they can still get most of its meanings. Yet on the other hand, if they are only allowed to watch the program without listening to it, they may get lost easily. This shows that the audio source, especially the embedded speech cue, is critical to a human's content understanding. Moreover, audio is also essential to people's enjoyment of the video content.

Audiovisual Content Analysis

As automatic video content understanding is a very challenging task, thus most recent work tends to exploit all possible media cues to achieve such a goal. In particular, as audio and visual cues are two inseparable parts of a video stream, and usually complement with each other during the content creation, people tend to integrate them together during the content analysis.



Video. Figure 1. A hierarchical representation of video content.

A general solution to audiovisual content analysis is to first perform individual visual or audio content analysis to obtain two sets of analysis results, then combine them together using certain fusion rules or applying a probabilistic framework. Another popular way of media integration is to employ different information sources at different processing stages [4]. For instance, the visual cue is first employed to generate coarse-level results, then audio cues is introduced to refine the results. However, when and how to efficiently and effectively integrate multiple media sources still remains to be an open issue, and needs further study.

Video Application

Besides the large amount of research efforts on video content analysis, there are also many attentions on studying various video applications. After all, making the bulky and unstructured video content convenient and efficient to access, present, share, search and deliver is the ultimate goal of the entire research community in this area. Below, a brief introduction is given to three major types of video applications, namely, video indexing and browsing, video abstraction, and content-based video retrieval.

Video Indexing and Browsing

As analog to indexing a book, video indexing aims at facilitating non-linear access of the video content. It is thus very critical for efficient video retrieval. For instance, television and film archives usually contain a vast amount of audiovisual materials. If these materials are properly segmented and indexed, studios can conveniently produce a new video clip by finding and reusing some pre-produced segments. Moreover, in audiovisual libraries or family entertainment applications, it would be very desirable if the needed video segments from a large video collection could be quickly located.

Video browsing refers to the activity where a user watches through a video to get quick ideas of its underlying content [11]. Video browsing can also assist users in query forming for video retrieval purpose. For instance, when the user lacks a clear idea of what he wants from a large video collection, he can gradually conceptualize his needs by browsing through the video clips to find the one that stimulates his desire. Once such clip is located, he can further submit it as a query seed to obtain more related clips.

Video Abstraction

Video abstraction, as the name implies, generates a short summary for a long video document. Specifically, a video abstract is a sequence of still or moving images, representing the video content in a way such that the target party is rapidly provided with concise information about the content, while the essential message of the original is well preserved [12]. Video abstraction is primarily used for video browsing, and is an inseparable part of a video indexing and retrieval system.

Theoretically, a video abstract can be generated both manually and automatically, but due to the huge volumes of video data and limited human power, it is getting increasingly important to develop fully automated video analysis and processing tools so as to reduce human involvement in the video abstraction process.

There are two fundamentally different kinds of video abstracts: still- and moving-image abstracts. The still-image abstract, also known as static storyboard or *video summary*, is a small collection of salient images (also called keyframes) extracted or generated from the underlying video source. The moving-image abstract, also known as moving storyboard or *video skim*, consists of a collection of image sequences, as well as the corresponding audio abstract extracted from the original sequence. Thus, it is itself a video clip but is of a considerably shorter length.

There exist some significant differences between video summary and video skim. First, a video summary can be built much faster, since generally only visual information is utilized and no handling of audio and textual information is needed. Also, once composed, it can be displayed more easily since there are no timing or synchronization issues. Second, more salient images such as mosaics could be generated to better represent the underlying video content. Third, the temporal order of all extracted representative frames can be displayed in a spatial order so that the users are able to quickly grasp the video content. Finally, all extracted stills could be easily printed out when needed.

There are also advantages in using video skim. Compared to the still-image abstract, a moving-image abstract usually makes more sense to users since it preserves part of the original audio track. Moreover, the possibly higher computational effort during the abstracting process pays off during the playback time: it is usually more natural and more interesting for users to watch a trailer than watching

a slide show, and in many cases, the motion is also information-bearing.

A comprehensive survey of various video summarization and video skimming techniques could be found in [3].

Content-Based Video Retrieval

By definition, a content-based video retrieval system (CBVRS) aims at assisting a human operator or user to retrieve a video sequence within a potentially large collection based on certain criteria. Generally speaking, there are three major aspects regarding a CBVRS, namely, *query formation*, *feature space selection*, and *similarity measurement*. In particular, the query formation deals with formulating a meaningful and clear query which faithfully captures what the user is looking for. Once the query is submitted, potential candidates will be evaluated within certain feature space, and the similarity between the query and candidate is measured. Such a process usually results in a ranked list of retrieved sequences sorted from the most similar to the least. Below gives a brief discussion on each of the above three aspects.

To formulate a query, the user can either submit still images (*a.k.a.* keyframes), short video sequence, textual keywords, or a combination of the above as the search seeds. Query formation is not as easy a task as it appears to be. One major reason for this is due to the semantic gap between the user's true intention and the constrained way of representing it. For instance, when the user submits a video sequence as a query example, he or she may mean to find some videos that present similar color schemes, motion, events, or objects, yet such intention is hard to be captured or understood by the system as it is embedded in the query and is highly subjective. To close such a semantic gap, one way is to use textual keywords in queries that precisely describe what the user is looking for. Nevertheless, this would require a thorough and correct textual annotation of the entire collection, which is not only very tedious but in fact, impossible for humans to perform. Recently, there have been some ongoing research efforts on automatic video annotation [10], yet many challenging issues still remain to be solved.

As another effort on bridging such a semantic gap, people have been introducing the so-called relevance feedback mechanism into the system, where the user interacts with the system by giving negative or positive feedback on the retrieved results [13]. The system then performs an online learning of the user's expectations by possibly adjusting or updating the weights of different features, the similarity metrics, and the learning algorithms. While there has been an increasing amount of work on this subject, how to design an efficient, effective and user-friendly relevance feedback system still deserves further study.

Once the query is formulated and submitted, the next step is to measure the similarity between the query example and possible candidates within certain feature space. Undoubtedly, finding a good set of features to represent the query is very critical to the success of the search. So far, various features at different semantic levels have been proposed including color, texture, motion, audio, object, scene, event etc. in both spatial and temporal domains [16].

Finally, a metric is required to measure the similarity, which produces a distance that would account for both the spatial and temporal differences between the two video sequences. Ideally, such metric should be jointly defined with the particular feature set that is applied in the system for a better result [1].

In a word, while at a first glance, a content-based video retrieval system is a natural extension of a content-based image retrieval system, it is in fact, far more complex due to the excessive dimensionality of the search space induced by the inherent temporal information in videos.

Key Applications

In October 1998, MPEG started a new work item called the "*Multimedia Content Description Interface*," or in short "MPEG-7," which aims to specify a standard set of descriptors and description schemes that can be used in describing various types of multimedia information [9]. This description shall be associated with the content itself, to allow fast and efficient search for materials of users' interests. The major objective of MPEG-7 is to make audiovisual material as searchable as text [8].

Various cases of professional and consumers applications have been identified and targeted by MPEG-7, which can be categorized into either *pull* or *push* applications [7]. Specifically, typical pull applications that are more related to video include: (i) storage and retrieval of video databases; (ii) delivery of pictures and video for professional media production, and

(iii) movie scene retrieval by memorable auditory events. In contrast, push applications follow a paradigm more akin to broadcasting and webcasting. Some key applications include: (i) user agent driven media selection and filtering; (ii) personalized television services; (iii) intelligent multimedia presentation; (iv) bio-medical applications; (v) remote sensing applications; (vi) semi-automated multimedia editing; (vii) educational applications, and (viii) surveillance applications.

Undoubtedly, with the amount of audiovisual information rapidly increasing and becoming widely available from many sources around the world, a lot more new applications will emerge so as to satisfy various urgent needs related to important academic, social and economic issues.

Cross-references

- ▶ Audio Content Analysis
- ▶ Content-Based Video Retrieval
- ▶ Image Retrieval
- ▶ Image Retrieval and Relevance Feedback
- ▶ Image Content Modeling
- ▶ Mid-to-High-Level Image Content Analysis
- ▶ Video Content Analysis
- ▶ Video Content Modeling
- ▶ Video Content Structure
- ▶ Video Representation
- ▶ Video Scene and Event Detection
- ▶ Video Segmentation
- ▶ Video Shot Detection
- ▶ Video Summarization
- ▶ Visual Content Analysis

Recommended Reading

1. Cheung S. and Zakhor A. Efficient video similarity measurement with video signature. IEEE Trans. Circ. Syst. Video Tech., 13(1):59–74, 2003.
2. Li Y. and Dorai C. SVM-based audio classification for instructional video analysis. In Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing, 2004.
3. Li Y. and Kuo C.-C. Video Content Analysis Using Multimodal Information: for Movie Content Extraction, Indexing and Representation. Kluwer, MA, USA, 2003.
4. Li Y., Narayanan S., and Kuo C.-C. Content-based movie analysis and indexing based on audiovisual cues. IEEE Trans. Circ. Syst. Video Tech., 14(8):1073–1085, 2004.
5. Mahmood T.S. and Srinivasan S. Detecting topical events in digital video. In Proc. 8th ACM Int. Conf. on Multimedia, 2000, pp. 85–94.
6. Mitchell J., Pennebaker W., Fogg C., and LeGall D. MPEG Video Compression Standard. Chapman & Hall, New York, NY, USA, 1992.
7. MPEG Requirements Group, MPEG-7 Applications Document v.8, ISO/MPEG N2860, MPEG Vancouver Meeting, July 1999.
8. MPEG Requirements Group, MPEG-7 Context, Objectives and Technical Roadmap, ISO/MPEG N2861, MPEG Vancouver Meeting, July 1999.
9. MPEG Requirements Group, MPEG-7 Requirements Document V.15, ISO/MPEG N4317, MPEG Sydney Meeting, July 2001.
10. Nock H., Adams W., Iyengar G., Lin C., Naphade M., Neti C., Tseng B., and Smith J. User-trainable video annotation using multimodal cues. In Proc. 26th Annu. Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 2003, pp. 403–404.
11. Oh J. and Hua K. Efficient and cost-effective techniques for browsing and indexing large video databases. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2000, pp. 415–426.
12. Pfeiffer S., Lienhart R., Fischer S., and Effelsberg W. Abstracting digital movies automatically. J. Vis. Comm. Image Represent., 7(4):345–353, 1996.
13. Yan R., Hauptmann A., and Jin R. Negative paeudo-relevance feedback in content-based video retrieval. In Proc. 11th ACM Int. Conf. on Multimedia, 2003, pp. 343–346.
14. Yeung M., Yeo B., and Liu B. Extracting story units from long programs for video browsing and navigation. In Proc. Int. Conf. on Multimedia Computing and Systems, 1996, pp. 296–305.
15. Zhang T. and Kuo C.-C. Audio content analysis for on-line audiovisual data segmentation. IEEE Trans. Speech Audio Process., 9(4):441–457, 2001.
16. Zheng W., Li J., Si Z., Lin F., and Zhang B. and Using high-level semantic features in video retrieval. In Image and Video Retrieval. Springer, Berlin Heidelberg, New York, 2006, pp. 370–379.

Video Abstraction

- ▶ Video Summarization

Video Analysis

- ▶ Video Content Analysis

Video Annotation

- ▶ Video Metadata

Video Chaptering

► Video Segmentation

Video Compression

► Video Representation

Video Content Analysis

Alexander Hauptmann
Carnegie Mellon University, Pittsburgh, PA, USA

Synonyms

Video analysis; Video content processing; Semantic analysis of video

Definition

Video content analysis deals with the extraction of metadata from raw video to be used as components for further processing in applications such as search, summarization, classification or event detection. The purpose of video content analysis is to provide extracted features and identification of structure that constitute building blocks for video retrieval, video similarity finding, summarization and navigation. Video content analysis transforms the audio and image stream into a set of semantically meaningful representations. The ultimate goal is to extract structural and semantic content automatically, without any human intervention, at least for limited types of video domains. Algorithms to perform content analysis include those for detecting objects in video, recognizing specific objects, persons, locations, detecting dynamic events in video, associating keywords with image regions or motion patterns, identifying visible actions or behaviors, and labeling scenes, activities, genres. The analysis may be performed on single frames (images), sequences of frames indicating change or motion by the camera or the subject, audio analysis through speech recognition and non-speech sound characterization as well as any combinations of still image analysis, image sequence analysis and audio analysis.

Historical Background

Digital video has proliferated dramatically since the 1990s, ranging from ever-growing personal video collections to professional news and documentary archives. As bandwidth and disk space accessible to users is increasing, video is becoming the most rapidly proliferating type of data on the Internet. Fueled by the popularity of social media sharing in the Web 2.0 paradigm, there has been exponential increase in videos available on the internet. With the availability of large amounts of multimedia data, there comes an urgent need for good video content analysis.

The first attempts at image and video content analysis were made in the 1960s and 1970s, with applications mostly in video compression and videophones. By the mid 1990s, a number of research systems that could perform video content analysis on sequences longer than a few minutes were introduced. With the widespread introduction of fast computers in the twenty-first century, digital video content analysis has become common, but is frequently not accurate enough for widespread commercial use, and the more sophisticated approaches are still too computationally complex to be applied to every frame in the video.

Video content analysis has become the only feasible way to analyze larger video collections for search, summarization, navigation, and re-use in other applications. Social media tagging has allowed collections on the internet to be searchable through text tags and comments, but these searches rely on precision, and reasonable tagging, which is often not the case, especially for some of the less popular video files.

Foundations

Video content analysis algorithms are usually built within a machine learning framework, where the system is first provided with a set of training instances that represent the type of analysis to be performed (e.g., soccer goal scoring), as well as negative training instances (normal play) and any of a large set of machine learning techniques is used to automatically determine which video analysis features are useful to predict this higher level analysis. However, the most impressive results in video content analysis are achieved when judicious constraints, rules or heuristics exploit the particular characteristics of the analysis to be performed, limiting the number of features that are considered and improving the accuracy of the analysis or classification result.

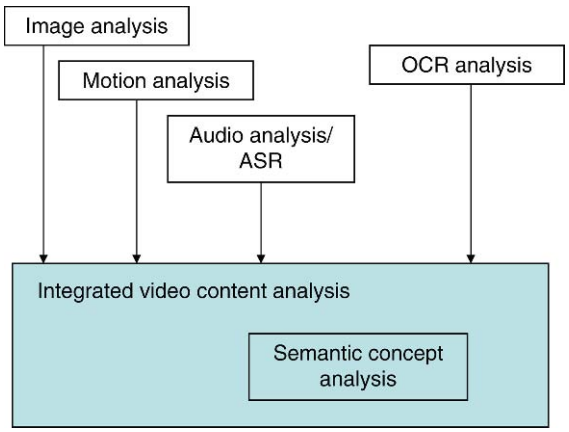
Much of the most interesting video content analysis is based on a fusion of analysis results at different levels of the different modalities (image, audio, motion, OCR). This type of content analysis has produced the most interesting results in sports video analysis, where the crowd and announcer noise indicates an interesting event on the field, while the image and motion analysis gives a clue to the type of event, and the OCR might indicate the players involved as well as the score.

Similarly, for broadcast video, a combination of image analysis for faces, audio analysis and camera analysis can determine which shots contain interviews with news subject, and OCR together with face recognition provides the identify of the person in the news. Video content analysis combining image similarity based on duplicate detection, identification of news studio shots and news anchor scenes can be used to provide the building blocks for news story segmentation. In order to summarize and browse through "scenes" or stories quickly, it is convenient to assign them to particular topic classes, which can again be done be integrating the results of different components of video content analysis.

In many ways, video content analysis fundamentally builds on image content analysis. Each frame in a video can be considered an image by itself, so all aspects of an image retrieval system can be applied. Text annotations may be available and can be searched as well. But video offers additional challenges and opportunities.

Video consists of additional types of information (e.g., audio sounds, human speech, motion) beyond what one might find in pure still image data. While a collection of a million images is considered quite large, just 10 h of video at 30 frames per second would provide a million frames (images). Of course, consecutive frames are often strongly related, providing high similarity and numerous slightly different versions of the same content in adjacent frames. However, the large amount of data, puts the burden on video content analysis to provide effective abstractions. The purpose of video content analysis is to provide extracted features and identification of structure that constitute building blocks for video retrieval, video similarity finding, summarization and navigation (Fig. 1).

Broadcast video consists of edited programs, whose structure can be exploited. Thus, news video consists of anchors, news stories, reporters, advertisements, etc. with each category providing context and constraint to what might be of use in further analysis. Videos can generally be partitioned temporally into keyframes, scenes, stories, and programs.



Video Content Analysis. Figure 1. Integrated video content analysis builds on Image, audio, motion and OCR analysis.

Partitioning Video

One of the earliest and most successful applications of video content analysis was the detection of shot boundaries. Shot boundary segmentation involves the identification of cuts, fades, and dissolves as typical editing effects added to video. Most techniques use sudden color, texture or edge changes and feature point tracks over various window sizes to determine shot breaks. In *movie or television video*, additional segmentation may aggregate several related shots into a scene or video paragraph, based on the similarity between sequences of shots.

In the case of *broadcast news video*, it is usually possible to segment a news story as a possible semantic unit, based on consistency in the text transcript, detection of news anchors, audio speaker analysis, and similarity in the news story footage. Special segmentation techniques involving black frames and rapid scene changes are used to identify commercial advertisements. Program or station logos, special transition effects, and recognition of distinctive backgrounds such as weather maps can also help determine news story boundaries.

Surveillance video has very different characteristics, since there are no editing artifacts, such as shot boundaries. Here, keyframe extraction is still desirable, but keyframes are usually selected from sections of video that contain motion, detected objects and people. In

surveillance video, the best representative keyframe captures an unusual event or a person/object when it is most clearly visible (largest) in the frame.

Audio Analysis

Audio analysis provides one component of video content analysis, supplementing visual analysis. From the audio track, it is possible to automatically generate a transcript to enable text-based retrieval from spoken language documents. Audio analysis can also identify changes in speakers and identify individual speakers.

Speech Recognition for Video Content Analysis and Retrieval The consensus from a number of published experiments in the area of automatic speech recognition for broadcast retrieval applications is that as long as speech recognition has a word error rate less than 35%, then information retrieval from the transcripts of spoken documents is only 3–10% worse than information retrieval on perfect text transcriptions of the same documents. The most accurate recognizers to date produce a word error rate under 20% for broadcast news, suitable for very accurate text-based retrieval.

Speaker and Audio Type Identification Another component of video content analysis is the use of audio features and speaker labeling. Many systems use audio analysis techniques to automatically extract additional metadata providing a description of the content of the audio channel. Audio processing can be used to detect music and other non-speech sounds, e.g., laughter and applause. Audio processing techniques can also be used to distinguish male versus female speakers, so that the respective audio regions can be passed to more accurately trained gender-specific speech recognizers. Existing audio analysis systems also identify well-known speakers for which a significant amount of training data is available, or group speakers across different speaker turns. Thus, for example, the techniques could identify the US President as well as most news anchors. Infrequently appearing speakers can be tracked when reappearing at different times within a video broadcast, although not labeled by name. This speaker change classification can be used to help break the video into coherent segments, and in general to characterize the audio channel of the data. Thus, video where one talker speaks for a long time can be labeled as a "speech," whereas an "interview" consists of two people switching back and forth, while a "forum" discussion includes many speakers. For broadcast data, it is possible to detect which speaker remains present throughout the entire audio data, thus identifying the anchor or narrator.

Content Analysis from Individual Image Frames

Most image content analysis for video usually happens on extracted keyframes. The full palate of image content analysis techniques can be applied, including low-level image content analysis, mid/high-level content analysis, image content modeling, image salient point extraction and representation, image segmentation and images similarity analysis, as described in the relevant sections.

Metadata Extraction Unique to Video

There are several types of content analyses that are unique to video.

Motion Feature Analysis One additional feature type not present in images is related to *motion*. There are several types of motion estimators that can be extracted, the motion blocks typically used in MPEG-style compression, the overall kinetic energy in the image and a fine-grained motion analysis based on optical flow. MPEG motion vectors can be directly extracted from the compressed result of an MPEG encoding process. These motion vectors can be aggregated and used for object and camera motion assessment. Kinetic energy measures the pixel variation within the shot. Optical flow algorithms typically warp one frame into another frame. They estimate the movements of high intensity interest points between two image frames as the motion vectors. While expensive to compute, optical flow analysis tends to provide the most refined estimates of object motion.

Recognizing object motion in video sequences is an important component in many applications. For example, in computer vision systems it enables the identification and tracking of the objects that make up a scene; while in video data compression it provides a means of reducing redundancy – knowing the motion of an object allows its position in successive frames to be predicted, removing the need to retransmit identical frame data and leading to a reduction in the bit rate required to transmit the video. Other applications include the generation of panoramic images, and recovery of 3-dimensional structure.

Recovering detailed motion from video sequences is a non-trivial effort. Factors such as the ambiguity

resulting from 3-D action recorded as a 2-D projection by the camera, poor contrast, and low spatial or temporal resolution, require sophisticated techniques to obtain reliable motion information. The range of approaches used includes affine models to estimate and track complex 2-D motions and techniques to deal with multiple component motion regions.

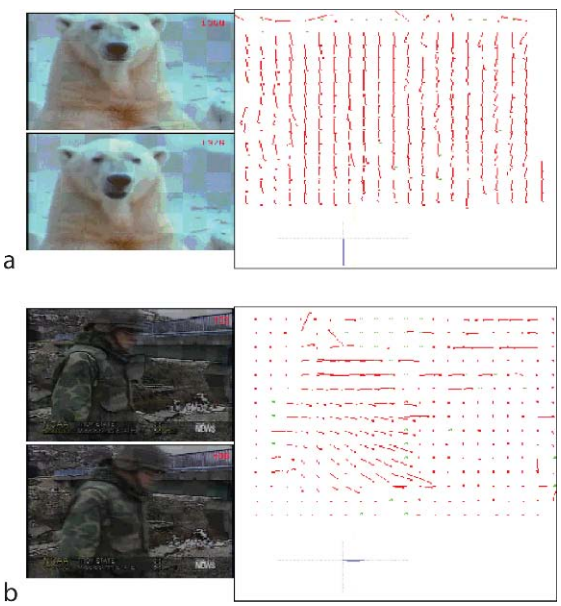
It has become popular in sports motion analysis to videotape subjects with markers placed on critical body parts, joints or equipment. The markers are used in each movement as references to the particular point of interest. Typically, the camera is carefully calibrated and distance markers placed in the field of view to allow accurate estimates of motion distances and velocities. This motion analysis allows customized systems to create models of motion and performance analysis relative to a specific sports activity. The motion may be translated into either 2-dimensional or 3-dimensional models and compared to reference motions.

Camera Motion Analysis One facet of video content analysis is concerned with analysis of camera motion, which includes pan (left/right), tilt (up/down), and zoom motions. More subtle analysis tries to distinguish dolly and boom motion, where the camera is physically moved from one place to another from the situation where the camera stays in place. Even further refinement of camera motion analysis tries to detect camera "shake," used for image stabilization. In principle, camera motion can also include rotation around a focal point; however this is a relatively infrequent event.

A wide variety of techniques exist for estimating the camera motion, most of which depend on the comparison between adjacent frames, based either on all pixels in the image or particularly salient pixels or regions. Since these pixels or regions only move slightly from frame to frame, their difference can be interpreted as a "track." From the overall patterns of tracks, models can be built to estimate if the motion pattern is more consistent with object motion, or camera shake, or camera pan/tilt/zoom, etc.

Most methods approach the problem of camera motion by deriving parameters proportional to the zoom, pan, rotate and tilt given a set of motion tracks. The methods usually determine an empirical threshold that classifies the derived parameters as representative for a particular camera motion. Optical flow analysis methods computed between consecutive images rely on tracking a large number of characteristic points across adjacent frames. Other methods use the MPEG motion vectors which encode one quantized motion vector per macro-block of pixels as an alternative to optical flow. Though these motion vectors are not directly equivalent to the true motion vectors of a particular pixel in the frame, there are typically enough motion vectors to robustly estimate camera motion parameters (Fig. 2).

Video OCR (VOCR) A somewhat different representation is derived by interpreting the text that is present in video images using optical character recognition (OCR). However, reading text present in the video stream requires a number of processing steps in addition to mere character recognition (see Fig. 3). First the text must be detected as present in a wide variety of scenes and backgrounds. Then it must be extracted from the image and finally converted into a binary (black and white) representation, since the commercially available OCR engines do not recognize colored text on variably colored background. The video OCR is further complicated because the text has very low resolution, frequently only about 10 pixels of height per character. Unlike text printed on white paper, the background of the image tends to be complex, with the character hue and brightness very near the background values. Among the solutions to these problems are



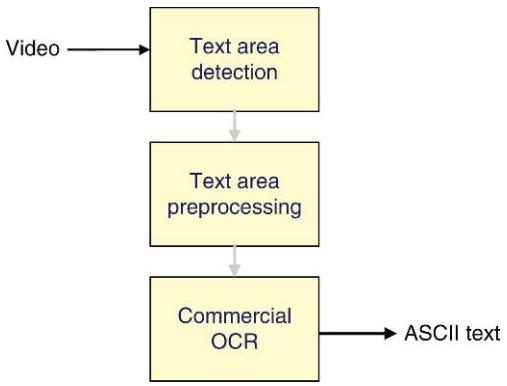
Video Content Analysis. Figure 2. Camera motion analysis for camera pan/tilt versus object motion using MPEG motion vectors.

interpolation filters, the integration of images across multiple frames and combinations of filters.

A text region detection is performed first, searching for horizontal rectangular structures of clustered sharp edges using variable orientation differential filtering techniques. Text boxes are identified based on their aspect ratio, absolute size and the fill factor of the bounding boxes.

Once a text area is detected, enhancement takes place. Multi-frame integration looks at the potential bounding boxes over several frames and finds the minimal (white) pixel values across that range. Potential text regions are sequentially filtered across consecutive frames, effectively increasing the resolution of each caption. Sub-pixel interpolation is performed to increase resolution without incurring jagged edges as artifacts (Fig. 4).

Adaptive thresholding on the gray-scale histogram is then used to create binarized black on white text before submitting it to an optical character recognition package. Systems will run OCR on multiple consecutive frames where text was detected, obtaining several nearly identical OCR results for a single occurrence of text on the screen that might last for a few seconds. These results can then be merged together for a single estimate. Once text is recognized, post-processing can correct some of the inevitable errors, e.g., using dictionary spelling correction.



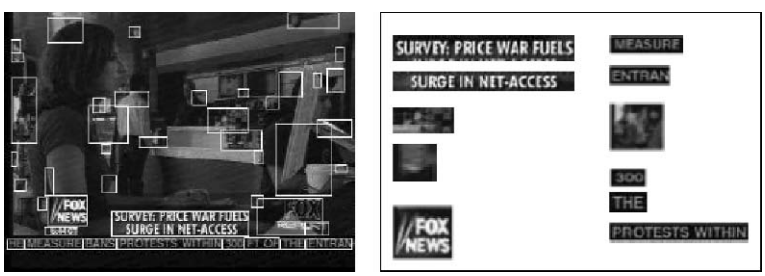
Video Content Analysis. Figure 3. Video OCR block diagram consisting of text area detection, text area preprocessing, and commercial OCR.

Combination of Audio Analysis, Image Analysis, and Motion Analysis for Semantic Content Analysis

Ultimately, video content analysis tries to achieve many of the same goals as mid-level and high level image analysis, except that the low level features available come from any number of modalities (audio, image, motion) and have already been performed at low, mid or high levels for this modality. The main task for video content analysis is then to fuse or combine these building blocks into a new characterization of the video.

Semantic Concepts as Mid or High-level Video Features. The effectiveness of low-level image analysis to represent image or video content is usually limited. The representations typically result in the notorious semantic gap between what users or applications expect and what content analysis systems can return, due to the systems' inability to capture the semantic meaning of the video content.

To provide a more semantic content analysis in video, an intermediate (mid to high-level) layer of hundreds of semantic concepts has been introduced in an effort to capture the semantic content of multimedia documents. The typical concepts includes a wide range of topics such as those related to people (face, anchor, etc.), objects (cars, buildings, bridges, graphics), location (outdoors, city, office setting), and genre (weddings, meetings, sports). Beyond the semantic concepts that are common to still image content analysis, semantic concepts for video can include temporally defined



Video Content Analysis. Figure 4. Candidate text regions as detected in a single frame of a video and some of the extracted individual text line.

events, such as an airplane landing, a vehicle stopping, etc. The successes of automatic semantic concept detection in recent years have demonstrated that a large number of high-level semantic concepts can be derived from the low-level multi-modal features with reasonable detection accuracy.

Detectors or classifiers for such semantic features are usually built off-line using large amounts of labeled training data combined with a number of machine learning approaches. Accuracy of the detectors depends critically on the number of training examples available as well as the difficulty of the semantic concept to be detected, e.g., a "left thumb" would be much more difficult to detect than a face. Face detection has been most effective for image and video retrieval, and significant added benefit can be gained from accurate face recognition, depending on the quality of the available images.

More recently, it has been proposed to combine these intermediate level semantic concepts into an ontology, which defines the relationships between concepts, their taxonomic structures and constraints. While researchers have shown some benefit to retrieval accuracy when using ontologies for limited domains, it is still unclear if a general-purpose ontology of visual concepts can be built and successfully applied. Similar efforts in the text analysis area have not yielded the expected results so far.

Key Applications

Search, browsing and summarization in video archives, real-time surveillance monitoring.

Cross-references

- ▶ Audio
- ▶ Audio Content Analysis
- ▶ Audio Segmentation
- ▶ Image Content
- ▶ Image Content Modeling
- ▶ Image Salient Points and Features
- ▶ Image Segmentation
- ▶ Image Similarity
- ▶ Low-Level Image Content Analysis (Color, Texture, Shape)
- ▶ Mid-to High-Level Image Content Analysis
- ▶ Video
- ▶ Video Content Modeling
- ▶ Video Content Structure
- ▶ Video Metadata
- ▶ Video Representation
- ▶ Video Scene and Event Detection
- ▶ Video Shot Detection

Recommended Reading

1. Chang S. and Sundaram H. Structural and semantic analysis of video. In Proc. IEEE Int. Conf. on Multimedia and Expo, 2000, pp. 687–690.
2. Hanjalic A. Content-Based Analysis of Digital Video. Kluwer Academic, Boston, 2004.
3. Jay K.C. Video Content Analysis Using Multimodal Information: for Movie Content Extraction, Indexing and Representation. Kluwer Academic, Norwell, MA, USA, 2003.
4. Marques O. and Furht B. Content-Based Image and Video Retrieval. Kluwer Academic, Norwell, MA, USA, 2002.
5. Multimedia Image and Video Processing, L. Guan, S.Y. Kung, J. Larsen (eds.) Multimedia Image and Video Processing, CRC, Boca Raton, FL, USA, 1999.
6. Naphade M.R. and Smith J.R. On the detection of semantic concepts at TRECVID. In Proc. 12th Annu. ACM Int. Conf. on Multimedia, pp. 660–667.
7. Smeulders A.W., Worring M., Santini S., Gupta A., and Jain R. Content-based image retrieval at the end of the early years. IEEE Trans. Pattern Anal. Mach. Intell., 22(12):1349–1380, 2000.
8. Smith M.A. and Kanade T. Multimodal Video Characterization and Summarization. Kluwer, 2005. Series in Video Computing, Vol. 9.
9. Snoek C., Worring M., and Hauptmann A.G. Learning rich semantics from news video archives by style analysis. ACM Trans. Multimedia Comp., Comm., and Appl., 2(2):91–108, 2006.
10. The Informedia Digital Video project http://www.informedia.cs.cmu.edu
11. Worring M. and Snoek C.G. Semantic indexing and retrieval of video. In Proc. 14th Annu. ACM Int. Conf. on Multimedia, pp. 13–13.

Video Content Description

- ▶ Video Metadata

Video Content Modeling

Lei Chen
Hong Kong University of Science and Technology, Hong Kong, China

Synonyms

Video data modeling

Definition

Video Content Modeling refers to representing the content of video data for search later on. Specifically, the content of video data includes the visual features, the temporal features, the contained objects, and the semantic concepts. With an effective modeling technique, people cannot only browse the video data, but also search the video with the specific features. Video content modeling is the basic for video data indexing and retrieval.

Historical Background

Video, as a popular type of multimedia, has been widely used by movie/TV industries and individuals. In the earlier 90s, people started search video data through annotated text information [20,21,23]. However, the low efficiency of manual annotation techniques prevents the text-based retrieval techniques applying to video data on a large scale. Thus, content-based video retrieval was proposed and studied. For the purpose of conducting effective and efficient search, significant works have been conducted on modeling video content. These techniques can be classified into three categories:

Segmentation-based models [24,18,12,8].
Video are recursively broken down into *scenes*, *shots* and *frames*. Key frames are extracted from shots and scenes to represent them, and visual features are extracted from key frames as indexes to key frames.

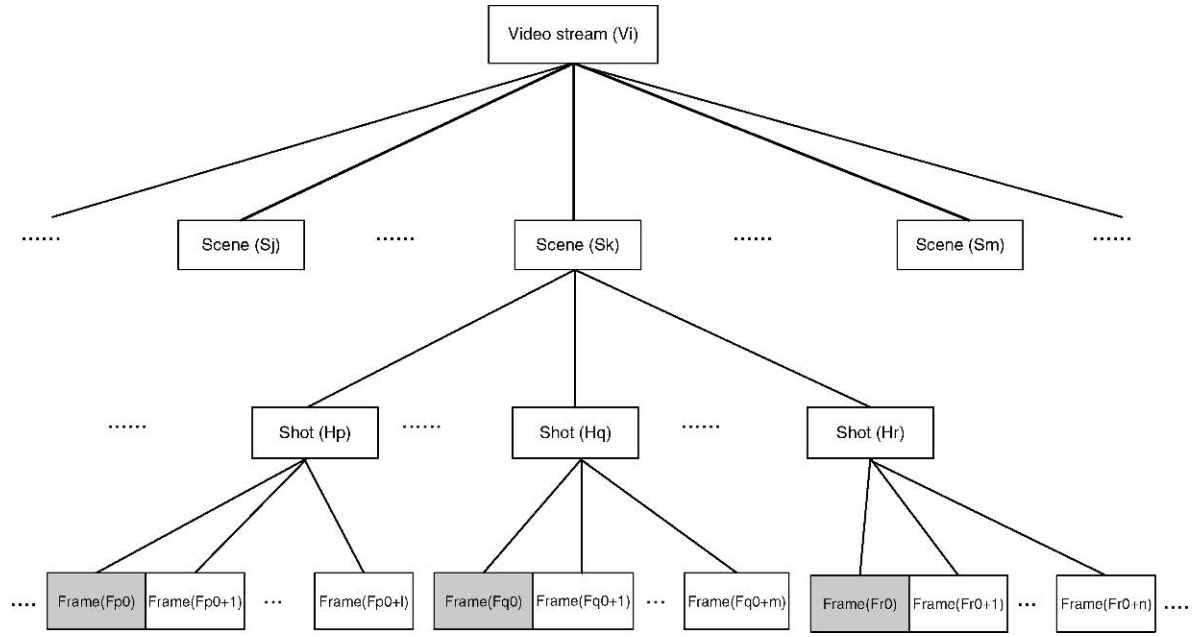
- Annotation-based models [20,23,22,14,3].
  In this model, a content description (annotation) layer is put on top of the video stream. Each annotation can be associated with a logical video sequence or physically segmented shots or scenes. An annotation describes the events or semantics of a video sequence.
- Salient object-based models [13,4,10,16,5,19]. Salient objects are extracted from video, and their audio-visual features and spatio-temporal relationships among them are described to express events or concepts.

Foundations

- Segmentation-based video data modeling:
  Modeling video data based on segmentation can be divided into three steps. First, video is segmented into *shots* (sometimes called *clips*); second, key frames are selected to represent the shots; finally, based on these shots, *scenes* or *story units* are constructed. During the last step, keyframes may also be selected to represent scenes and story units. Usually,visual features are extracted from the key frames as indexes to the video data. A video data model based on video segmentation possess a hierarchical structure (Fig. 1): a video stream contains several scenes or story units, each scene contains a set of shots and each shot contains asequence of video frames. The shaded frames in Fig. 1 are examples of key frames. The definition of shots, key frames and scenes are as follows [18]:
  - A shot is an unbroken sequence of frames recorded from a single camera.
  - A key frame is the frame selected from a shot which can represent the salient content of that shot.
  - A scene is a sequence of shots which conveys a concept or story.

The techniques for shot detection, key frame selection and scene construction, as briefly reviewed, follows:

- Video shot detection
  There are two basic types of video shot transitions: *abrupt* and *gradual* [24]. Abrupt transitions (cuts) occur when stopping and restarting cameras. Gradual transitions are introduced when two shots are artificially combined together by applying cinematic effects (fade-in, fade-out, dissolve, etc) [24]. Compared to the abrupt shot transitions, the gradual shot transitions are much more difficult to detect, because camera operations and big object movements may cause temporal variance similar to gradual shot transitions.
- Key frame selection
  Key frames provide a suitable abstraction and framework for video indexing, browsing and retrieval [1]. Users can get an overview of the video by only browsing the key frames. Furthermore, using key frames greatly reduces the amount of data required in constructing higher level video structure units (e.g., scenes, story units). Most of the key frame selection techniques are based on shot detection; in other words, key frames are mainly defined at the shot level. In order to select key frames, the video stream is first partitioned into shots, and different techniques are applied to extract key frames for each shot [25].



Video Content Modeling. Figure 1. A hierarchy structure of video data.

- Scene construction from shots

  Individual shots are fundamental representation units of a video. However, in a normal video stream, there are thousands of shots, and usually a single shot conveys very little semantics. Shots can be grouped into higher-level segments to form *scenes*, which can convey much more semantics than shots [18]. All the scene construction algorithms follow similar steps: (1) Shots are clustered together based on the similarity which is computed from some visual features (e.g., color, texture, edge, etc.), (2) Clusters that are temporally close to each other are grouped into scenes.

- Annotation-based video data modeling

  Annotation-based video modeling techniques associate annotations (keywords or free text) to video sequences. These annotations describe the semantics of the video sequences. Based on the structure of annotations, an annotation-based video data model can be classified into: a *single value annotation structure* and an *attribute-value pair structure*.

  - Single value annotation structure approach

    In this model, annotations are associated to logical [20,23] or physical [11] frame sequences directly.

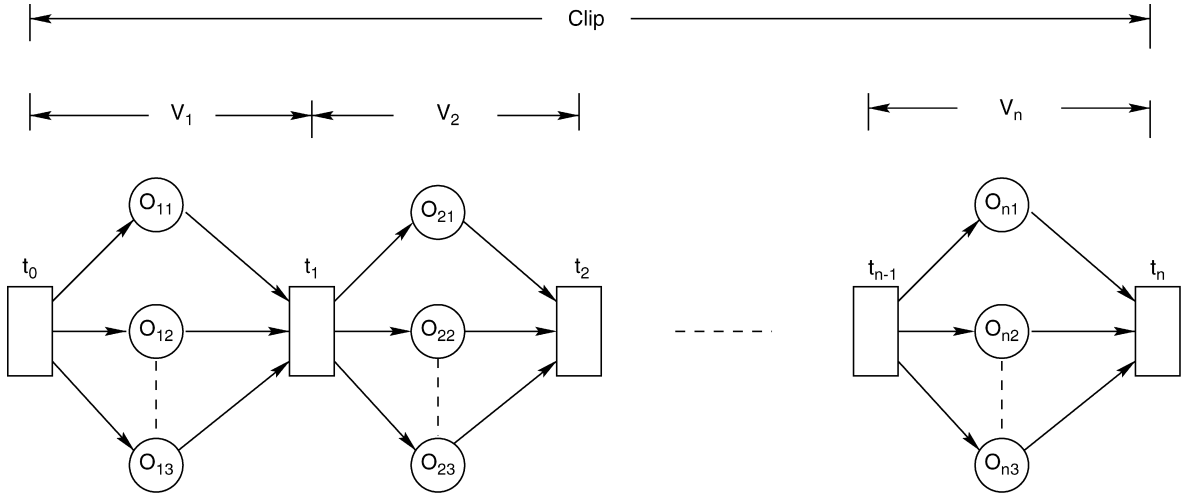
    - Annotations to logical frame sequences. Smith et al. [20] proposed a layered annotation representation model called the stratification model that segments the contextual information of the video. The stratification model divides the video sequence into a set of overlapping strata. A stratum consists of descriptive information such as title, keywords and the boundaries it represents. The strata may be contained in other strata and may encompass a multitude of other descriptions. The content information is derived by examining the union of all the contextual descriptions that are associated with it. Each stratum is logically independent and will make its contribution to the content when its descriptive information meets the user requirements.
    - Annotations to physical frame sequences

      Different from the stratification system which directly links the logical video segment to the raw video data at frame level, a generic video model [11] is developed which incorporates a segmentation-based video data model to associate text to a well defined structure. Hjelsvold's model supports sharing and reuse of the video data, temporal interval operations and structure abstraction. However, the arbitrary definition and

reuse of a frame sequence introduces complicated relationships between structure units and thematic annotations.

- Attribute-value pair structure approaches
  Instead of using the single value for annotations, attribute-value pairs are adopted to annotate the logical video segments. Attributes define the types, ranges and semantics of the values [17].

- Salient object-based video modeling
  Events that happen in a video such as dialogs, explosions, car chases are considered basic semantic units that the video intends to express. Through analysis of characteristics of video content, the event can be expressed by describing spatio-temporal relationships among the *video objects* that appear in that video. The models of this category can be further divided into *motion segmentation-based* and *spatio-temporal relationship-based*.
  - Motion segmentation-based approaches
    Motion based approaches try to model video data through investigating motion directions, trajectories and velocities of video objects. These motion parameters are extracted together with video objects from video data and stored together. Motion segmentation-based approaches can befurther divided into *graph-based* approaches and *motion vector-based* approaches.
    - Graph-based approaches
      Approaches in this category use a graph to describe moving objects, their trajectories and spatio-temporal relations with other moving video objects [9,13,15].
    - Motion vector-based approaches
      Instead of using a graph, motion vector-based approaches model moving video objects with a set of motion vectors [6].
  - Spatio-temporal relationship-based approaches
    Motion segmentation-based approaches only address motion parameters of video objects. However, the spatial and temporal relationships among video objects (not only moving video objects [15,13]) possess many more semantics. Several approaches have been proposed and they can be classified into two categories: *spatio-temporal logic-based* and *graph model-based*.
    - Spatio-temporal logic-based approaches
      Del Bimbo et al. [4] propose a Spatial-Temporal Logic (STL) language to describe the spatio-temporal relationships among video objects within image sequences.
    - Graph model-based approaches
      Different graphs are used in these approaches to model the spatio-temporal relationships among the video objects. From the point of view of dealing with semantic heterogeneity of video data, Day et al. [10] propose a Video Semantic Directed Graph (VSDG), which is used to construct users' heterogeneous views of the video data. VSDG is created to model spatio-temporal interactions between video objects. An example of a VSDG is shown in Fig. 2. A video clip is composed of n video segments, labelled $V_1, V_2, \ldots V_n$. Each video



Video Content Modeling. Figure 2. VSDG representation of a clip.

object (O_{11},O_{12},...O_{nn}) has attributes that describe the duration of its appearance in the video segments and its motion vector. Each rectangular node corresponds to an event in the video clip whenever a new physical object appears.

From the point of view of multimedia presentation, two other approaches have been proposed to model the spatio-temporal relationships among the video objects. HPNs [2] use Hierarchical Petri nets to capture the multi-level content of video data, which includes motion trajectories of moving objects and spatial temporal relationships among video objects. Chen et al. [7] model the salient objects, their content, and their spatio-temporal relationships in a hierarchy model and propose a video query language for users to search related contents.

Key Applications

Content Based Video Retrieval

Content-based video retrieval (CVIR) is the the problem of searching for videos that have similar content to the query video in a large database.

Video Databases

With a proper video content modeling technique, the video data can be stored in a database for efficient retrieval and analysis later on.

Cross-references

► Image Content Modeling

Recommended Reading

1. Aigrain P., Zhang H., and Petkovic D. Content-based representation and retrieval of visual media: A state-of-the-art review. Multimed. Tool. Appl., 3(3):179–202, 1996.
2. Al-Khatib W. and Ghafoor A. An Approach for Video Meta-Data Modeling and Query Processing. In Proc. 7th ACM Int. Conf. on Multimedia, 1999, pp. 215–224.
3. Bertini M., Bimbo A.D., and Torniai C. Automatic video annotation using ontologies extended with visual information. In Proc. 13th ACM Int. Conf. on Multimedia, 2005, pp. 395–398.
4. Bimbo A.D., Vicario E., and Zingoni D. Symbolic Description and Visual Querying of Image Sequences Using Spatio-Temporal Logic. IEEE Trans. Knowl. and Data Eng., 7(4):609–622, 1995.
5. Browne P. and Smeaton A.F. Video information retrieval using objects and ostensive relevance feedback. In Proc. 2004 ACM Symp. on Applied Computing, 2004, pp. 1084–1090.
6. Chang S.F., Chen W., Meng H.J., Urama H., and Zhong D. A fully automated content-based video search engine supporting spatiotemporal queries. IEEE Transaction on Circuits and Systems for Video Technology, 8(5):602–615, 1998.
7. Chen L., Oria V., and Özsu M.T. Modeling Video Data for Content Based Queries: Extending the DISIMA Image Data Model. In Proc. 9th Int. Conf. on Multimedia Modeling, 2003, pp. 169–189.
8. Cooper M. Video segmentation combining similarity analysis and classification. In Proc. 12th ACM Int. Conf. on Multimedia, 2004, pp. 252–255.
9. Courtney J.D. Automatic video indexing via ojbect motion analysis. Pattern Recognition, 30(4):607–625, 1999.
10. Day Y.F., S.D., Iino M., Khokhar A., and Ghafoor A. Object-oriented conceptual modeling of video data. In Proc. 11th Int. Conf. on Data Engineering, 1995, pp. 401–408.
11. Hjelsvold R. and Midtstraum R. Modelling and Querying Video Data. In Proc. 20th Int. Conf. on Very Large Data Bases, 1994, pp. 686–694.
12. Lefèvre S., Holler J., and Vincent N. A review of real-time segmentation of uncompressed video sequences for content-based search and retrieval. Real-Time Imaging, 9(1):73–98, 2003.
13. Li J., Özsu M.T., and Szafron D. Modeling of moving objects in a video databas. In Proc. Int. Conf. on Multimedia Computing and Systems, 1997, pp. 336–343.
14. Martinez M. and Moran F. Authoring 744: Writing Descriptions to Create Content. IEEE MultiMedia, 10(4):94–99, 2003.
15. Nabil M., Ngu A.H.H., and Shepherd J. Modeling Moving Objects in Multimedia Database. In Proc. 8th Int. Conf. Database and Expert Syst. Appl., 1997, pp. 67–76.
16. Naphade M.R. and Huang T.S. Extracting semantics from audio-visual content: the final frontier in multimedia retrieva. IEEE Transactions on Neural Networks, 13(4):793–810, 2002.
17. Oomoto E. and Tanaka K. Ovid: Design and implementation of a video-object database system. IEEE Trans. Knowl. and Data Eng., 4(5):629–643, 1993.
18. Rui Y., Huang T.S., and Mehrotra S. Exploring Video Structure Beyond the Shots. In Proc. Int. Conf. on Multimedia Computing and Systems, 1992, pp. 237–240.
19. Shibata T., Kato N., and Kurohashi S. Automatic object model acquisition and object recognition by integrating linguistic and visual information. In Proc. 15th ACM Int. Conf. on Multimedia, 2007, pp. 383–392.
20. Smith T.G.A. and Davenport G. The stratification System: A Design Environment for Random Access Video. In Proc. Int. Workshop on Networking and Operating System Support for Digitial Audio and Video, 1992, pp. 250–261.
21. Smoliar S. and Zhang H. Content-based video indexing retrieval. IEEE Multimedia, 1(2):62–72, 1994.
22. Vendrig J. and Worring M. Interactive Adaptive Movie Annotation. IEEE MultiMedia, 10(3):30–37, 2003.
23. Weiss R., Duda A., and Gifford D.K. Composition and search with a video algebra. IEEE Multimedia, 1(2):12–25, 1994.
24. Zhang H., Kankanhalli A., and Smoliar S. Automatic partitioning of full-motion video. Multimedia Systems, 1:10–28, 1993.
25. Zhang H.J., Low C.Y., Smoliar S.W., and Wu J.H. Video parsing, retrieval and browsing: An integrated and content based solution. In Proc. 3rd ACM Int. Conf. on Multimedia, 1995, pp. 15–24.

Video Content Processing

► Video Content Analysis

Video Content Structure

Xian-Sheng Hua, Meng Wang
Microsoft Research Asia, Beijing, China

Synonyms

Video structuring; Video structure analysis

Definition

Mining video content structure is an elementary step of video content analysis. Direct access to a video without indexing is usually not an easy task, due to its length and unstructured format. On the other hand, analogous to text documents that can be decomposed into chapters, paragraphs, sentences and words, videos can be segmented into units like scenes, shots, and keyframes. The analysis of video content structure can be viewed as the process of hierarchically decomposing videos into units and building their relationships. Through such a process, a table-of-content can be constructed for each video, which facilitates the access and manipulations of the video data. For example, the keyframes extracted from the video can be used as its entries for indexing and browsing.

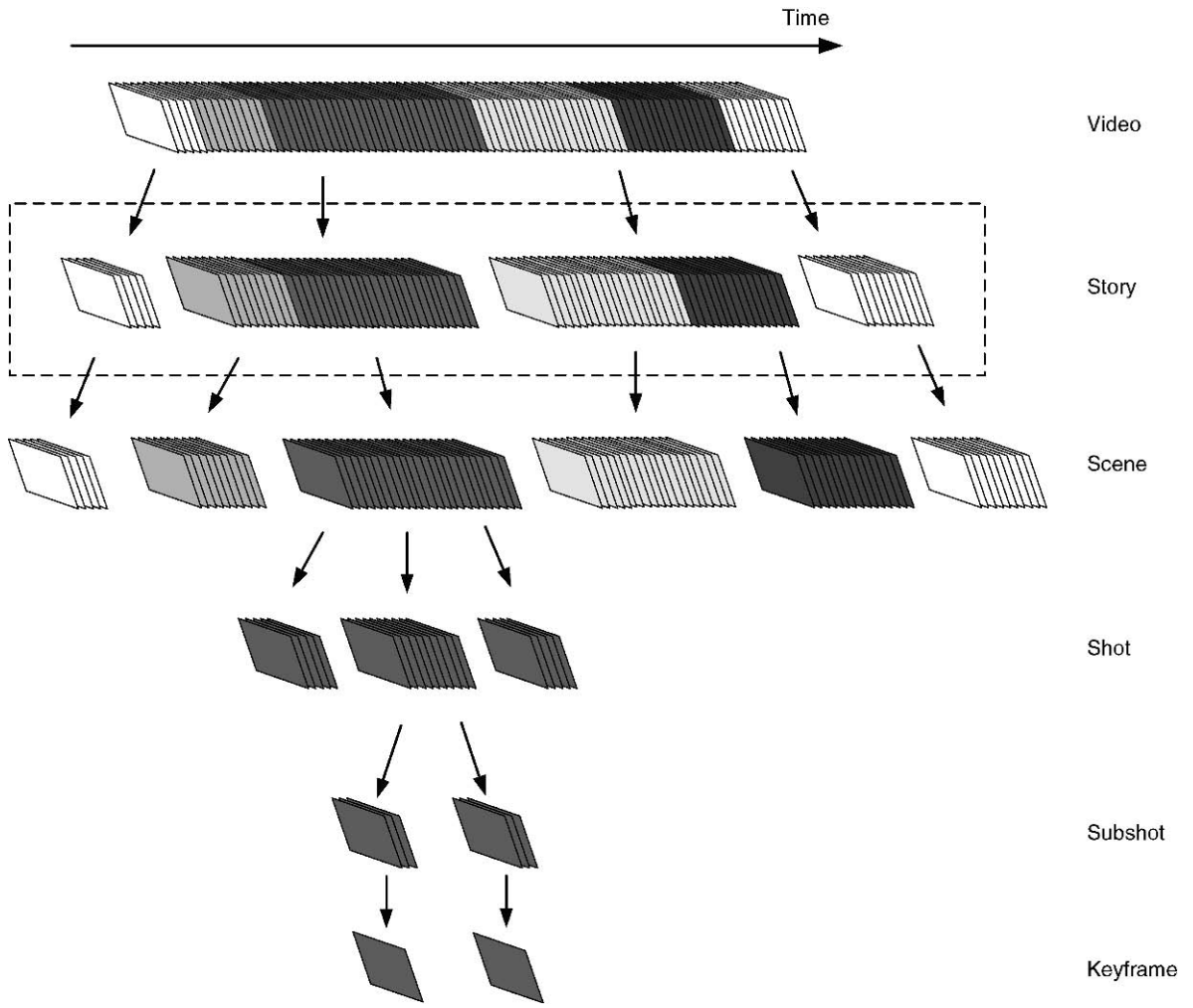
Historical Background

More and more *video* data has become available to ordinary users due to the advances in storage devices, networks and compression techniques. However, the efficient access to an unstructured video is a challenging task due to the huge number of frames. Video structuring is proposed to tackle this difficulty. In [14], Zhang et al. first proposed the scheme that partitions videos into shots, and then selects the first frame of each shot as the keyframe for indexing. From 2001, the National Institute of Standard and Technology (NIST) has established video shot detection as a evaluation task in the TREC Video Retrieval Evaluation (TRECVID) benchmark (http://www.itl.nist.gov/iaui/894.02/projects/trecvid/). In [10], Rui et al. proposed to group shots into scenes, such that the video content can be structured at a higher level. For several video genres, such as news and movies, "story" is also widely applied as the unit for content organization. Kim et al. [6] proposed a method to further segment shots into subshots in order to facilitate more detailed implementations. For each shot or subshot, one or more keyframes can be extracted to represent its content. Thus, as illustrated in Fig. 1, generally a video can be structured in a hierarchical form as "video → scenes → shots → subshots → keyframes," and it can also be segmented into stories if it belongs to certain genres, such as news video. The definitions of these terminologies are as follows:

Shot: a shot is an uninterrupted clip recorded by a single camera. It is a physical entity and often forms the building block of video content.
Scene: a scene is defined as a collection of semantically related and temporally adjacent shots, depicting and conveying a high-level concept or story. A scene usually consists of a series of consecutive shots that are recorded in the same location.
Story: a story is referred to as a clip that captures a continuous action or a series of events, and it may be composed of several scenes and shots. Note that the story lines are usually only clear for rigidly structured videos. Currently, most story identification methods are developed for news videos. Thus, here only "news story" is considered. A definition of a news story in TRECVID is "a segment of a news broadcast with a coherent news focus which contains at least two independent, declarative clauses."
Subshot: subshot is a segment within a shot that corresponds to a unique camera motion. A shot can be divided into one or more consecutive subshots according to the movement of the camera.
Keyframe: a keyframe is the frame which best represents the content of a shot or a subshot. According to the content complexity, one or more keyframes can be extracted for each shot or subshot. Keyframes can be used as the entries of the video data for manipulations, such as indexing and browsing.

Foundations

As previously introduced, the analysis of video content structure may involve five techniques, namely, shot detection, scene grouping, story identification, subshot segmentation, and keyframe extraction.



Video Content Structure. Figure 1. Hierarchical decomposition and representation of video content.

Shot Detection

Shot detection is the process of identifying the boundaries between two consecutive shots, such that the frame sequence can be grouped into a set of shots. According to the transition style of the consecutive shots, the shot boundaries can be mainly categorized into two types, i.e., cut and gradual transitions. Cut indicates that the change between the two shots is abrupt, whereas the gradual transition means that there is a gradual special effect in the transition of the two shots. Many different shot detection methods have been proposed (including the detection and categorization of gradual transitions), and a most straightforward approach to shot detection is to measure the change between every two consecutive frames and a shot boundary can be declared if there is a significant change. Yuan et al. have provided a comprehensive survey on shot detection in [13], and more details about this technique can be found in the entry *Video Shot Detection*.

Scene Grouping

Scene grouping is usually implemented based upon the results of shot detection. From a global point of view, scene grouping can be viewed as a shot *clustering* task. With an appropriate pairwise similar definition, many different clustering algorithms can be applied to accomplish this task. Intuitively, two criteria should be considered in a scene grouping algorithm, namely, content similarity and temporal continuity. Content similarity means that the shots within the same scene should have similar content, whereas the temporal continuity indicates that these shots should be close to each other

temporally. The content similarity is usually defined based on a low-level feature space, and many different features can be applied, including visual, audio and text features (the text features can be extracted by several existing techniques such as "Automatic Speech Recognition" and "Optical Character Recognition").

Gu et al. [3] have categorized the existing scene grouping methods into three approaches: merging-based, splitting-based, and model-based. The merging-based approach groups shots in a bottom-up way. In [10], the scene grouping is implemented via a two-step process: (i) assign the shots into groups according to their visual similarities and temporal continuities; and (ii) merge similar groups into a unified scene. Rasheed et al. proposed another two-step scene grouping process in [9]: (i) cluster shots according to Backward Shot Coherence (BSC); and (ii) merge the clusters into scenes based on an analysis of the shot length and the motion content in the potential scenes.

As opposed to the merging-based approach, splitting-based scene grouping methods are implemented in a top-down style, i.e., a video is split into a set of coherent scenes in turn. In [8,12], two different graph-based splitting methods are proposed. In these two methods, a video is represented by a graph, in which the vertices are denoted by the shots and the edges are determined by the similarities of the shots and their temporal localities. A graph is then partitioned into several subgraphs and each subgraph can be regarded as a scene. In [12], the graph is named as a Scene Transition Graph (STG), and it is partitioned into several subgraphs with the complete-link method, whereas in [8] the graph is named as a Shot Similarity Graph (SSG) and it is partitioned by the normalized cuts method.

Different from the above two approaches, model-based methods group shots into scenes with statistical models. Tan et al. [11] implemented scene grouping using the Gaussian Mixture Model (GMM), with each Gaussian component indicating the distribution of a scene, and the number of scenes can be determined by the Bayesian Information Criterion (BIC) method. Recently, Gu et al. [3] proposed an energy minimization scheme, in which the constraints of time and content of scene grouping are indicated by energy items. It is able to take both the global and local constraints into consideration, and the number of scenes can be established by the Minimum Description Length (MDL) principle.

Story Identification

Story identification needs more semantic understanding of video content (Fig. 2), and it is usually only applied for certain rigidly structured video genres such as news video. In TRECVID 2003 and 2004, news story identification was established as an evaluation task. The existing story identification methods can be mainly classified into two categories, i.e., rule-based and learning-based. The rule-based story identification methods are usually based on certain domain knowledge. For example, it has been observed that many news stories begin with an anchorperson shot and end with the start of another anchorperson shot. In [15], the pattern of news stories has been analyzed, and the task of story identification is accomplished via detecting the shots of certain types, such as anchorperson shot. However, such an approach highly depends on the adopted knowledge, and can hardly handle diverse video sources with different features and production rules. The learning-based approach is able to tackle this difficulty. In typical learning-based story identification methods, a set of story boundary candidates is first established (such as the shot boundaries and audio pauses), and then each candidate is classified as "story boundary" or not according to the model learned from a training set [1]. More details about these story identification methods can be found in [1,15] and references therein.

Video Content Structure. Figure 2. A typical news story in a video. The keyframes of the video have been illustrated, and the story boundaries can be identified according to the anchorperson analysis.

Subshot Segmentation

Subshot is a sub segment within a shot. Generally, a subshot is defined to contain a unique camera motion. Thus, subshot segmentation can be accomplished through camera motion detection. For example, consider a shot in which the camera moves as follows: zoomed out, then panned from left to right and zoomed in to a specific object, and then stopped. This shot then comprises three subshots, including one zoom out, one pan to right, and one zoom in. The camera motion between two adjacent frames can be estimated based on a two-dimensional affine model, in which the motion vector (v_x, v_y) at pixel (x, y) can be expressed as

$$\begin{pmatrix} v_x \\ v_y \end{pmatrix} = \begin{pmatrix} a_1 \\ a_4 \end{pmatrix} + \begin{pmatrix} a_2 & a_3 \\ a_5 & a_6 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \quad (1)$$

where $a_i (i = 1,2,...,6)$ denote the motion parameters. The motion parameters can be represented by a more meaningful set of terms as follows

$$\begin{cases} pan = a_1 \\ tilt = a_4 \\ zoom = \frac{a_2 + a_6}{2} \\ rot = \frac{a_5 - a_3}{2} \\ hyp = \frac{|a_2 - a_6| + |a_3 + a_5|}{2} \end{cases} \quad (2)$$

where *pan* corresponds to the pan movement of camera, *tilt* corresponds to tilt and boom, *zoom* corresponds to dolly and the change of focus, *rot* corresponds to roll, and *hyp* indicates that object motion is predominant. For more details about video motion analysis, please refer to [6].

Based on the analysis of camera motion, the subshot segmentation can be implemented and each subshot will be categorized into one of the following six classes: *pan*, *tilt*, *zoom*, *rot*, *objectmotion*, and *static*. For an individual frame, its motion category can be determined by thresholding the related terms in Eq. (2). However, it is observed that generally a camera movement will be maintained for a period of, say, at least a half second [6]. Thus, a typical subshot segmentation process consists of three steps, i.e., frame-level motion detection, segment-level motion detection, and post-processing. More details can be found in [6]. Figure 3 illustrates an example, in which the shot can be segmented into seven subshots.

Keyframe Extraction

Keyframes are the frames in the video sequence that can best preserve the content of shots or subshots. They can be used as the entries of the videos for access. The most widely-applied methods for this task can be categorized into two approaches, i.e., analysis-based and clustering-based. The analysis-based methods extract keyframes by analyzing video content, such as the quality and the attractiveness of frames. For example, in [7], Ma et al. adopted an attention model, and the frames that attract the most user attention are extracted as keyframes. In the clustering-based approach, a clustering process is carried out, and then the cluster centroids can be established as keyframes. In [4], Hanjalic et al. adopted a partitioning-based clustering method, and the number of clusters (i.e., the number of keyframes) can be determined by a cluster-validity approach.

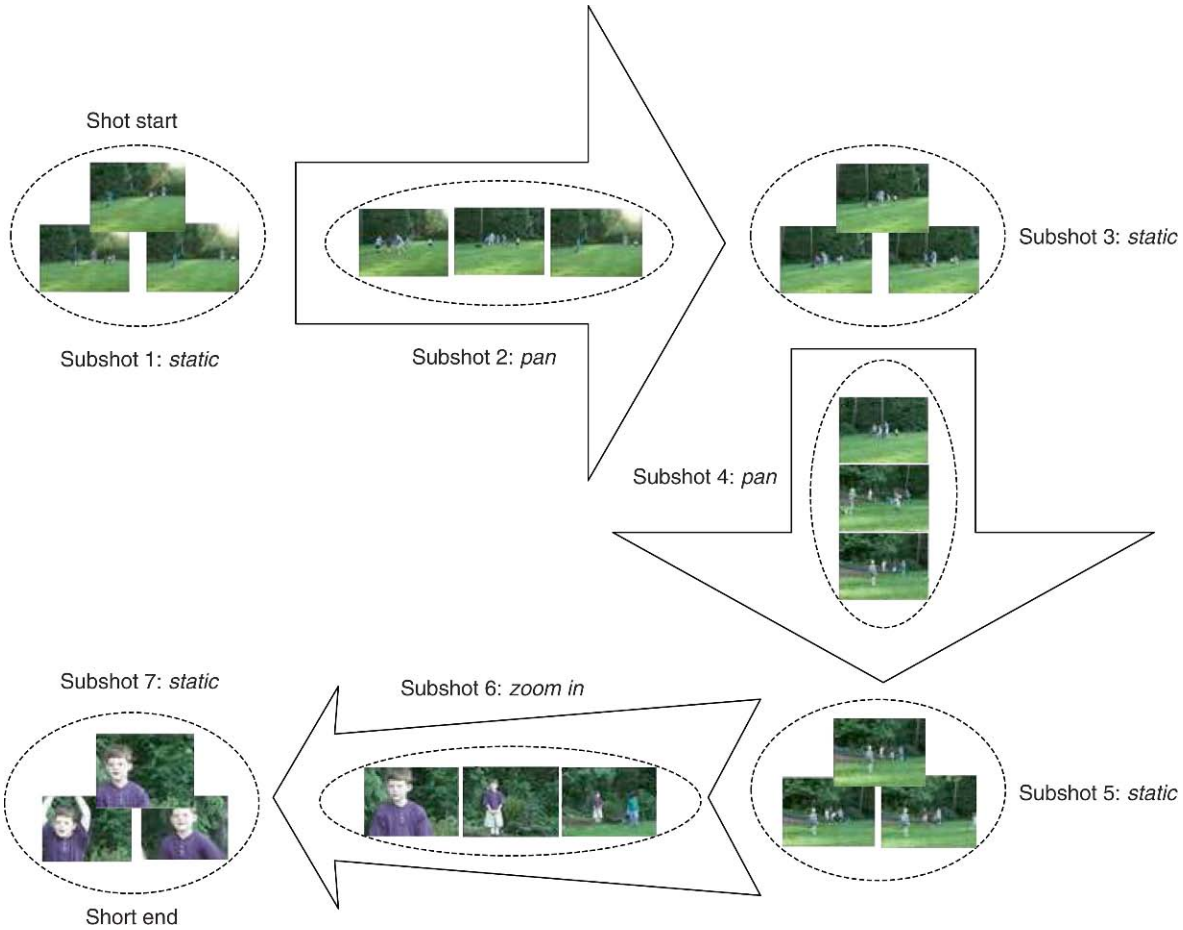
A recent development in keyframe extraction is to formulate it as a learning task [5]. It is observed that the representativeness of a video frame involves several elements, such as its image quality, user attention, and visual details. Thus, the frame representativeness can be modeled through a training set that regards these elements as features. This learning process is able to simulate a human's perception on keyframe extraction. The method in [5] builds a model based on four elements extracted from each frame, including *frame quality*, *visual details*, *content dominance* and *attention measurement*, and encouraging results have been reported in both subjective and objective evaluations.

Key Applications

As previously mentioned, analyzing video content structure is the first step of *video content analysis*, and it is thus a prerequisite for many video applications, such as *video abstraction*, *video summarization* and *content-based video retrieval*.

Future Directions

The existing video structuring methods are mainly carried out based on low-level features. However, from the above introduction it can be found that several structuring techniques involve the understanding of video content. For example, for scene grouping, it is better to group shots with coherent semantic concepts rather than those with close low-level features. Thus, it is believed that better structuring performance can be achieved by leveraging semantic features in the existing algorithms. This can be named as "semantic-based video structuring"



Video Content Structure. Figure 3. An illustrative example of subshot segmentation.

approach. For details about the application of semantic features (or semantic feature space), please refer to [2]. It is worth noting that structuring is currently always regarded as an independent and preliminary step of video semantic analysis. To use semantic features for structuring, the semantic analysis and structuring may have to be integrated or unified.

Experimental Results

Generally, for each presented method, there is an accompanying empirical evaluation in the corresponding reference. For shot detection and story identification, the results achieved in TRECVID can be found in the corresponding notebook papers which can be downloaded from the TRECVID website (http://www-nlpir.nist.gov/projects/tvpubs/tv.pubs.org.html).

Cross-references

- ▶ Clustering
- ▶ Content-Based Video Retrieval
- ▶ Video
- ▶ Video Abstraction
- ▶ Video Content Analysis
- ▶ Video Shot Detection
- ▶ Video Summarization

Recommended Reading

1. Chua T.-S., Chang S.-F., Chaisorn L., and Hsu W. Story boundary detection in large broadcast news video archives - techniques, experiences and trends. In Proc. 12th ACM Int. Conf. on Multimedia, 2004.
2. Ebadollahi S., Xie L., Chang S.-F., and Smith J.R. Visual event detection using multi-dimensional concept dynamics. In Proc. IEEE Int. Conf. on Multimedia and Expo, 2006.

3. Gu Z., Mei T., Hua X.-S., Wu X., and Li S. EMS: Energy minimization based video scene segmentation. In Proc. IEEE Int. Conf. on Multimedia and Expo, 2007.
4. Hanjalic A. and Zhang H.-J. An integrated scheme for automated video abstraction based on unsupervised cluster-validaty analysis. IEEE Trans. Circ. Syst. Video Tech., 9(8):1280–1289, 1999.
5. Kang H.-W. and Hua X.-S. To learn representativeness of video frames. In Proc. 13th ACM Int. Conf. on Multimedia, 2005.
6. Kim J.-G., Chang H.S., Kim J., and Kim H.M. Efficient camera motion characterization for MPEG video indexing. In Proc. IEEE Int. Conf. on Multimedia and Expo, 2000.
7. Ma Y.F., Lu L., Zhang H.-J., and Li M. A user attention model for video summarization. In Proc. 10th ACM Int. Conf. on Multimedia, 2002.
8. Rasheed Z. and Shah M. Detection and representation of scenes in videos. IEEE Trans. Multimed., 7(6):1097–1105, 2005.
9. Rasheed Z. and Shah M. Scene detection in holleywood movies and tv shows. In Proc. Int. Conf. on Computer Vision and Pattern Recognition, 2005.
10. Rui Y., Huang T.S., and Mehrotra S. Constructing table-of-content for video. Multimed. Syst., 7:359–368, 1999.
11. Tang Y.-P. and Lu H. Model-based clustering and analysis of video scenes. In Proc. Int. Conf. Image Processing, 2002.
12. Yeung M., Yeo B., and Liu B. Segmentation of videos by clustering and graph analysis. Comput. Vis. Image Understand., 71(1):94–109, 1998.
13. Yuan J., Wang H., Xiao L., Zheng W., Li J., Lin F., and Zhang B. A formal study of shot boundary detection. IEEE Trans. Circ. Syst. Video Tech., 17:168–186, 2007.
14. Zhang H.-J., Kankanhalli A., and Smoliar S.W. Automatic paritioning of full-motion video. Multimed. Syst., 1: 10–28, 1993.
15. Zhang H.-J., Tan S.Y., and Smoliar S.W. Automatic parsing and indexing of news video. Multimed. Syst., 2(6):256–265, 1995.

Video Data Modeling

► Video Content Modeling

Video Format

► Video Representation

Video Indexing

► Video Sequence Indexing
► Visual Content Analysis

Video Metadata

Frank Nack
University of Amsterdam, Amsterdam, The Netherlands

Synonyms

Video annotation; Video content description

Definition

Digital video is recorded in two different image capture formats: interlaced and progressive scan. Interlaced video establishes an image by recording alternating sets of lines, where one set of odd or even lines is referred to as a "field," and a consecutive pairing of two fields of opposite parity is called a frame. In a progressive digital video each frame is recorded in a distinct manner, with both fields being identical. Both operate at the same number of frames per second, which is in NTSC roughly 29 images and in Pal around 25 images per second. As of 2007, the highest resolution demonstrated for digital video generation is 33 megapixels (7,680 × 4,320) at 60 frames per second ("UHDV").

Metadata is data about data of any sort in any media, describing an individual datum, content item, or a collection of data including multiple content items. In that way, metadata facilitates the understanding, characterization, use and management of data.

Video metadata is structured, encoded data that describes content and representation characteristics of information-bearing video entities to facilitate the automatic or semiautomatic identification, discovery, assessment, and management of the described entities, as well as their generation, manipulation, and distribution.

Historical Background

In the late 1970s to the early 1980s of the twentieth century the first digital video production equipment, such as time base correctors (TBC) and digital video effects (DVE) units, were developed. This type of

equipment was, however, not part of the general production flow, as the digitized and processed video still needed to be converted back to standard analog video.

Digital video was first introduced commercially in 1986 with the Sony D-1 format, which recorded an uncompressed standard definition component video signal in digital form. Due to its costs, the D-1 was primarily used by large television networks. Over the years it was replaced by cheaper systems, such as Sony's Digital Betacam, which were still expensive so that they were mainly used as a field recording format by professional television producers.

Consumer digital video first appeared in 1991 in the form of QuickTime, Apple's framework for time-based and streaming data. Shortly after, Microsoft's AVI format followed and then MPEG-1, MPEG-2 and MPEG-4 (the basis of MPEG-4 was QuickTime) formats gained popularity. The introduction of the DV tape in 1996 gave digital video another acceptance push as it allowed recording directly to digital data and thus simplified the editing process, allowing non-linear editing systems to be deployed wholly on desktop computers (e.g., FAST 601, Softimage DS, Appel's Final Cut line).

The work on video codec technology from the early 1990s allowed a great deal of research directed on computer environments that seek to interpret, manipulate or generate digital video either in a manual, semi-automatic, or automatic way [1–6]. In all these research projects the description of video content in form of metadata played a central role. The steady infiltration of those technological advances in everyday production finally allowed the general public to really include video into their everyday communication. Examples of such infiltrated technologies are simple non-linear video editing systems, environments for new media authoring (e.g., Director/Shockwave, Flash), and web authoring environments (e.g., Dreamweaver, Frontpage, and SMIL).

The rapid growth of professionally created and exploited video databases, as well as the slow but steady growth of user-generated video material on the web (the growth was much slower than the one of images), forced research to address similar problems as those already explored for digital image databases, namely how to efficiently search and then exploit video databases. Over the years a number of metadata standards have been developed, which address various aspects of video data, such as the Dublin Core Metadata Initiative (http://www.dublincore.org/), the Society for Motion Pictures and Television Engineering (SMPTE) (http://www.smpte.org/home/), the Moving Picture Expert Group (MPEG) (http://www.chiariglione.org/mpeg/), the TV-Anytime Consortium (http://www.tv-anytime.org/), and the International Press Telecommunications Council(IPTC) (http://www.iptc.org/pages/index.php). The common definition language between all these languages has been in one way or another the Extensible Markup Language (XML) [7], defined by W3C.

The most relevant initiative for addressing segmentation, indexing and content-based retrieval of digital video based on low-level features, usually automatically extracted from the content, is the TrecVid series (http://trecvid.nist.gov).

The by far largest contribution to digital video was provided by the three ISO standards MPEG-4 [8], MPEG-7 [9] and MPEG-21 [10]. With MPEG-4 the group entered the realm of media content, arisen due to the growing need for content manipulation and interaction. MPEG-4 expanded MPEG-1 to support video "objects," 3D content, low bitrate encoding and support for Digital Rights Management. Yet, MPEG-4 lacked the means to identify video objects on a semantic level – a necessary element for efficient search in and the manipulation of video. Addressing this problem, the MPEG-7 standard was started in the late 1990s. One core part of MPEG-7 (Part 3: Video – and also larger sections of Part 5: The Multimedia Description Schemes) is devoted to video annotation only. The beginning of the twenty-first century established an even faster exchange of multimedia data via the web, as higher bandwidth as well as access to high quality data became a commodity. Media businesses, such as the film industry, feared, due to peer-to-peer technology for their markets and requested strict digital rights management enforcement. MPEG reacted with MPEG-21, defined as the multimedia framework.

Since 2005 the web saw an exploitation of video content. The main reasons were:

- Easy access to digital video cameras
- The provision of Adobes Flash player, which can be easily integrated into web pages
- Easy ways of sharing enabling technology, of which YouTube (http://www.YouTube.com), among others, is the most well-known example

The relevance of YouTube is not so much grounded in its technology advances (YouTube plays back videos limited in size (320 × 240 pixel) and quality (a bitrate of around 314 kbit/s with a frame rate depending on the uploaded video), but rather on the distribution of content. YouTube lets their clientèle act as they please, without enforcing editorial decisions or odds and ends such as copyright. As many Web2.0 applications YouTube applies folksonomy tagging (also known as collaborative tagging, social classification, social indexing, and social tagging) a method of collaboratively creating and managing tags to annotate and categorize content. In folksonomy tagging metadata is not only generated by experts but mainly by creators and consumers of the content, where a tag is a keyword or term associated with or assigned to a piece of information (a picture, a map, etc.), which enables keyword-based classification and search. The advantage of tagging is its ease of use – creating a vocabulary based on freely chosen keywords instead of a controlled set of terms and structures. This approach, though highly popular, carries serious problems. Typically there is no information about the semantics of a tag, no matter if it is a single tag or a bag of tags. Additionally, different people may use drastically different terms to describe the same concept. This lack of semantic distinction can lead to inappropriate connections.

The success of YouTube, which consumed in 2007 as much bandwidth as the entire Internet in 2000, and similar sites let the W3C start a new video on the web initiative in 2008, which will address the access of video on the web (http://www.w3.org/2008/WebVideo/Activity.html).

Foundations

The description of video content is to some extent very similar to the one of images, simply because a video is the representation of single images over time. The following text, therefore, only addresses those issues of video content representation that are not already addressed in the "Image Metadata" entry of this encyclopedia. As applications like YouTube reestablished the question of rights management on a large scale, this section also addresses this problem in some detail.

The two essential aspects of video that enhance its "meaning making" over that of an image is the concept of a shot and the sequence of shots, or time and structure of time.

Shot Description

A shot is a single piece of film, however long or short, without cuts, exposed continuously. The significant additional element here is time, which provides the basis for the understanding of action, distance and the relationship among characters, based on the relationship between frames within a shot and their rhythmical variations. The compositional use of *focus*, for example, through which the foreground, middle ground or background are emphasized, is one way to guide the perception of a shot. If all planes are represented in focus, they are attributed with the same level of importance, whereas emphasis can be achieved by use of focus for a part of a frame. Of even stronger impact than focus is *camera movement* around the imaginary vertical axis (pan), the horizontal axis (tilt), and the longitudinal axis (distance from lens to the subject). The *tilt*, for example, presents the eye-level from which a scene is perceived and thus can affect the importance ascribed to an object (for example, high-angle shots may diminish the perceived importance of an object). The *tempo* of a shot can also provide information. The intense feeling of fast movement may excite, while calm movement expressed, for example, through the slow rolling of waves filmed from a static camera position, may encourage feelings of relaxation. Related to tempo, is the *perceived duration* of the shot. The actual duration of a long shot full of people and action may well be identical to one of the close-up of a face, and yet the latter will be perceived as being longer. Hence, the organization of perceived duration is more complex than the actual duration of a shot.

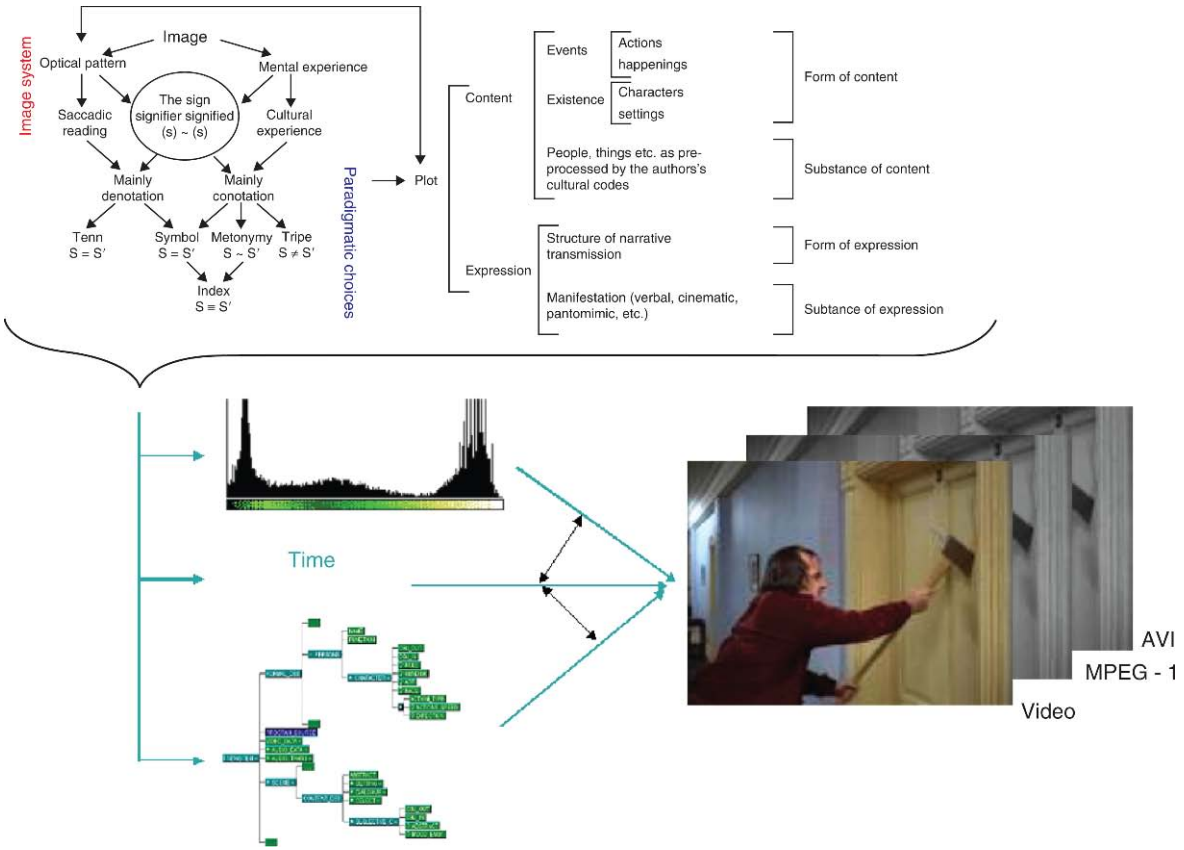
There are, on the perceptional level, various automatic methods to extract shot boundaries, namely the border between two different shots, forms of transitions between shots, as well as temporal or conceptual concepts (color, shape or texture-based). A good summary of these methods can be found in [11]. The advantage of automatic extraction is its low costs. The disadvantage, similar to the problems in automatic image indexing, is that it is exclusively organized around the sensory surface structures of media, i.e., the physical features, which are able to grant access to the representation of conceptual items but not to the higher semantics users wishes to access. Detailed description of schemata, that allow high-level semantic descriptions, namely for the essential content categories actor, appearance (features of a character), action (chronology of, or direction of, the action) location, and cinematographic devices

(e.g., camera and lens movement, camera position, etc) are described in [3], which also establishes an icon based annotation method, and [4]. Both approaches make use of hierarchical schemata structures.

Important for all of these annotation forms, might they be automatic, semi-automatic or manual, is a proper representation of time. The current ways or representing time are either based on:

- Physical time (full clock value (e.g., 7:45:23.76, where the last two items present ms), partial clock values (any sort of short base notation), time count values (numbers with a additional type string, e.g., 10S for "ten seconds"), and time context values, which are represented in three parts: a date field (YYY:MM:DD), a time field, and a timezone field). These descriptions allow the location of an annotation based on the start- and end-frame.
- Logic descriptions, as best described by [12], which are useful for a symbolic representation of a time relation instantiation in a triple structure, as requested by the Resource Description Framework (RDF) [13].
- Or as a hierarchical ontological description, that describes time not on its physical form but rather on its relevance for the shot content, e.g., The presented epoch (e.g., seventeenth century), season (e.g., Summer), or daytime (e.g., Midday) [3].

The essential concepts of video metadata are summarized in Fig. 1. The upper left corner describes the possibilities of interpretation of an image, the upper right part outlines the structural elements of a story (of which the visual information forms a subpart). Both together form the schematic basis for the annotation process, which can be performed in an automatic, semi-automatic or manual fashion, as described in the lower left part of Fig. 1, which are performed on various instantiations of the same content material in form of differently coded digital video, as outlined in the lower right part.



Video Metadata. Figure 1. Essential metadata concepts for the description of video content.

Sequence

The final level of generating meaning with video material to be considered is the way in which content of a shot can be affected by other shots.

- The meaning of a shot depends on the context in which it is situated.
- A change in the order of shots within a scene changes the meaning of the shot as well as the meaning of the scene.

Important is that not every combination of shots creates a meaning, but there are restricted conventions that can help create larger meaningful entities. The key elements for creating meaning by joining shots are *assertions* and *associative cues* [14].

An assertion is the relationship between two elements. There are many different types of such relationships. For example, the description of an attribute (such as *red* for a car) could be as important as a simple action (two men shaking hands).

Associative cues result from the combinations of the indicators that make the creation of meaning possible. One essential cue was already discussed, namely human action. The other one is surrounding space. Most human activities, human roles or objects are associated with specific locations. The conceptualization of space is, therefore, an elementary principle of the analysis and organization of video material. The sequential structures built up through juxtaposition provide the complex and intricate syntax for film narrative. The two essential works for structural metadata concepts for video are described by Aguierre Smith et al. [1] and in the MPEG-7 part 5 [15].

Aguierre-Smith designed the Stratification System to support an anthropological video study in the state of Chiapas, Mexico. The idea was to provide a number of researchers with random access to a video archive, in which video could be annotated with complementary or even contradictory descriptions. The video material was stored on a laserdisc. The annotations used in the Stratification System were keywords organized in hierarchical classes which were implemented as directory trees in UNIX. The novel feature introduced by Aguierre-Smith was the multiple partially overlapping annotation, where each annotation is related to a precise time index (begin and end frame). To provide a visual representation of the distinct layers of the representation, the Stratification System used a histogram, where the keyword classes are displayed as buttons along the *y*-axis and the time code (frame numbers on the laserdisc) form the *x*-axis. Aguierre-Smith's stream-based content representation for video enables the dynamic development of context while maintaining the completeness of the original footage. The notion of multiple partially overlapping annotations establishes the Stratification System as a breakthrough in the effective representation of video content, despite its weaknesses, i.e., the keyword approach and the lack of a true representation of the semantics of the video.

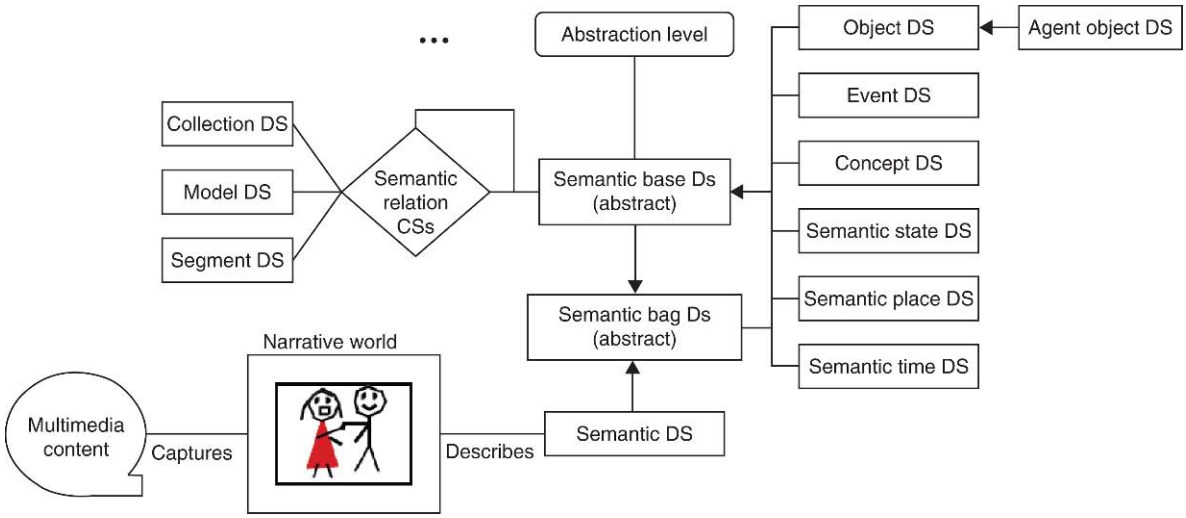
In MPEG-7's Part 5: Multimedia Description Schemes (MDS) the organization structure follows the overall hierarchical document structure, which at the end is instantiated as an XML schema file. Figure 2 visualizes description schemata that can be used to describe real-life concepts or narratives, which include objects, agent objects, events, concepts, states, places, times, and narrative worlds, all depicted by or related to video content. This means that the description schemata can be used to annotate the narrative in the video or the narrative world of the video, which then can be linked to the video.

The most relevant description schemata in Fig. 2 are:

- Semantic DS, which describes narrative worlds that are depicted by or related to the video content.
- Object DS, which describes objects in a narrative.
- AgentObject DS, which describes objects that are persons, organizations, or groups of people in a narrative.
- Event DS, which describes events in a narrative.
- Concept DS, which describes abstract semantic entities that cannot be described as abstractions of any objects, events, times, places, or states.
- SemanticState DS, which describes states or parametric attributes of semantic entities and semantic relations, at a given time or location in a narrative.
- SemanticPlace DS, which describes locations in a narrative.
- SemanticTime DS, which describes times in a narrative.

As often in MPEG-7, the description options are flexible, which allows the user to establish complex annotation structures. Yet, the taken approach also carries a number of problems, essentially:

- There are a great number of abstract elements, which are used to establish class structure. However,



Video Metadata. Figure 2. Illustration of the relationships of the tools for describing the semantics of video content.

abstract elements cannot appear in instantiations. When an element is declared to be abstract, a member of that element's substitutable class must appear in the instance document. To indicate that the derived type is not abstract, the XML namespace mechanism is used (xsi:type). Thus, a thorough understanding of schemata development is required, which makes instant schemata development for distinct domains hard, especially if the required schemata should cover simple descriptions, where the theoretical overhead is actually not required.

- The interlocked nature of schemata, providing an ontology-like yet general set of schemata for describing media semantics, makes it very difficult for a user to identify the appropriate schemata and to use them in isolation.
- Due to the lack of a fundamental data model the structures provided show inconsistencies and duplications, which makes manual schemata generation difficult.

There are, however, projects in real world domains, such as the TV Anytime Forum, that give an indication of how media-aware semantic structures, such as those provided by MPEG-7, will be used in the future. The TV Anytime Consortium (http://www.tvanytime.org) developed specifications for services based on consumer digital storage devices. The semantic structures, all written in XML Schema, are proprietary and cover the essential aspects of media description, i.e., content description, content referencing and location, rights management and protection, systems and transport. Though the TV Anytime schemata are similar to the equivalent structures in MPEG-7, they are less complex in their organizational structure. TV Anytime includes, for example, the MPEG-7 schemata on user-modeling, though without incorporating the complete MPEG-7 organizational overhead. Rather, TV Anytime uses MPEG-7 as a namespace and is thus able to incorporate only the required schemata.

Rights Management

MPEG-21 [10] addresses, among others, two important aspects when it comes to user interaction with video content (in fact MPEG-21 describes that for all types of multimedia): first the identification and declaration of video items, and second the description and protection of rights.

MPEG-21 is understood as a framework that supports users to access, exchange, trade and consume digital media. The basic unit for MPEG-21 is the Digital Item (DI) which can be distributed and on which transactions can be performed. The DI is defined as a "virtual container" for metadata (created based on MPEG-7) and content (in MPEG-1, MPEG-2 or MPEG-4 format). The Digital Item Declaration (DID) defines the resources (e.g., MPEG-4 files) and the related metadata (e.g., Dublin Core) which the author identified to be part of the DI. The Digital Identification framework then supports devices to uniquely identify various objects and items of the DI.

The provision of DIs only makes sense if there are mechanisms that allow to exploit those structures. For that MPEG-21 developed the Rights Expression

Language (REL), which is an XML-based machine-readable language that allows the specification of specific rights and conditions associated with the distribution of say video content. REL does not aim for replacing legal rights methods but rather specifies a grant specifying that a "principal" has a "right" over a "resource" under certain "conditions." REL is based on the Extensible Rights Markup Language (XCML) 2.0 which was selected by MPEG due to its expressiveness and unambiguaty. In addition MPEG-21 has designed the Rights Data Dictionary (RDD), which contains the key terms required to describe rights. Finally, the Intellectual Property Management and Protection Component (PMP) was established, which uses again metadata to allow the inclusion of protected and governed content into a DI. This mechanism facilitates terminals to process protected content. As MPEG-21 is at time of writing still relatively new the future has to show if that framework is functioning and thus will be accepted by the public.

Key Applications

The annotation of video is useful for the retrieval, reuse, manipulation, generation and distribution of video for domains, such as medicine, entertainment, distributed games, all sort of experience related applications and education.

Cross-references

► Image Metadata
► Multimedia Metadata

Recommended Reading

1. Adams B. Mapping the Semantic Landscape of Film: Computational Extraction of Indices through Film Grammar. Ph.D. thesis, Curtin University of Technology, Perth, 2003.
2. Aguierre Smith T.G. and Davenport G. The stratification system. a design environment for random access video. In Proc. ACM Workshop on Networking and Operating System Support for Digital Audio and Video, 1992.
3. Allen J.F. Maintaining knowledge about temporal intervals. Commun. ACM, 26(11):832–843, 1983.
4. Barry B. Mindfull documentary. Massachusetts Institute for Technology. Ph.D. thesis, MIT, Boston, 2005.
5. Davis M. Media Streams: Representing Video for Retrieval and Repurposing. Ph.D. thesis, MIT, Boston, 1995.
6. Extensible Markup Language (XML), http://www.w3c.org/XML/.
7. Gregory J.R. Some Psychological Aspects of Motion Picture Montage. Ph.D. University of Illinois, USA, 1961.
8. ISO MPEG-7 MDS Text of ISO/IEC 15938–5/FCD Information Technology – Multimedia Content Description Interface – Part 5: Multimedia Description Schemes, ISO/IEC JTC 1/SC 29/WG 11 N4242, 23/10/2001, 2001.
9. MPEG-4: ISO/IEC JTC1/SC29/WG11 N4668 March 2002, http://www.chiariglione.org/mpeg/standards/mpeg-4/mpeg-4.htm.
10. MPEG-21: ISO/IEC JTC1/SC29/WG11/N5231 Shanghai, October 2002, http://www.chiariglione.org/mpeg/standards/mpeg-21/mpeg-21.htm.
11. MPEG-7: ISO/IEC JTC1/SC29/WG11N6828 Palma de Mallorca, October 2004, http://www.chiariglione.org/mpeg/standards/mpeg-7/mpeg-7.html.
12. Nack F. AUTEUR: The Application of Video Semantics and Theme Representation in Automated Video Editing. Ph.D. thesis, Lancaster University, 1996.
13. Nack F. and Putz W. Designing annotation before it's needed. In Proc. 9th ACM Int. Conf. on Multimedia, 2001, pp. 251–260.
14. Resource description Framework (RDF), http://www.w3c.org/RDF/.
15. Tonomura Y., Akutsu A., Taniguchi Y., and Suzuki G. Structured video computing. IEEE MultiMedia, 1(3):34–43, 1994.

Video Partitioning

► Video Segmentation

Video Querying

Ichiro Ide
Nagoya University, Nagoya, Japan

Synonyms

Image Querying; Query by Example; Similarity in Video; Near-duplicate Video Retrieval

Definition

Video querying is a way to issue a query for retrieving video data from a database without explicitly expressing the contents of the video-in-search by high-level semantics, such as keywords. It, instead, expects that a user issues a query as a video segment in hand. The system will then search and retrieve similar video data from the database. Since video data is composed of sequences of still frame images, the search for similar video data in a database necessarily requires a huge computation cost when comparing the query with the data in the database. Thus, the main issue when

processing a video query is to reduce both the computation time for the search itself, and for the comparison of image sequences, to evaluate the similarity of the video segments. Video querying is an efficient way to issue a query when a user already has in hand an exact or an extremely relevant segment of the video in-search. For specific genres of video data, such as those obtained from a fixed surveillance camera or sports programs, querying based on certain motions, gestures or trajectories can be useful. This kind of querying can also be considered as a kind of video querying in the sense that it handles a query as a time-sequence transition of an interest point. In most other cases, keyword-based querying can be more efficient and realistic in general.

Historical Background

Although there are some academic works that make use of video segments as queries [3], video querying is still not widely employed for the purpose of video retrieval in general. It has, however, been a key technology for some real-world problems.

Most early works that make use of video segments as queries, aim to detect advertisements from a broadcast video stream; when a source video segment (an advertisement) is given, the system returns where it appears (repeatedly) in the archived long video stream [4,6]. This is often used by commercial sponsors to monitor if a television broadcaster has actually aired an advertisement at specific hours and/or for specific times, according to the contract.

Recently, the technology has started to be applied by copyright holders for monitoring video data illegally distributed on the internet. While the advertisement monitoring task requires exact matching, this task needs to handle near-duplicate video data, which are almost identical but may have slightly changed from the original data during the distribution process. The changes may include size, frame rate, compression, editing (ex. insertion of captions and logos) and so on.

The easiest approach to compare a pair of video data is to do so frame-by-frame. This approach is, however, highly expensive in computation cost; not only that the comparison needs to be done as many times as the number of the frames, but also the cost to compare each frame is itself relatively high compared to audio or text data. Thus, the key to realize this technology in a realistic speed is to reduce the size of the feature, and also to reduce the times of comparison.

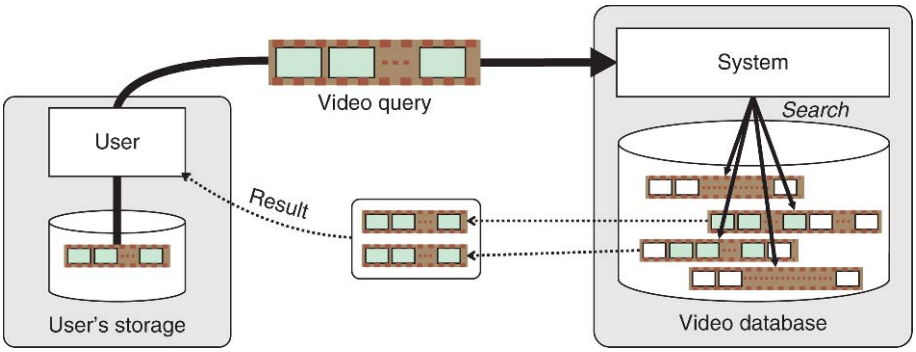
Foundations

Figure 1 shows the process flow of video querying. A user issues a query in the form of a video segment to a video database system. The query video segment should be readily available at the users hand, but it can also be provided externally through an interactive interface with a search function. For specific purposes such as action detection from fixed surveillance video, sports video, and so on, querying by motion, gesture, or trajectory [1, 2] can also be considered as a kind of video querying.

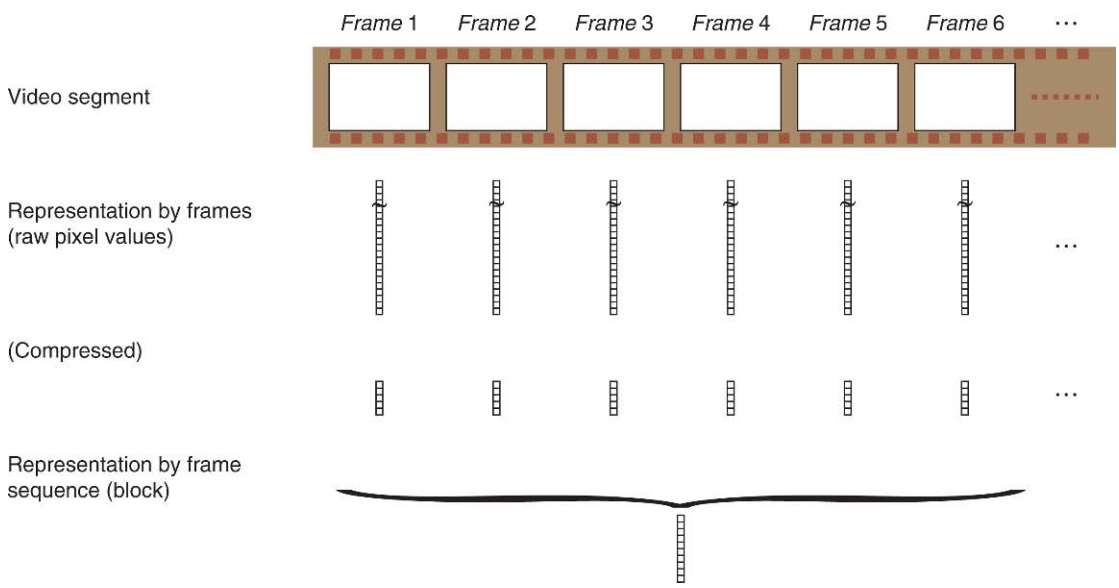
In order to process a video query, evaluation of the similarity between the query and the video data in the database is necessary.

The first question here is how to represent a video segment. Figure 2 shows an illustration of three representations explained below.

The simplest representation of a video segment is as a sequence of still frame images. Each image is represented by a feature vector composed of raw pixel values of all pixels in the frame image. It is, however, rarely the case



Video Querying. Figure 1. Process flow of video querying.



Video Querying. Figure 2. Representations of a video segment.

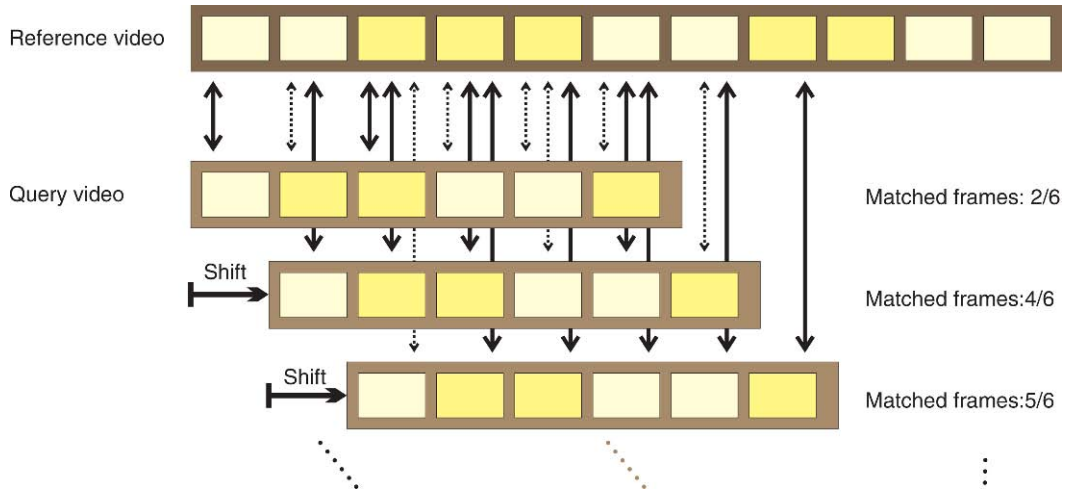
that exact pixel values between two video sequences are identical, due to noise, slight change of size / ratio, color adjustment, and so on. To cope with this problem, the image and the color resolutions are usually reduced, which is also the case for the original frame image used in the following two representations. Using raw pixel values is accurate, but the size of a feature vector tends to be large.

Since comparing large vectors is computationally expensive, it is better to reduce the size of a feature vector; the dimension of a feature space. Reduction of the dimension of a feature space can be done in various ways; from random sampling to hashing, signatures and transformations that preserve the original feature better by principal component analysis, and so on. On the other hand, the feature used to represent a frame can be more abstract features than the raw pixel values, such as edge, color histogram, or combinations of these features. In any case, representation in the compressed feature space does not usually guarantee the same distinguish ability as in the original feature space. When an identical result is needed, comparison in the original feature space after obtaining candidates in the compressed feature space is needed as a post-process in order to eliminate false positives. Note that in order to ensure that no false negative (oversight) exists among the candidates, the vector comparison method (distance measure) needs to theoretically guarantee it.

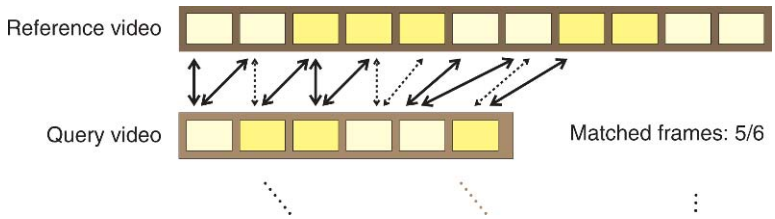
While the above two representations consider the feature of a video segment by a sequence of frame-wise features, it is more natural to represent a video segment by blocks of successive frames. In this case, image features from a fixed number of multiple successive frames represent a video segment. Ways to represent a block of frames can be a simple concatenation of frame-wise features (usually compressed afterwards), or can be more sophisticated features that make use of the redundancy of video data such as those used for video compression (MPEG and so on). This representation usually makes the comparison more robust and fast than the previous two representations, but it cannot represent, and accordingly compare and retrieve shorter segments than the block size.

The next question is how to compare the segments. Figures 3–5 show illustrations of the comparison methods explained below.

As shown in Fig. 3, the simplest method to compare the query video segment to the video data in the database (hereafter, "reference video") is to compare it against all possible segments in the reference video. In this case, the query video is compared by shifting it frame by frame against the reference video until it reaches the end. For each iteration of the comparison, all frames are sequentially compared. This method is not robust even against slight change in the frame sequence; frame-rate, and editorial effects such as slow-motioning or fast-forwarding.



Video Querying. Figure 3. Simple frame-by-frame comparison.



Video Querying. Figure 4. Frame-by-frame comparison by continuous dynamic time warping (DTW), also known as DP-matching.

An improved version of this method that takes the above-mentioned weak-point in consideration, is the method that employs continuous Dynamic Time Warping (DTW; also known as DP-matching). This method allows the comparison of a query video frame to multiple (usually a fixed number of frames after the frame matched in the previous iteration) reference video frames. As shown in Fig. 4, the method could cope, to some extent, with temporal expansion and contraction in both sides of the comparison.

Although there are many works on speeding-up the above-mentioned methods, they essentially require large computation costs due to the frame-by-frame shift and comparison approach.

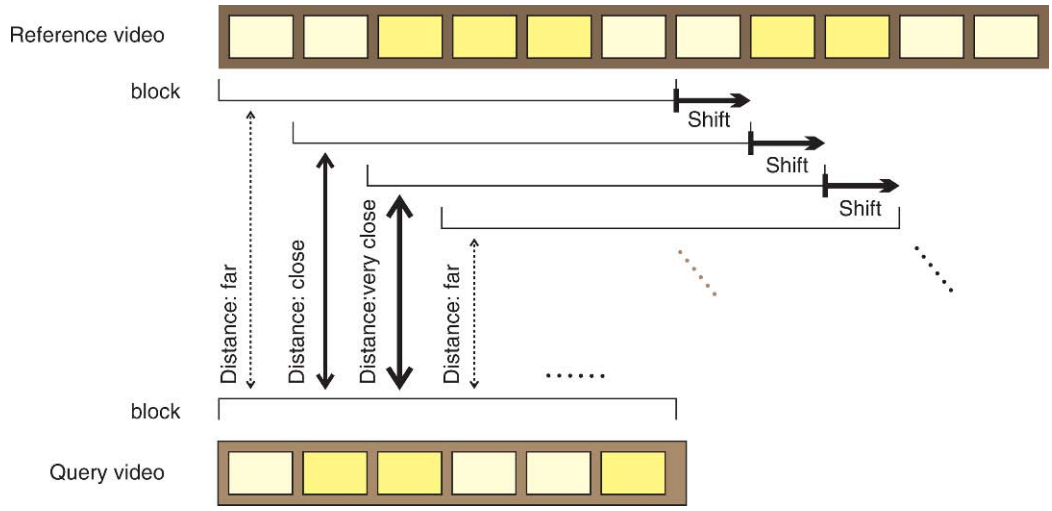
A different approach is to compare the video data as blocks of sequential frames. It does not necessarily have to be a simple concatenation of frame-wise features, but can be an integrated feature that represents the image features in the block. Block-wise comparison is robust to slight change of features in a frame sequence. Although in practice, it can tolerate time-wise expansion and contraction to some extent, it is mainly not due to the nature of the method, but rather due to the continuous nature of video contents. One of the fastest algorithm that takes this approach for video querying is the Time-Series Active Search method proposed by Kashino et al. [4], which reduces the computational cost by skipping the comparison under certain criteria.

Key Applications

As mentioned in the "Historical Background," the technology has been used by commercial sponsors of television programs and copyright holders of video contents.

Future Directions

Future applications include online retrieval of near-duplicate video data available on the internet. This application can be used not only for detecting illegal posting of copyrighted video data on the internet, but also for retrieval of web contents based on the video querying technology.



Video Querying. Figure 5. Comparison by block.

Another challenging task is to retrieve near-duplicate video segments from a video query based on feature points matching. This technology enables the retrieval of video segments shooting the same target from a different video camera, but current technology does not allow such processing in realistic time.

Cross-references

- ▶ Content-Based Video Retrieval
- ▶ Image Querying
- ▶ Multimedia Data Querying
- ▶ Principal Component Analysis

Recommended Reading

1. Aghbari Z., Kaneko K., and Makinouchi A. Content-trajectory approach for searching video databases. IEEE Trans. Multime., 5(4):516–531, 2003.
2. Chang S.-F., Chen W., Meng H.J., Sundaran H., Zhong D. VideoQ: An automated content based video search system using visual cues. In Proc. 5th ACM Int. Conf. on Multimedia, 1997, pp. 313–324.
3. Dimitrova N. and Abdel-Mottaleb M. Content-based video retrieval by example video clip. In Proc. Storage and Retrieval for Image and Video Database, 1997, pp. 59–70.
4. Kashino K., Kurozumi T., and Murase H. A quick search method for audio and video signals based on histogram pruning. IEEE Trans. Multime., 5(3):348–357, 2003.
5. Lienhart R., Effelsberg W., and Jain R. VisualGREP: a systematic method to compare and retrieve video sequences, Multime. Tool Appl., Kluwer, Hingham, MA, 10(1):47–72, 2000.
6. Lienhart R. Kuhmünch C., and Effelsberg W. On the detection and recognition of television commercials. In Proc. Int. Conf. on Multimedia Computing and Systems, 1997, pp. 509–516.

Video Representation

Ying Li
IBM T. J. Watson Research Center, Hawthorne, NY, USA

Synonyms

Video format; Video compression

Definition

Video representation, as the name implies, specifies a way of representing a video. While some work refers to video representation as the way to present or express video content through some extracted or summarized content units such as scenes or objects, the majority regard it more as the way the video content is stored. In other words, it is about *video format* which describes the sequence, structure and content of frames that create the moving video image, along with any possible audio or text (closed caption) information.

Historical Background

A video can be represented in either analog or digital formats. Typical analog formats include NTSC (National Television System Committee), PAL (Phase Alternating Line) and SECAM (Séquentiel couleur à mémoire, French for "Sequential Color with Memory,") which are commonly used in the domain of commercial broadcast. On the other hand, due to the rapid development of computer technologies, the

continuously improved transmission rate and the increasing ubiquity of digital video capturing devices such as digital cameras and camcorders, digital videos have become widely available. This has entailed the development of various kinds of digital video formats suited for various purposes. Some of the most popularly used ones are DVD (Digital Versatile Disc), QuickTime, H.261, H.263, H.264, MPEG-1 (Moving Picture Experts Group), MPEG-2, and MPEG-4.

Digital video was first introduced commercially in 1986 with the Sony D-1 format, which recorded an uncompressed standard definition component video signal in digital form instead of the high-band analog forms that had been commonplace until then. Due to the expense, D-1 was used primarily by large television networks. It was eventually replaced by cheaper systems using compressed data.

Consumer digital video first appeared in the form of QuickTime around 1990, which is Apple Computer's architecture for time-based and streaming data formats. Also around the same time, the ITU-T (Telecommunication Standardization Sector of International Telecommunication Union) Video Coding Experts Group (VCEG) developed H.261 standard which aims for video transmission over ISDN (Integrated Services Digital Network) lines whose data rates are multiples of 64 kbit/s [6]. Based on the experience from H.261, H.263 was developed, which is a low-bitrate compressed format for video-conferencing [7]. Its first version was completed in 1995 and provided a suitable replacement for H.261 at all bit rates. The next enhanced compression mechanism developed by ITU-T VCEG is the H.264 standard, also known as AVC (Advanced Video Coding) and MPEG-4 part 10. H.264 provides a significant improvement in capability beyond H.263 [8]. Now, most new video-conferencing products include H.264 as well as H.263 and H.261 capabilities.

Along the same direction, but targeting at different commercial applications, the Moving Picture Experts Group (MPEG) was formed in 1998 to establish an international standard for the coded representation of moving pictures and associated audio on digital storage media. So far, there have been three established MPEG standards: MPEG-1, MPEG-2, and MPEG-4. Specifically, MPEG-1 was originally designed to achieve VHS-video quality at 1.5 Mbit/s data rate [2]. MPEG-1 video is usually used for the Video CD (VCD) format. Compared to MPEG-1, MPEG-2 targets at a very high bandwidth with up to 40 Mbits/s data rate, and is widely used as the format of digital television signals that are broadcast by terrestrial (over-the-air), cable, and direct broadcast satellite TV systems [4]. It also specifies the format of movies and other programs that are distributed on DVD and similar disks. In late 1998, an effort on MPEG-4 was initiated which added many new features such as VRML (Virtual Reality Modeling Language) support for 3D rendering, object-oriented composite files (including audio, video and VRML objects), support for externally-specified Digital Rights Management (DRM) and various types of interactivity, besides absorbing many features from MPEG-1 and MPEG-2 [5]. Major MPEG-4 applications include streaming media on web, video-phone and broadcast television.

Foundations

Basics of Video Representation

Frame is the fundamental building block of a video, which is basically a rectangular image consisting of a series of lines, known as scan lines. A scan line contains a given number of pixels, and a pixel is represented by a certain number of bits (*a.k.a.* bits per pixel or bpp). A frame can consist of two or more fields, which are sent sequentially and displayed over time to form a complete frame. This kind of assembly is known as interlace. An alternative way is to send the entire frame as a single entity, which is known as a progressive scan frame.

The basics that are needed to represent a video or to define a video format is listed below.

- *Video resolution*, which equals the frame size measured in pixels.
- *Aspect ratio*, which describes the dimensions of video screens and video picture elements or pixels. The screen aspect ratio of a traditional television screen is 4:3, while a high definition television is 16:9. On the other hand, while pixels on computer monitors are usually square, pixels used in digital videos usually have non-square aspect ratios.
- *Color space*, which specifies the video's color representation. For instance, the YUV color model where Y stands for the luma component (the brightness) and U and V are the chrominance (color) components, is used in the PAL, NTSC, and SECAM composite color video standards. Other popularly

used color models include RGB (Red, Green and Blue components) and YCbCr (Y for luma component, Cb and Cr for the blue and red chroma components).

- *Depth of color*, which indicates the number of distinct colors that could be represented by a pixel. This depends on the number of bits per pixel (bpp).
- *Interlacing* or *progressive*. Interlacing was invented as a way to achieve good visual quality within the limitations of a narrow bandwidth. The horizontal scan lines of each interlaced frame are numbered consecutively and partitioned into two fields: the odd field (upper field) consisting of the odd-numbered lines and the even field (lower field) consisting of the even-numbered lines. NTSC, PAL and SECAM all use interlaced formats. In contrast, with progressive scan systems, each refresh period updates all of the scan lines. This results in a higher perceived resolution and a lack of various artifacts that can make parts of a stationary picture appear to be moving or flashing.
- *Frame rate*, which indicates the number of frames per time unit of the video. PAL and SECAM standards specify 25 frames per second (fps), while NTSC specifies 29.97 fps. Film is shot at a slightly lower frame rate of 24 fps. Generally speaking, the minimum frame rate for achieving an illusion of moving images is about 15 fps.
- *Bit rate*, which measures the rate of the information content in a video stream. Bit rate is only defined for digital videos, and is usually quantified in units of bit per second (bit/s or bps) or Megabit per second (Mbit/s or Mbps). A higher bit rate generally indicates a better video quality. For instance, VCD has a bit rate of about 1 Mbps while DVD, which has a much better quality, has a bit rate of around 20 Mbps. Variable bit rate (VBR) is a strategy to maximize the video quality and minimize the bit rate. For fast motion scenes, VBR uses more bits than it does for slow motion scenes of similar duration to achieve a consistent visual quality. However, for real-time and non-buffered video streaming applications where the available bandwidth is fixed, *e.g.*, for video-conferencing delivered on channels of fixed bandwidth, a constant bit rate (CBR) must be used.

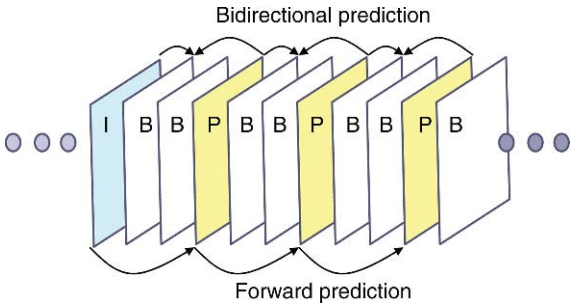
Video Compression

A video can occupy a vast amount of storage space when represented in digital form. For instance, suppose the video frame is of size 360 pixels by 288 pixels, and each pixel uses three color primaries with 8-bit precision for each color component, then each frame will occupy approximately 311 Kbytes. Further assume that the video is sent uncompressed at 24 frames/s, then the raw data rate for the video will be about 60 Mbps, which quickly amounts to 448 Mbytes for a 1 min video clip [10]!

Undoubtedly, there is a need of some techniques to compress or reduce the amount of video data so that videos can be conveniently and efficiently stored, transmitted and delivered, while retaining the original video quality as much as possible. Many research efforts on video compression started to emerge in 1990s. However, while there have been various kinds of compression techniques being proposed and standardized for various different application areas, most of them are based upon the same fact that video data contains both spatial and temporal redundancy. Specifically, to reduce the spatial redundancy, an intra-frame compression is applied which registers differences between parts of a single frame. On the other hand, to reduce the temporal redundancy, an inter-frame compression is exploited which registers differences between neighboring frames.

Below the basic spatial and temporal compression techniques that are applied in MPEG-1 video, are briefly discussed. Many other popularly used video compression standards such as H.261, H.263, H.264, MPEG-2 and MPEG-4, are more or less built upon similar techniques with certain improvements, extensions and modifications so as to meet different application requirements. These compression, together with the corresponding decompression algorithms are generally implemented in computer softwares as video codecs.

Frame Types in MPEG-1 There are basically three different types of frames in MPEG-1, namely, I-frame, P-frame and B-frame. Specifically, I-frames, also known as intra-coded frames, are coded independently without reference to any other frames. In contrast, P-frames (predictive-coded frames) obtain predictions from temporally preceding I- or P-frames in the sequence, which is known as forward prediction. Between the I- and P-frames, there may be zero or more bidirectionally predictive-coded frames, or B-frames. B-frames are interpolated between the preceding and/or upcoming I- or P-frames in the sequence. Figure 1 shows a group of pictures consisting



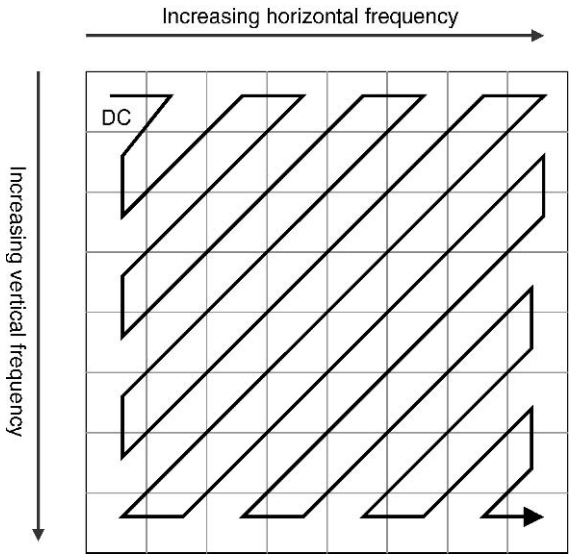
Video Representation. Figure 1. A typical group of pictures in display order in MPEG-1.

of one or more pictures of the three different types of frames.

Two different compression techniques are thus developed to code these three different types of frames. Specifically, intra-frame compression is applied to code I-frames, while inter-frame technique is applied to P- and B-frames. Both techniques are briefly discussed in the following sections.

Intra-Frame Compression *Macroblock*, which is a very important concept in MPEG coding, forms the basic building block of a coded frame. Specifically, a macroblock consists of a 16 sample array of luminance (gray-scale) samples together with one 8 × 8 block of samples for each of two chrominance (color) components. The 16 sample array of luminance samples is actually composed of four 8 × 8 blocks of samples, which form the units of data that are actually fed to the compression models. Notice that a lower resolution is used for the chrominance blocks because the human eyes resolve high spatial frequencies in luminance better than in chrominance.

At the heart of both intra-frame and inter-frame coding in MPEG is a mathematical transform known as discrete cosine transform (DCT). DCT transform is a mapping function from time or space domain to frequency domain. Given an 8 × 8 image block, a DCT transform will convert it into 64 DCT coefficients. Among them, the first coefficient is called *DC coefficient* or DC term, and the others, *AC coefficients*. Specifically, the DC coefficient contains the block's average intensity, *i.e.*, the low frequency information, while the rest 63 AC coefficients contain high frequency information. Figure 2 shows the zigzag ordering which approximately orders the DCT coefficients in ascending spatial frequency.



Video Representation. Figure 2. Zigzag scanning order of DCT coefficients.

The advantages of applying DCT in the compression models are twofold: i) it de-correlates the original signal. That is, the 64 DCT coefficients are independent of each other, thus they can be coded separately; ii) it distributes the signal energy to only a small set of coefficients. Generally speaking, after the transform, many AC coefficients in high frequencies will be very small, which may be discarded without, or with little, loss of visual quality.

After each of the four 8 × 8 blocks of luminance samples and two 8 × 8 blocks of chrominance samples of a macroblock is processed by the DCT, a quantization process is carried out on the obtained DCT coefficients, which aims to represent the coefficients of high spatial frequencies with less precision. Specifically, it divides and truncates each of the transformed coefficients by individual quantization values. These values are usually given in a quantization matrix, which typically contains higher values towards the lower right, thus giving several of the less important coefficients at high spatial frequencies a zero value. This allows the encoder to selectively discard non-important high frequency activity that the human eye cannot readily perceive.

The next step is to code the resulted DC and AC coefficients. Specifically, as there could be some correlation between the DC coefficients of neighboring

blocks, the DC term is coded separately from the AC terms using a predictive DPCM (differential pulse code modulation) technique. As for AC terms, they are first arranged qualitatively from low to high spatial frequency following the zigzag scan order as shown in Fig. 1. Such zigzag scan approximately orders the coefficients according to their probability of being zero. Finally, each nonzero AC coefficient is coded using a so-called *run-level* symbol structure where run refers to the number of zero coefficients before the next nonzero coefficient, and level refers to the amplitude of the nonzero coefficient. For more details on this part, please refer to [10].

Inter-Frame Compression Compared to the intra-frame compression, the inter-frame compression codes the difference between a frame and its reference frames, thereby exploiting the similarities or temporal redundancy from one picture to the next.

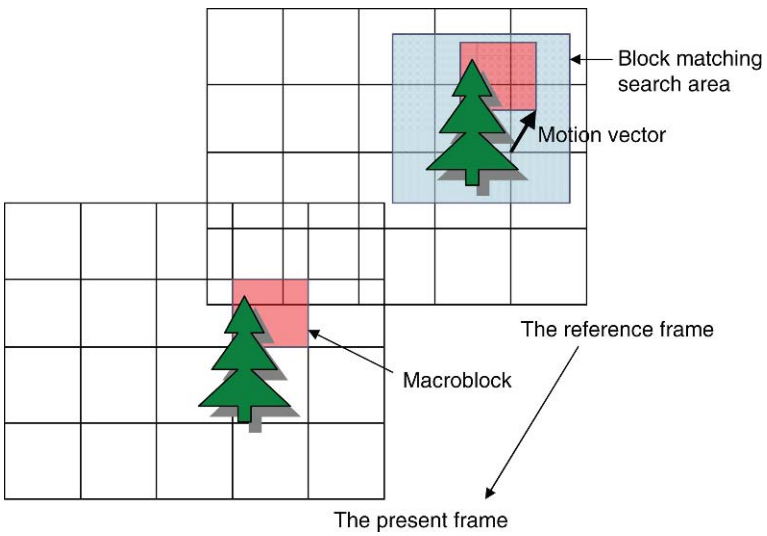
The most important concept in inter-frame compression is *motion estimation*. Specifically, motion estimation refers to the determination of the motion displacement between two frames which is expressed by motion vectors. In real video scenes, motion can be a complex combination of translation and rotation, which could be very difficult to estimate and may require large amount of processing. Nevertheless, translational motion could be easily estimated and has been successfully used for motion compensated coding.

There are two main classes of motion estimation techniques, namely, *pel-recursion* and *block matching*. Pel-recursion techniques are used primarily for systems where the motion vectors can vary from pixel to pixel. In contrast, in block matching techniques, there is only one single motion vector applied to a block of pixels. Due to its simple hardware realization and less computational complexity, the block matching approach has been more popularly used.

Figure 3 shows the basic idea of the block matching approach. Specifically, for each macroblock in the present frame, it is matched against a candidate block within a search area on the reference frame. If a good match is found based on certain criterion such as MSE (mean square error) and MAD (mean absolute distortion), the current macroblock will be represented by a motion vector pointing to the reference block, along with the DCT-coded residual. Otherwise, the macroblock will be intra-coded. Note that for B-frames, there are even more options for coding a macroblock, it could be forward predicted using the preceding I- or P-frame as the reference, backward predicted using the next I- or P-frame, and bidirectionally predicted using both preceding and succeeding I- or P-frames. Moreover, to achieve an even higher compression ratio, some macroblocks can be skipped.

Audio Compression

When a video stream also contains audio information, some kind of audio compression algorithm is usually



Video Representation. Figure 3. Block matching based motion estimation.

applied to condense the audio data, in addition to the aforementioned video compression. Generally speaking, many video codecs such as those introduced earlier (MPEG-1, MPEG-2, H.263 and H.264) also have corresponding specifications on encoding the embedded audio signals. For instance, MPEG-1 has defined three layers of audio coding in the ascending order of coding complexity: MPEG-1 Audio Layer 1 (MP1), Audio Layer 2 (MP2) and Audio Layer 3 (MP3) [3]. Below, a very high-level discussion on audio coding is given.

As with image or video compression, both lossy and lossless compression algorithms are used in audio compression. In both cases, techniques such as coding, pattern recognition and linear prediction are applied to reduce the amount of redundant information within the data. However, while lossless compression is good for the archival purpose, it generally produces much lower compression ratios due to the nature of audio waveforms and the fast-changing values of audio samples. Consequently, the lossy audio compression is much more popular. So far, it has found applications in video DVDs, digital television, streaming media, satellite and cable radio.

The key breakthrough in lossy audio compression is to rely on psychoacoustic models to identify the important audio signals that must be preserved, while throwing away unimportant ones. For instance, signals that can be perceived by the human auditory system must be kept, while perceptually irrelevant sounds could be ignored or coded with decreased accuracy as they are anyway very hard to hear. Typical examples of such sounds are those that have very high frequencies, or those that occur at the same time with other much louder sounds.

In particular, the following four human hearing characteristics, as recognized by the psychoacoustic model, are exploited in audio compression: *human hearing sensitivity*, *frequency masking*, *temporal masking* and *critical bands*. Specifically, by carefully conducting experiments, it is found that the general range of human hearing is from 20 Hz to 20 kHz, and the most sensitive range is from 2 kHz to 4 kHz. By frequency masking, it refers to the phenomenon in which a weak signal is made inaudible (*i.e.*, masked) by a simultaneously occurring stronger signal. In contrast, the temporal masking indicates the phenomenon in which a soft tone could not be heard immediately when there is a loud sound nearby. Finally, critical bands are obtained using a bandwidth classification scheme such that within each band, the width of frequency masking region is approximately uniform. As it could be seen that, by relying on these four characteristics, the compression algorithm could save bits on signals that are either out of human hearing ranges, temporally masked, or frequency masked within a critical band. During the actual coding, specific masking thresholds will be calculated by the psychoacoustic model to determine the audibility of each spectral component within particular critical band.

There are two general types of audio coding schemes, one in the frequency domain and the other, the time domain. Transform coding and subband coding belongs to the first category, while PCM (pulse code modulation), DPCM (differential pulse code modulation) and ADPCM (adaptive differential pulse code modulation) are in the second category.

Streaming Videos

Since the late 1990s, there have been some great advances in computer networking such as a higher network bandwidth, an increased access to networks, especially, the Internet, a widely accepted use of standard protocols and formats including TCP/IP (transmission control protocol/Internet protocol), HTTP (hyper text transfer protocol), and HTML (hyper text markup language), and the commercialization of the Internet. These advances, combined with increasingly powerful home computers and modern operating systems, have made streaming media practical and affordable for ordinary consumers.

By definition, streaming videos are the videos that are continuously received by and displayed to the end-users while the videos are being delivered from the providers. The name "streaming" refers to the delivery method of the medium rather than the medium itself. A video stream can be either *on demand* or *live*. On demand streams are stored on a server for a long period of time, and are available to be transmitted at a user's request. Live streams are only available at one particular time, such as the video stream of a live sporting event.

Some popular streaming video and streaming media technologies include Microsoft Windows Media, RealMedia from RealNetwork, Quicktime, MPEG-4, and Adobe Flash. And some major streaming video content providers include YouTube, Google Video, Netflix, Stage 6, and Metacafe.

Key Applications

With the proliferation of videos across the world, how to efficiently and effectively store, process, transmit, deliver and present them to end users has become an increasingly popular research topic. Video representation is undoubtedly one of these important issues. On one hand, the wide availability of videos has motivated the development of advanced video compression techniques, yet on the other hand, accomplishment achieved in the compression areas has made videos further distributed.

The benefits brought by efficient video representation are significant. Some popular applications that take advantage of advanced video compression techniques include: (i) HDTV and mobile broadcast terrestrial digital television (DTV); (ii) gaming consoles; (iii) network database services such as video library and video information provider; (iv) video on demand (VOD); (v) digital storage media such as CD-ROM videos, DVD movies and digital recorders; (vi) interactive communications such as two-way video phones, video-conferencing and videotex; and (vii) video surveillance.

Future Directions

Approximately every 3 to 4 years, a significantly new standard on video compression is developed. The latest generation is the MPEG-4 Part 10 advanced video coding (AVC). This is also known as the H.264 standard and is developed to achieve broadcast-quality video at much lower bit rates. The algorithm offers compression percentages that are twice as much as those offered by MPEG-4 Part 2 with the same image quality. It is also designed for a wide variety of consumer and pro AV (audiovisual) applications including TV set-top boxes and DVD players. MPEG-4 Part 10 (or H.264) improves older compression schemes by using significantly more sophisticated predictive coding, as well as variable block-size motion compensation and variable block-size integer discrete cosine transform [8]. However, the not-so-good news about MPEG-4 AVC is that it is extremely computation-intensive and requires a very high horsepower hardware platform to correctly implement it [9].

H.264 is now still in its preliminary stage in terms of adoption, although many vendors and analysts believe that it will eventually grab a big market share. It is expected that H.264 will be used by many digital video delivery networks and the new DVD standards such as high definition (HD)-DVD and Blu-Ray.

Another two directions to watch for future video compression are object-based video coding and scalable video coding (SVC). The basic idea of object-based video coding is to first segment a video scene into objects, then code them separately. MPEG-4 has actually explored such idea in its video codec, nevertheless, as object segmentation is a fairly challenging task and remains to be an open research topic, such object-based video coding still has a long way to go before it can truly take off. As for the scalable video coding, it is sort of a natural follow-on of H.264. In particular, it aims to provide a better way for dealing with wild cards such as disparate network types and different endpoint abilities, including display resolution and processing power [1].

In a word, while the network bandwidth has constantly increased in recent years, the emerging of high-definition videos effectively cancels out some of these gains. Moreover, considering the formidable high bit-rate of raw video data, video compression techniques are bound to be in great need for the future.

Cross-references

- ▶ Audio Representation
- ▶ Image Representation
- ▶ Video

Recommended Reading

1. International Standard ISO/IEC JTC1/SC29/WG11. N7315: Introduction to SVC extension of advanced video coding, 2005.
2. International Standard ISO/IEC 11172-2. Information technology – Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbit/s – Part 2: Video, 1993.
3. International Standard ISO/IEC 11172-3. Information technology – Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbit/s – Part 3: Audio, 1993.
4. International Standard ISO/IEC 13818-2. Information technology – Generic coding of moving pictures and associated audio information: Video, 2000.
5. International Standard ISO/IEC 14496-2. Information technology – Coding of audio-visual objects – Part 2: Visual, 2004.
6. International Telecommunication Union, Telecommunication Standardization Sector (ITU-T). ITU-T recommendation H.261: Line transmission of non-telephone signals – Video codec for audiovisual services at $p \times 64$ kbits, March 1993.
7. International Telecommunication Union, Telecommunication Standardization Sector (ITU-T). ITU-T recommendation H.263: Line transmission of non-telephone signals – Video coding for low bitrate communication, 2005.

8. International Telecommunication Union, Telecommunication Standardization Sector (ITU-T). ITU-T recommendation H.264: Advanced video coding for generic audiovisual services, 2005.
9. Kridel T. The future of video compression. Downloadable at: http://proav.pubdyn.com/Tech_Apps/Septemb912200532206PM.htm.
10. Mitchell J., Pennebaker W., Fogg C., and LeGall D. MPEG video compression standard. Chapman and Hall, New York, NY, USA, 1992.

Video Retrieval

► Video Sequence Indexing

Video Scene

► Video Scene and Event Detection

Video Scene and Event Detection

Noboru Babaguchi, Naoko Nitta
Graduate School of Engineering, Osaka University, Osaka, Japan

Synonyms

Video scene and event extraction

Definition

A video scene, also called a Logical Story Unit [7] or simply a story unit, can be defined as a semantically related consecutive series of image frames that depicts and conveys a high-level concept such as *event*, *topic*, *object*, *location*, and *action*, which constitutes a story in a video. Especially, an event can be defined as an incident or situation, which occurs in a particular place during a particular interval of time, for example – homerun in a baseball game, actor's entrance on stage, car explosion on a highway, etc. Under these definitions, video scene and event detection is to find all video intervals corresponding to a specific event from a given video.

Historical Background

Video scene and event detection has been an active research area in the community of multimedia signal processing and computer vision, and has attracted much interest in many applications such as multimedia information retrieval, video archive indexing and management, real-time adaptive streaming, and security monitoring.

For example, in sports videos, various score event scenes, e.g., touchdown in American football, homerun in baseball, and 3-point shoot in basketball are actually what viewers want to watch. Video retrieval based on such events is a typical example of multimedia information retrieval. Real-time adaptive streaming can also be achieved by transmitting only these score event scenes with full motion audio-video, while transmitting other irrelevant video segments with less information such as only keyframes. In addition, *annotations* of each event scene about "Five Ws and One H" – who, when, where, what, why, and how – are of much importance because such data enables efficient and effective video archive indexing and management. Event detection provides when and what attributes, e.g., *the homerun* in *the top of the 8th inning*. As another example, the security monitoring can be realized by detecting unusual events in surveillance videos in order to reduce the volume of data presented to security personnel. As can be seen, video scene and event detection is one of the most important tasks in video content analysis.

Foundations

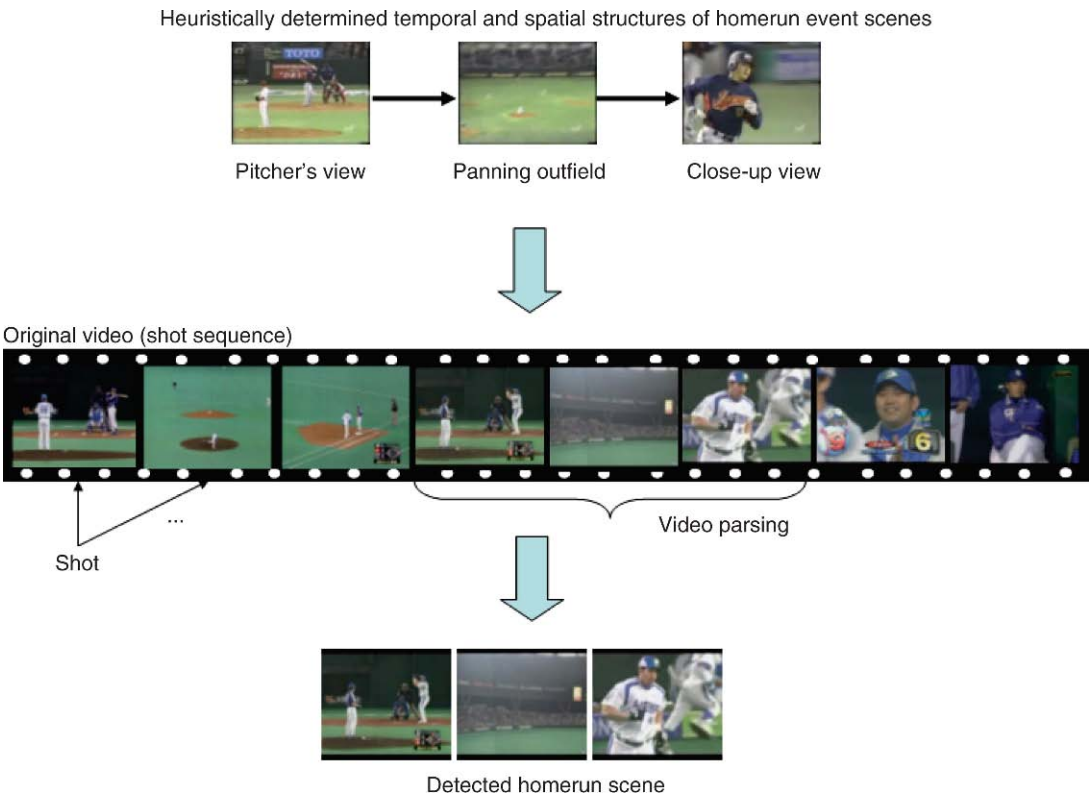
Video scene and event detection requires both video scene detection and event detection. Here, note that the target events can be categorized into two types: *known* and *unknown* events. For both types of event, shot detection, where a *shot* is defined as a set of contiguously recorded image frames, is usually the first step towards video scene and event detection. For unedited continuously captured videos, this can be realized by segmenting the video into fixed-length units. After that, there are mainly two types of approach: i) detecting events and scenes simultaneously based on heuristically/statistically determined rules, and ii) detecting scenes first and then classifying the detected scenes into predefined event classes. Both approaches can be applicable to detect known events, while the latter approach is usually used for detecting unknown events. More details are discussed below.

The primary event detection approach is to detect known event scenes based on heuristically determined rules. An example of known events is *homerun* in a baseball game. Broadcast baseball videos usually have

well-defined structures, for example, a typical home-run event usually consists of three or more shots, which starts from a pitcher's view, followed by a panning outfield and audience view in which the video camera tracks the flying ball, and ends with a global or close-up view of the player running to home base. These elemental shots consist of characteristic low- to mid-level image features such as color, texture, shape, motion, object trajectories, and human faces. Therefore, according to such *domain knowledge*, a series of certain types of shots is heuristically predefined as the temporal and spatial structures of the event scene, and then a video is parsed to find the corresponding event scenes as shown in Fig. 1.

It is noted that the approach discussed above focuses on single modality of the video data, in this case, the image stream. The single modality based approaches have low computational load, but the accuracy of event detection is low, as only using single modality is not able to fully characterize the events. For example, there are several cases for touchdown in American football games, such as the touchdown by pass, by running, and after turnover. It is not easy to construct a compact visual model that covers a number of cases of concern.

Alternatively, great emphasis has been placed on multimodality of the videos in recent years [2,3,8,11,13]. As is well known, the video data is composed of temporally synchronized *multimodal streams*: visual, auditory, text, and graphics streams, which are closely related to each other. The visual stream is a sequence of image frames, and the auditory stream is a mixture of a couple of auditory sources such as speech, music, and sound. In addition, the closed caption (CC) text, which is a transcript of the speech part of the auditory stream, can be viewed as the text stream for broadcast videos. The graphics stream is a sequence of overlays or video captions that have rich information about the content of the video data. The aim of multimodality based approaches is to improve the reliability and efficiency in analyzing the semantic content of videos. Although the computation for analyzing the visual stream is most costly, the use of other streams may be capable of reducing it. It should also be added that external metadata such as scenarios and web-cast game statistics is often used as one of the multimodal

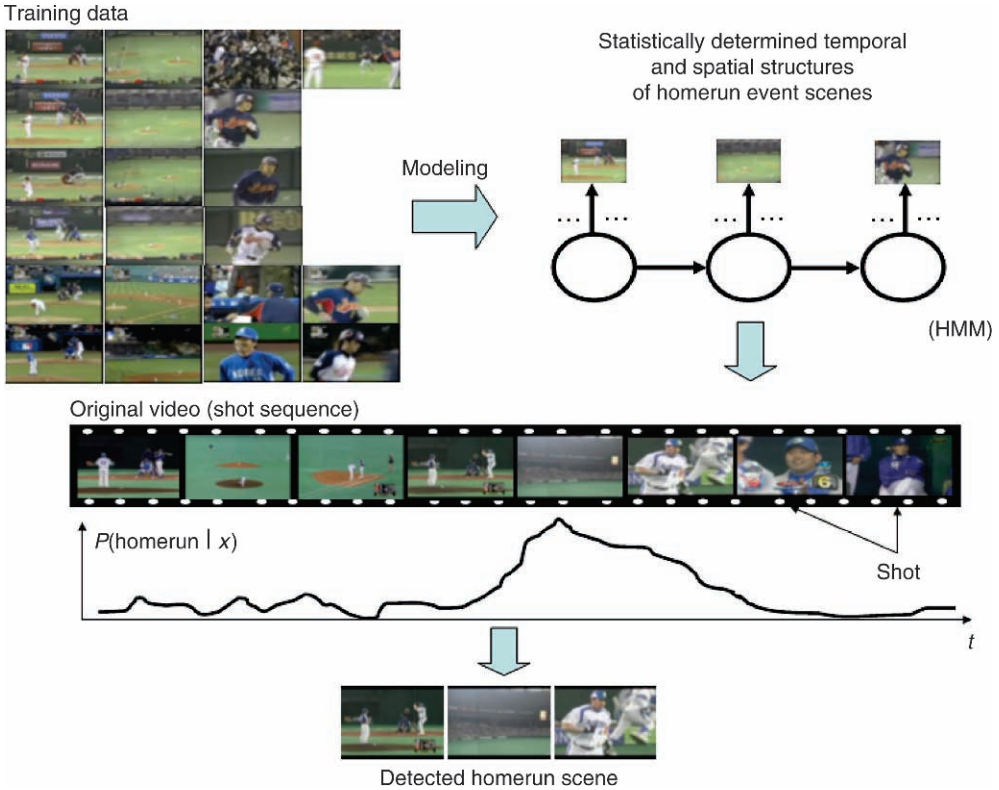


Video Scene and Event Detection. Figure 1. Heuristically determined rule-based approach for known events.

streams [4]; however, since the metadata is generally created independently from the video, it needs to be synchronized with other streams according to the semantic content at some point. Note that these multimodal streams are also used for other types of approaches described later.

These traditional rule-based approaches, where humans have to discover the domain knowledge and encode it into a set of programming rules, are too costly and incompetent for multimedia content analysis because knowledge for recognizing high-level events could be very complex, vague, or difficult to define. Therefore, on alternative solution is to use statistical models to learn the temporal and spatial structures of the event scene from training data which is given beforehand as shown in Fig. 2. Hidden Markov Models (HMMs), being a popular choice for probabilistic representation of sequences, are often used to learn these structures of a specific event and to detect the corresponding scenes from the video [15]. Several effective extensions to HMMs have been used to model more complex structures and to achieve better performance; especially, the coupled HMM (CHMM) has been developed to model interacting processes of multimodal streams.

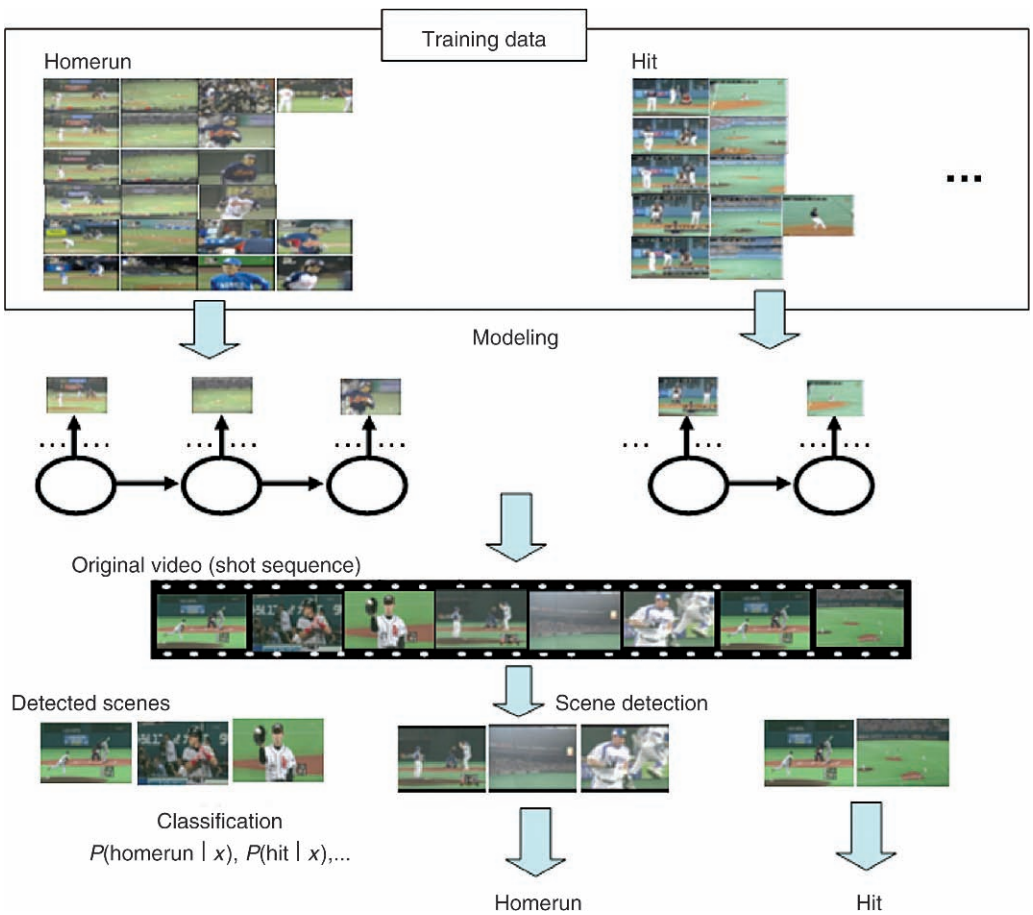
Now, there is a different type of solution for event detection, which is to firstly detect scenes and then classify the detected scenes into predefined event classes. In this case, semantically related shots are firstly grouped into a scene under the assumption that semantically related shots are usually temporally close to each other and have visual/audio similarity [7,9,10,12]. For example, if a person is talking in different shots, his/her speeches in these shots should present similar audio characteristics and these shots can be grouped as a scene. Then, the detected scenes are classified into a certain number of event classes based on the statistical model of each event [1]. More precisely, given a finite set of event classes $C = \{1,2,...,K\}$ and a scene s, probabilistic classification methods typically compute the probabilities $P(k|s)$ that s belongs to class $k \in C$, and then classify



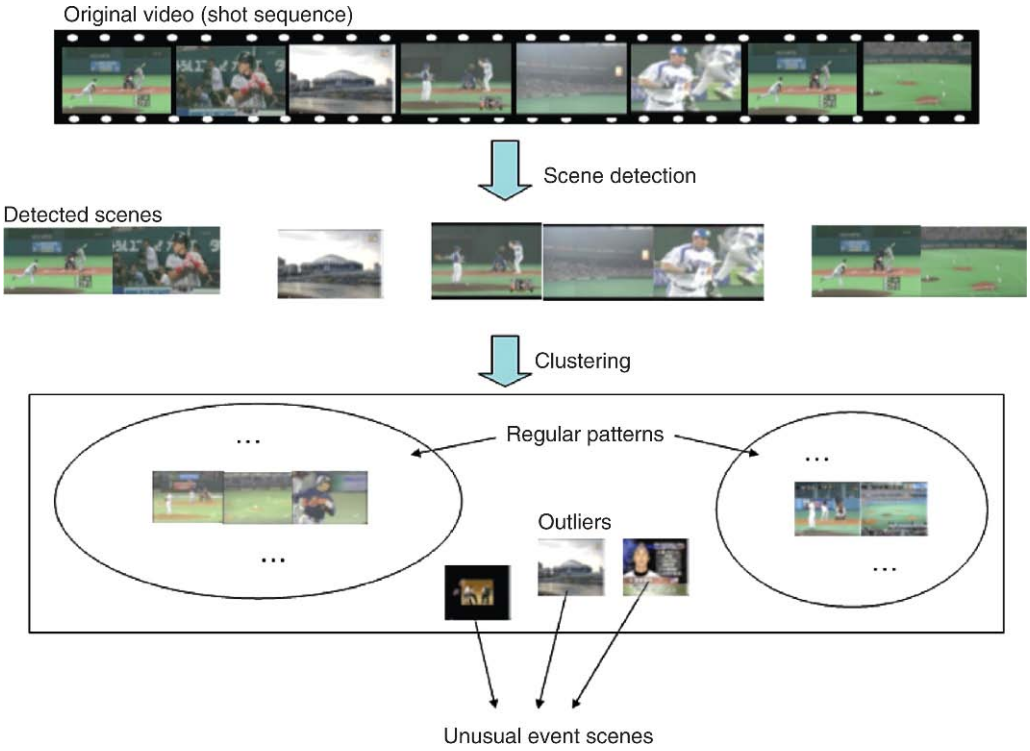
Video Scene and Event Detection. Figure 2. Statistical model-based approach for known events (with scene segmentation).

s into the class l that has the highest conditional probability $l = \text{argmax}_k P(k|s)$. Figure 3 shows the schematic diagram of this approach. In general, there are two ways of learning $P(k|s)$: generative and discriminative. Discriminative models strive to learn $P(k|s)$ directly from the training set without the attempt to model the observation s. Generative models, on the other hand, compute $P(k|s)$ by first modeling the class-conditional probabilities $P(s|k)$ as well as the class probabilities $P(k)$, and then applying the Bayes' rule as follows: $P(k|s) \propto P(s|k)P(k)$. Popular generative models include HMMs, while representative discriminative models include Support Vector Machines (SVMs) [6].

Although high classification accuracy can often be achieved by these statistical model-based approach, the drawback is that the model developed for one application usually does not fit for others. For example, the homerun model developed for baseball games is obviously not suitable for detecting the 3-point shoot scenes in basketball videos. Similarly, when detecting unknown events, it is quite unnatural to have their statistical models created from the training data since there would be so much variety for unknown events. Recently, much work has been done to detect unusual events as unknown events. The basis of detecting unusual events is the notion of usual events and an underlying distance/similarity metric. The existing work mainly solves this problem by finding regular patterns, which are typically found with clustering operations on a given dataset, as usual events. Here, unsupervised classification methods automatically partition the given data set into the predefined number of clusters. Some methods are even able to automatically guess the optimal number of clusters into



Video Scene and Event Detection. Figure 3. Statistical model-based approach for known events (without scene segmentation).



Video Scene and Event Detection. Figure 4. Statistical model-based approach for unknown events.

which the given data set should be partitioned. Then, unusual events are detected by finding outliers that do not fit the current set of usual event clusters [5] as shown in Fig. 4.

Key Applications

Video Indexing Systems: are used to insert metadata about events for the corresponding video scenes so that they can be easily found at a later time.

Video Scene Retrieval Systems: are used to find one or more media clips that match an event-based user query, which can be specified in a variety of ways, such as text and some media examples.

Video Summarization Systems: are used to represent one media clip or a collection of clips in a condensed form. The presentation style includes the *storyboard*, where the representative images of all/important events are displayed in the spatial layout, and *skims*, where only the important event scenes are concatenated. Event metadata can help decide which part of the content shall be presented or emphasized in a concise and coherent manner.

Future Directions

Despite a great amount of research efforts, the success of video scene and event detection methods is limited due to the big *semantic gaps* between the low-level features used by these systems and the high-level semantics of events, that is, the low-level features such as color, texture, and motion are insufficient to capture the intrinsic nature of events even throughout the videos of the same genre, mainly due to the differences of their recorded situation. Therefore, a conclusion that can be drawn here is that the key to the success of video scene and event detection systems lies in the degree to which the semantic gaps can be bridged.

Experimental Results

Common evaluation measures for event detection are:

$$precision = \frac{the\ number\ of\ correctly\ detected\ events}{the\ number\ of\ all\ detected\ events}$$

$$recall = \frac{the\ number\ of\ correctly\ detected\ events}{the\ number\ of\ the\ events\ that\ should\ be\ detected}.$$

In general, for most existing methods, there are accompanying experimental evaluations with these measures; however, it is quite difficult to compare their performances since a different dataset is used in these experiments.

Data Sets

The TREC Video Retrieval Evaluation (TRECVID) [14], sponsored by the National Institute of Standards and Technology (NIST) and other U.S. government agencies, has been used as a benchmark dataset to encourage research in video content analysis. Although video scene and event detection has not been explicitly stated as their tasks, there are some related tasks, e.g., story segmentation can be considered as the equivalence to scene detection and high-level feature extraction includes the detection of events such as Monologue, Physical violence, People walking, and Airplane takeoff. Further usage of such benchmark datasets is expected for video scene and event detection in order to create objective measurements of the performance of each approach.

Cross-references

- ▶ Content-based Video Retrieval
- ▶ Video Content Analysis
- ▶ Video Content Modeling
- ▶ Video Content Structure
- ▶ Video Segmentation
- ▶ Video Sequence Indexing
- ▶ Video Shot Detection, Video Summarization

Recommended Reading

1. Adams B., Amir A., Iyengar G., Lin C.-Y., Naphade M., Neti C., and Smith J.R. Semantic indexing of multimedia content using visual, audio and text cues. EURASIP J. Appl. Signal Processing, 2:1–16, 2003.
2. Babaguchi N., Kawai Y., and Kitahashi T. Event based indexing of broadcasted sports video by intermodal collaboration. IEEE Trans. Multimedia, 4(1):68–75, 2002.
3. Babaguchi N. and Nitta N. Intermodal collaboration: a strategy for semantic content analysis for broadcasted sports video. In Proc. Int. Conf. Image Processing, 1:13–16, 2003.
4. Chua T.-S. and Xu H. Fusion of AV features and external information sources for event detection in team sports video. ACM Trans. Multimedia Comput. Commun. Appl., 2(1):44–67, 2006.
5. Goh K.-S., Miyahara K., Radhakrishan R., Xiong Z., and Divakaran A. Audio-visual event detection based on mining of semantic audio-visual labels. MERL, TR-2004-008, 2004.
6. Gong Y. and Xu W. Machine Learning for Multimedia Content Analysis. Springer, Berlin, 2007.
7. Hanjalic A., Lagendijk R.L., and Biemond J. Automated high-level movie segmentation for advanced video-retrieval systems. IEEE Trans. Circ. Syst. Video Techn., 9(4):580–588, 1999.
8. Hauptmann A.G. and Smith M.A. Text, speech, and vision for video segmentation: the informedia project. In Proc. AAAI Symp. on Computational Models for Integrating Language and Vision, 1995, pp. 90–95.
9. Li Y. and Kuo C.-C.J. Video content analysis using multimodal information: for movie content exraction, indexing and representation. Kluwer, Norwell, MA, USA, 2003.
10. Lienhart R., Pfeiffer S., and Effelsberg W. Video abstracting. Commun. ACM, 40(12):55–62, 1997.
11. Merlino A., Morey D., and Maybury M. Broadcast news navigation using story segmentation. In Proc. 5th ACM Int. Conf. on Multimedia, 1997, pp. 381–391.
12. Rui Y., Huang T.S., and Mehrotra S. Constructing table-of-content for videos. ACM Multimed. Syst. J., 7(5):359–368, 1999.
13. Sundaram H. and Chang S.-F. Computable scenes and structures in films. IEEE Trans. Multimedia, 4(4):482–491, 2002.
14. The National Institute of Standards and Technology (NIST). TREC video retrieval evaluation. 2001–2007, http://www-nlpir.nist.gov/projects/trecvid/
15. Xie L., Xu P., Chang S.-F., Divakaran A., and Sun H. Structure analysis of soccer video with domain knowledge and hidden Markov models. Pattern Recogn. Lett., 25(7):767–775, 2004.

Video Search

- ▶ Video Sequence Indexing

Video Segmentation

Nevenka Dimitrova, Lalitha Agnihotri, Mauro Barbieri, Hans Weda
Philips Research, Eindhoven, The Netherlands

Synonyms

Video partitioning; Scene change detection; Shot-cut detection; Shot segmentation; Video shot-cut detection; Video chaptering; Logical story unit segmentation

Definition

Video (temporal) segmentation is the process of partitioning a video sequence into disjoint sets of consecutive frames that are homogeneous according to some defined criteria. In the most common types of segmentation, video is partitioned into *shots*, *camera-takes*, or

scenes. A *camera take* is a sequence of frames captured by a video camera from the moment it starts capturing to the moment it stops. During *montage*, camera takes are trimmed, split, and inserted one after the other to compose an *edited version* of a video. The basic element of an edited video is called *shot*. A *shot* is a contiguous sequence of frames belonging to a single camera take in an edited video. Content-wise, shots usually possess some degree of visual uniformity. A *scene* is a group of contiguous shots that form a semantically meaningful set.

If a video V is represented as a finite sequence of frames $V = (f_1,...,f_n)$, the temporal segmentation $S(V)$ of a video is a partition of the video into non-overlapping *video segments* v:

$$S(V) = \{v_1, v_2,...,v_m\},$$

with $v_i \cap v_j = \phi$ and $\cup v_i = V \quad \forall i, j \in [1,...,m]$, where a *video segment* is a finite sequence of consecutive frames $v = (f_p,...,f_q\} \quad 1 \le p \le q \le n$.

Historical Background

There have been many reports in the literature on methods for video shot-cut detection. The first report was published in 1992 (from a conference held in 1991) by Nagasaka et al., and described a way of indexing video objects by first detecting video cuts and then finding important objects in those frames [6]. Some of the reported methods worked on various ways of comparing pixel differences between frames. The methods presented in [5–7] operated in the spatial domain. In order to analyze an MPEG or motion JPEG stream, the frames in the stream would have to be fully decompressed, which was very computationally expensive. Later, other techniques have been applied in the compressed domain [2,12]. Some of these techniques require images obtained from an uncompressed stream to be fully compressed to take advantage of some of their special features. A method that was introduced by Swanberg et al. [7] actually went beyond the notions at that time that "cutting" and boundaries have to be at a consecutive frame level. The conceptual notion of "video parsing" was introduced where "video" is a multimedia language conveying more than the sum of the individual pixels and signals.

A model driven approach to digital video segmentation is presented by Hampapur and his colleagues in [5]. The paper deals with extracting features that correspond to cuts, spatial edits, and chromatic edits. The authors present extensive formal treatment of shot boundary identification based on models of video editing effects.

Arman et al. [1] have used a DCT approach on both JPEG and MPEG streams. For MPEG streams, only I-frames are analyzed. This implementation employed a two step approach. The first pass compares the images based on using selected DCT coefficients from selected blocks. Video frames are represented by a vector of this subset of DCT coefficients. Then the normalized inner product is subtracted from one another and compared to a threshold. If a potential cut is detected, the images can be decompressed for further processing.

A similar multi-pass approach has been used by Zhang et al. but their technique also analyzes the B and P frames in an MPEG stream [12]. The first pass compares the images based on DCT coefficients. The DCT comparison is based on a pair-wise block comparison algorithm which compares the DCT coefficients of corresponding blocks of consecutive video frames. For example, if block n in frame i has a DCT coefficient $C_{n,k}(i)$ where k is from 1 to 64 and n depends on the frame size, then the difference between two frames which are j frames apart is:

$$Diff_n = \frac{1}{64}\sum_{k=1}^{64}\frac{|C_{n,k}(i) - C_{n,k}(i+j)|}{\max(C_{n,k}(i), C_{n,k}(i+j))}$$

If $Diff_n$ is greater than some threshold t, then the block is different between the two frames. If the percentage of blocks changed is greater than some threshold t_b, it is considered to be a shot-cut. It is reported that t tends not to vary across video sources and can be easily experimentally determined. However, t_b is computed from the overall statistics for the values of the percentage of blocks that have changed which exceed the mean by approximately five standard deviations. In the second pass the number of motion vectors is compared to a threshold. If there are less motion vectors than some threshold, a shot-cut is determined to occur. This is based on the fact that if there are high residual errors, then there are no motion vectors, consequently, this block is sufficiently different and can not be predicted. If a large number of blocks can not be predicted, than it probably is a shot-cut. The performance of this method depends to a certain extent on the quality of the encoding scheme. An encoder with poor motion estimation may lead to poor shot-cut detection performances.

Yeo et al. have investigated using only the DC values of the DCT coefficients for comparing frames in the compressed domain [9]. They sum the DC difference between successive frames. If the difference is the maximum and n times larger than the next largest peak in a sliding window of frames, then it is a cut. They also detect "gradual transitions" such as dissolves and fades, by comparing each frame to the subsequent *k*-th frame over some time interval.

An alternative shot segmentation method was proposed by Zabih et al. in [11]. Their approach is based on the observation that, during shot-cuts, new edges appear far away from old ones, and old edges disappear in locations far away from new ones. Shot-cuts are detected by counting the number of edges *entering* and *exiting* pixels in consecutive frames.

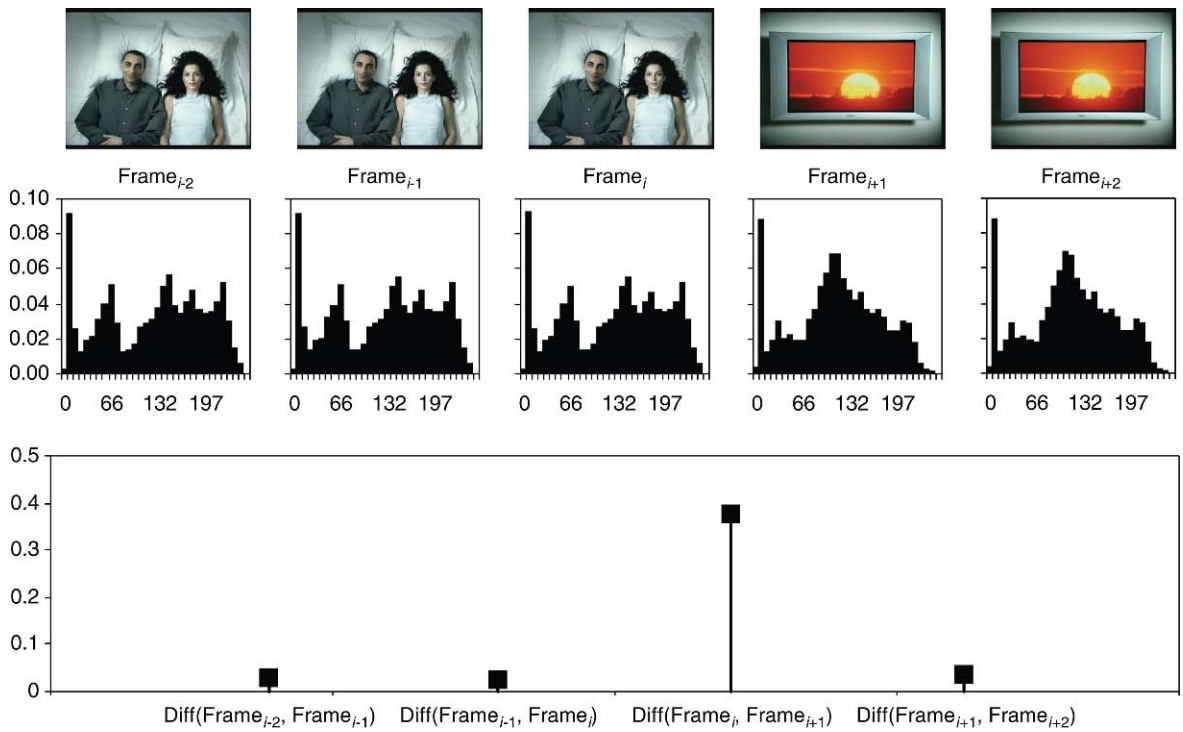
All of the above techniques have reported good results for shot-cut detection. However, a comparison of algorithms to detect shots boundaries has been performed by Boreczky and Rowe [2]. They selected and implemented some of the above algorithms. Their results showed that the simplest algorithms outperformed the more complicated methods and that DCT based algorithms had the lowest precision for a given recall.

Foundations

Shot-Cut Detection

Video segmentation aims at finding the temporal boundaries that separate visually homogeneous sequences of frames. The basic idea used by most methods for video segmentation is to compare subsequent frames for determining (based on some kind of threshold) when a significant change in the content occurs. For this purpose, it is necessary to define a metric for the comparison of subsequent frames and a thresholding method.

The key concept is visualized in Fig. 1. On the top of the figure, five frames across a shot boundary are shown. Below each frame, the corresponding luminance histogram is plotted. The plot shows on the horizontal axis the luminance values (from 0 to 255), and on the vertical axis the fraction of the pixels (from 0 to 1) in the frame image that have a particular luminance value. In this case, the metric for comparing two subsequent frames is the difference between the luminance histograms. The value of such distance is plotted in the graph at the bottom of the figure. As it can be seen in the histogram difference plot, the distance between the histograms of frames *i* and *i*+1 is considerably higher than



Video Segmentation. Figure 1. Example of a basic method for shot cut detection.

the differences between the histograms of other pairs of frames. The shot boundary located between the frames i and i+1 is easily detectable by means of a threshold located, for example, at 0.3.

A very simple metric for comparing frames is the *pixel-wise frame difference* consisting in the absolute difference of the intensity of the pixels between two consecutive frames. If $Y(x,y,n)$ and $Y(x,y,m)$ indicate the intensities of the pixels at position (x,y) in the frames n and m, the pixel-wise frame difference $\Delta_{n,m}$ is defined as follows:

$$\Delta_{n,m} = \sum_{x,y} |Y(x,y,n) - Y(x,y,m)|,$$

where the summation extends to the whole frame.

The pixel-wise frame difference is sensitive to noise, object and camera motion, and can lead to a high number of false detections. Better performances can be achieved by performing block-based differences instead of comparing single pixels.

Even better performances can be achieved by comparing image histograms. With respect to comparing pixels of consecutive frames, the histogram of a digital image increases robustness to noise, object and camera motion and is rotation invariant. Although image histograms do not retain any spatial information, they represent a good compromise between computational cost and performance for shot-cut detection.

If $H(i, n)$ represents the i-th histogram value for frame n and $H(i, m)$ represents the i-th histogram value for frame m, a simple but effective comparison metric is the *bin-to-bin* difference:

$$\Delta_{n,m} = \sum_{i} |H(i,n) - H(i,m)|$$

Shot-cuts are detected whenever the *bin-to-bin* difference exceeds a threshold. Other comparison metrics for histograms are the *intersection*, the *chi-square test* and the *correlation* defined as follows:

Histograms intersection	$\Delta_{n,m} = 1 - \sum_i \min(H(i,n), H(i,m))$
Chi-square test	$\Delta_{n,m} = \sum_i \frac{[H(i,n)-H(i,m)]^2}{[H(i,n)+H(i,m)]^2}$
Histograms correlation	$\Delta_{n,m} = 1 - \frac{\sum_i [H(i,n)-\mu_n][H(i,m)-\mu_m]}{\sigma_n \sigma_m}$

where μ_n and σ_n represent the mean and the standard deviation, respectively, of the histogram of the frame n.

The choice of a thresholding method is critical regardless of the comparison metric used. A too low threshold may create a high number of false detections, while a too high threshold may lead to many missed detections. A fixed threshold can be easily determined experimentally for one video. However, different videos might require different threshold values. To limit this problem, techniques have been proposed that link the threshold value to the mean and standard deviation of the distribution of frame differences across the whole video. Other methods use running averages in windows of multiple frames to detect shot-cuts.

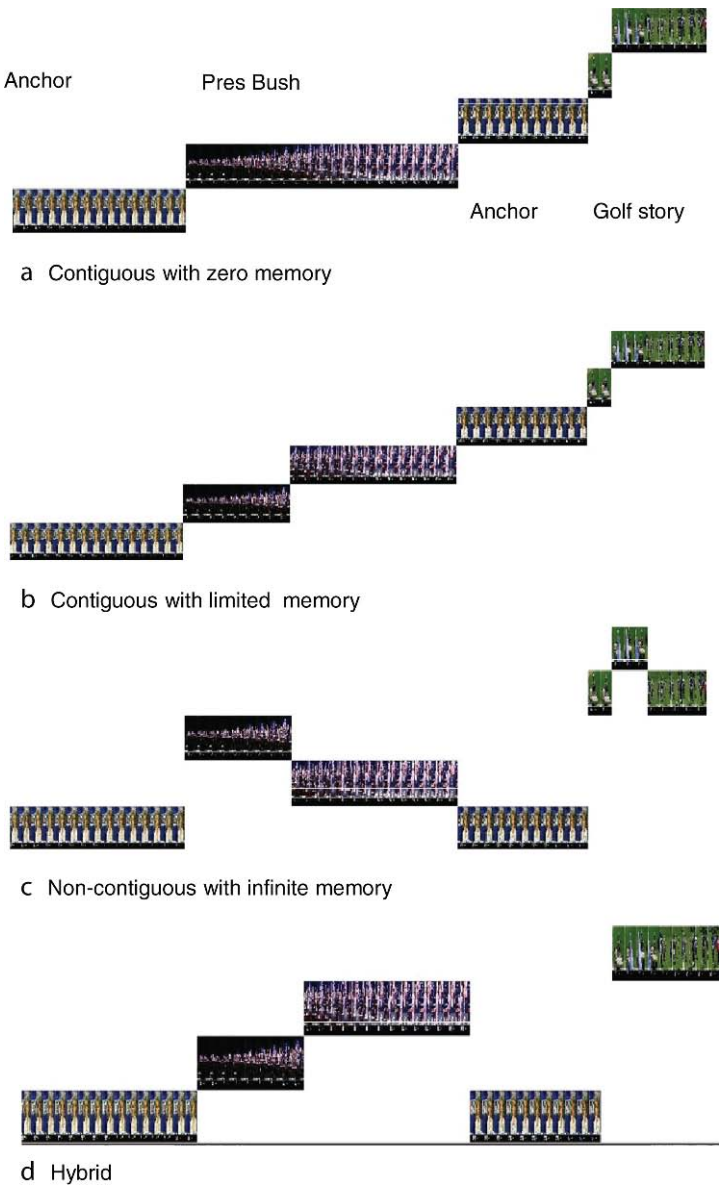
Scene Boundary Detection

The methods described above typically use the differences in contiguous frames to detect shot boundaries. In order to detect scene boundaries, groups of shots are determined that contain semantically related information such as the same physical location, or the same action. While a shot boundary is defined rather unambiguously, scene boundaries are much more difficult to capture. Different individuals tend to identify scenes in videos in different ways. This makes obtaining ground truth for the scenes a non-trivial task.

An overview and a benchmark of different methods for scene boundary detection are presented in [8]. Common techniques used for scene boundary detection use audio analysis and shot clustering [10].

One of the methods to automatically detect scene boundaries is by means of *superhistograms* [3]. Using this method, first the color histograms for individual frames are computed, and then they are merged into a single cumulative histogram called a *family histogram* based on a comparison measure. As new frames are added, the family histogram accumulates the new colors from the respective frames. However, if the histogram of a new frame is different from the previously constructed family histograms, a new family is formed. In the end, there are a few families of histograms to represent the entire video. This set of families is ordered with respect to the length of the temporal segment of video that they represent. The ordered set of family histograms is called superhistogram.

When computing the family histograms there are several dimensions to think about when making comparisons between frames based on (i) the amount of memory (time) (ii) contiguity of compared families, (iii) the representative structure for a family. Memory can be thought as one-dimension, that is, how much history of the computed family histograms is kept: a process that has "zero memory" and compares only the current frame with a previous frame histogram vs. a process with infinite memory that compares the current frame with all the previous family histograms. As shown in Fig. 2, the results can be quite different. Another dimension is time, which means whether contiguous or noncontiguous comparisons are made. The third dimension is the determination of a representative histogram for the family. This can be the average histogram or the histogram of a selected frame (e.g., the last frame). There are different



Video Segmentation. Figure 2. Merging strategies for creating family histograms.

merging strategies [15]: contiguous vs. noncontiguous comparison and comparison with a short term and longer term memory:

1. Contiguous with zero memory: A new frame histogram is compared with the previous frame histogram. This strategy allows to keep a pan sequence in a single family as the addition of new colors happens slowly and the difference between consecutive frames (in case the last frame is compared) is very small.
2. Contiguous with limited memory: Contiguous families are produced by comparing the new frame histogram only with the previous family histogram. For commercial detection applications such strategy needs to be used because, otherwise, many families would be created during the commercials when new colors are introduced with each shot. This enables the differentiation of commercials from regular programs where the families are usually longer.
3. Noncontiguous with infinite memory with family histogram comparison: This is useful to capture the global colors of a complete program. For applications for characterization of videos for clustering and retrieval, this strategy needs to be used as it captures all the colors present in the video in the least number of clusters possible. A new frame histogram is compared with all previous family within the same video.
4. Hybrid: First a new frame histogram is compared using (3) above and then the generated family histograms are merged using (1) or (2). This enables the combination of global information that is preserved when using the (3) strategy to the local information that is achieved by combining the families using strategies (1) or (2). In this case, the family histograms that are produced by (3) effectively become the frame histograms. These are now merged with each other to generate a hierarchical picture of the video. This enables to cluster colors that are repeated at regular intervals throughout the program while keeping pans etc. together in a single family.

An example of the results obtained using these four strategies is shown in Fig. 2 for a portion of a news video program.

Key Applications

Video segmentation is usually the first step in content-based automatic video indexing and retrieval. It is also widely used for video browsing and automatic video summarization. Additionally, another field of application is the object-based video compression. To correctly detect and track objects in video, a temporal segmentation step is in general needed.

Cross-references

- ▶ Video Representation
- ▶ Video Shot Detection
- ▶ Video Summarization

Recommended Reading

1. Arman F., Hsu A., and Chiu M-Y. Image processing on encoded video sequences. Multimed. Syst., 1(5):211–219, 1994.
2. Boreczky J.S. and Rowe L.A. Comparison of video shot boundary detection techniques. In Proc. SPIE 1996 Int. Symp. on Electrical Image Science and Technology. Storage and Retrieval for Image and Video Databases IV, vol. 2670, 1996, pp. 170–179.
3. Dimitrova N., Martino J., Agnihotri L., and Elenbaas H. Superhistograms for video representation. In Proc. Int. Conf. Image Processing, 1999, pp. 314–318.
4. Dimitrova N., Agnihotri L., and Jainschi R. Temporal video boundaries. In Video Mining, A. Rosenfeld, D. Doermann, and D. Dementhon (eds.). Kluwer, Boston, 2003, pp. 61–90.
5. Hampapur A., Jain R., and Weymouth T., Digital Video Segmentation. In Proc. 2nd ACM Int. Conf. on Multimedia, 1994, pp. 357–364.
6. Nagasaka A. and Tanaka Y. Automatic video indexing and full-video search for object appearances. In Visual Database Systems II, E. Knuth, L. Wegner (eds.). Elsevier, Amsterdam, The Netherlands, 1992, pp. 113–127.
7. Swanberg D., Shu C.F., and Jain R. Knowledge guided parsing and retrieval in video databases. In Proc. SPIE Storage and Retrieval for Image and Video Databases, vol. 1908, 1993, pp. 13–24.
8. Vendrig J. and Worring M. Systematic evaluation of logical story unit segmentation. IEEE Trans. Multimed., 4:492–499, 2002.
9. Yeo B. and Liu B. A unified approach to temporal segmentation of motion JPEG and MPEG compressed video. Multimed. Tools Appl., 1(1):81–88, 1995.
10. Yeung M. and Yeo B.-L. Video visualization for compact presentation and fast browsing of pictorial content. IEEE Trans. Circuits Syst. Video Technol., 7(5):771–785, 1997.
11. Zabih R., Miller J., and Mai K. A feature-based algorithm for detecting and classifying scene breaks. In Proc. 3rd ACM Int. Conf. on Multimedia, 1993, pp. 189–200.
12. Zhang H.J., Chien Y.L., and Smoliar S.W. Video parsing and browsing using compressed data. Multimed. Tools Appl., 1(1):89–111, 1995.

Video Sequence Indexing

Heng Tao Shen
The University of Queensland, Brisbane, QLD, Australia

Synonyms

Video retrieval; Video search; Video indexing

Definition

A video is usually defined as a sequence of high-dimensional feature vectors. Video sequence indexing consists of describing the content of video sequences from a video database to allow effective and efficient search and retrieval. Given a query video sequence, video sequence indexing aims to find its similar video sequences from a video database quickly. Typically, it includes the following major components: effective summarization of the high-dimensional sequence, effective access method for indexing the obtained summarization, and efficient query processing method.

Historical Background

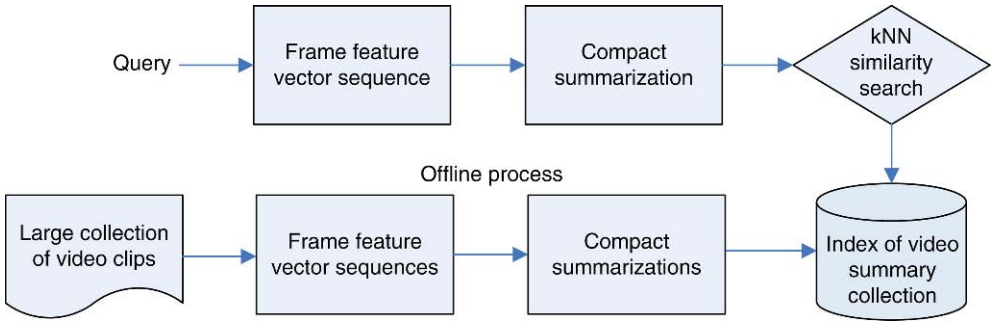
Video feature extraction and content analysis have been studied for several decades since the emergence of video data. Recently, video sequence indexing has attracted plenty of attention because of the huge amount of video data. With ever more heavy usage of video devices and advances in video processing technologies, the amount of video data has grown rapidly and enormously for various usages, such as filming, advertising, news video broadcasting, personal video archive, medical video data, and so on. Based on the statistics from Berkeley's "How Much Information" project 2003, 5,660 motion pictures are produced every year, amounting to almost 6,078 h. 21,264 television stations broadcast for about 16 h/day, producing 31 million hours of original programming annually, and huge amount of personal and organizational video data stocks are accumulating. Interestingly, the popularity of WWW enables enormous video data to be published and shared. The mainstream media is moving to the web. The media content online and video streaming is growing aggressively. While the pool of video data is super large, such as millions of videos currently available in web, video search engines provide users convenient ways for finding videos of their interests. Consequently, video sequence indexing has become a key part of the future of digital media [1].

The goal of video sequence indexing is to quickly find the similar videos to a query, and a video is usually represented by a sequence of high-dimensional frame feature vectors at a frame rate of 25–30/s. Due to the high complexity of video data, early work mainly focused on reducing the temporal redundancy of video data by identifying the shot which is the basic unit in video data. Shot detection is a fundamental task in video content manipulation. Representing the video by a small number of shots can reduce the data complexity in temporal dimension significantly. One central task in annual TRECVID evaluation is the shot boundary detection [2]. Many shot detection methods have been proposed and the field has matured [2,3]. Beside shot representation, recent work also considered the closeness of frames within a video across different shots and summarized similar frames into a compact representation [4,5]. Typically, temporal information is ignored for such aggressive summarizations. As the amount of video data increases explosively, there is also a trend to investigate various high-dimensional indexing methods to effectively manage and organize the video summaries for fast retrieval [5,6].

Foundations

Figure 1 shows the generic architecture for video sequence indexing. Given a collection of videos, their visual frame features, such as colour, tensity, and texture, is first extracted. Semantic features like objects and motions can also be further extracted from frame features. The sequence of frame features for each video is further summarized into compact representations which are indexed for fast retrieval. Given a query video, its features are first extracted and used to generate compact summarization. A similarity search is then performed on the indexed summary collection. The video similarity is approximated based on their compact representations.

To achieve efficient retrieval, most existing works emphasize on the summarization step to reduce the video data complexity and can be categorized into content-based and semantic-based approach. Content-based approach deals with frame visual features. In literature, two types of summarization techniques have been proposed: summarize the sequence as a statistical distribution and summarize the sequence into fewer representatives. The first type typically assumes that the frames are distributed in a model like Gaussian, or mixture of Gaussian [7]. In the second type, keyframe is often used [8]. Video signature introduces a



Video Sequence Indexing. Figure 1. A generic architecture for video sequence indexing.

randomized summarization method which randomly selects a number of seed frames and assigns a small collection of closest frames to each seed [4]. However, the selection of seeds may sample non-similar frames from two almost-identical sequences. Video Triplet (ViTri) models a cluster of similar frames as a tightly bounded hypersphere described by its *position*, *radius*, and *density* [5]. The ViTri similarity is measured by the volume of intersection between two hyper-spheres multiplying the minimal density, i.e., the estimated number of similar frames shared by two clusters. The total number of similar frames is then estimated to derive the overall similarity between two video sequences.

The semantic-based approach typically handles the motion trajectory of objects in a video sequence. Motion trajectories suggest the temporal movement of objects. Moving objects are first detected and segmented, and the motion trajectories of objects are then detected by analyzing inter-frame correspondences. Videos are represented based on these raw motion trajectories within a shot [8,9]. However, this approach is limited by the accuracy of object detection and only applicable so applications where the objects can be clearly identified.

In content-based image retrieval, the similarity/distance of two images is typically computed by the Euclidean distance. When extending the distance function to video sequences, many proposals have been proposed. One widely used video similarity measure is the approximate percentage of similar frames shared by two sequences [4,13]. Some studies also take temporal information into consideration. In [5], a time warping based approach is proposed to deal with video temporal variations, including local shifting. Hausdorff distance is used to measure the maximal dissimilarity between two shots [2]. The Earth Mover's Distance was introduced in computer vision to better approach human perceptual similarities. It models similarity as the amount of changes necessary to transform one image feature into another one. A tight and computationally simple approximation of the Earth Mover's Distance was proposed for matching video sequences [1]. Other measures widely used in time series, such as Longest Common Subsequence [15], Edit Distance and its extensions [3], can also be extended for comparing video sequences. However, all these measures need to compare most, if not all, frames pairwise. The full similarity computation requires storage of the entire sequence and time complexity is typically quadratic.

To index low-dimensional video trajectories, trajectories are typically first split into segments which are typically represented by Minimum Bounding Rectangles (MBR) [11]. The obtained MBRs can then be indexed by some tree structures like Trajectory Bundle Tree (TB-tree) [10]. Efficient query processing typically deploys the GEMINI-like framework which pruning the search space by the established lower bound lemma. When temporal information is not considered, existing high-dimensional indexing structures and their query processing methods can be directly applied [4,13].

Key Applications

There are many applications for video sequence indexing. Typically applications include video search, video monitoring, video copy detection, video annotation, video digital library, etc.

Future Directions

The indexing structures that are used to manage high-dimensional video sequences have not been well

explored in the database community. There is the also an urgent need for online monitoring, indexing and searching, as the media content online and video streaming is growing aggressively.

Experimental Results

In general, for every presented method, there is an experimental study in the corresponding reference.

Data Sets

TRECVID provides reasonably large video data sets for testing and experimental purposes [14].

URL to Code

Published code for video sequence indexing is rather limited in the database society.

Cross-references

- ▶ Indexing
- ▶ Similarity Measure
- ▶ Time Series
- ▶ Video Summarization

Recommended Reading

1. Assent I., Wenning A., and Seidl T. Approximation Techniques for Indexing the Earth Mover's Distance in Multimedia Databases. In Proc. 22nd Int. Conf. on Data Engineering, 2006, p. 11.
2. Chang H., Sull S., and Lee S. Efficient video indexing scheme for content-based retrieval. IEEE Trans. Circuits Syst. Video Technol., 1999.
3. Chen L., Özsu M.T., and Oria V. Robust and Fast Similarity Search for Moving Object Trajectories. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2005, pp. 491–502.
4. Cheung S. and Zakhor A. Efficient video similarity measurement with video signature. IEEE Trans. Circuits Syst. Video Technology, 2003.
5. Chiu C.-Y., Li C.-H., Wang H.-A. Chen C.-S., and Chien L.-F. A time warping based approach for video copy detection. In Proc. 18th Int. Conf. on Pattern Recognition, 2006, pp. 228–231.
6. Cotsaces C., Nikolaidis N., and Pitas I. Video shot detection and condensed representation: a review. IEEE Signal Process. Magaz., 23(2):28–37, 2006.
7. Iyengar G. and Lippman A. Distributional clustering for efficient content-based retrieval of images and video. In Proc. Int. Conf. Image Processing, 2000, pp. 81–84.
8. Keogh E.J., Palpanas T., Zordan V.B., Gunopulos D., and Cardle M. Indexing large human-motion databases. In Proc. 30th Int. Conf. on Very Large Data Bases, 2004, pp. 780–791.
9. Lee J., Oh J.-H., and Hwang S. STRG-index: spatio-temporal region graph indexing for large video databases. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2005, pp. 718–729.
10. Pfoser D., Jensen C.S., and Theodoridis Y. Novel Approaches in Query Processing for Moving Object Trajectories. In Proc. 26th Int. Conf. on Very Large Data Bases, 2000, pp. 395–406.
11. Rasetic S., Sander J., Elding J., and Nascimento M.A. A trajectory splitting model for efficient spatio-temporal indexing. In Proc. 31st Int. Conf. on Very Large Data Bases, 2005, pp. 934–945.
12. Sarukkai R.R. Video search: opportunities and challenges. Keynote Speech at ACM Multimedia Information Retrieval Workshop for ACM Multimedia Conference, 2005.
13. Shen H.T., Ooi B.C., Zhou X., and Huang Z. Towards effective indexing for very large video sequence database. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2005, pp. 730–741.
14. TRECVID. http://www-nlpir.nist.gov/projects/trecvid/. 2007.
15. Vlachos M., Hadjieleftheriou M., Gunopulos D., and Keogh E. Indexing multi-dimensional time-series with support for multiple distance measures. In Proc. 9th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, 2003, pp. 216–225.

Video Shot Detection

Chong-Wah Ngo
City University of Hong Kong, Hong Kong, China

Synonyms

Shot boundary detection; Camera break detection

Definition

Shot is a contiguous sequence of video frames with smooth and continuous camera motion. Shot can thus be viewed as an uninterrupted video frame sequence of space, time, and graphical configuration. The boundary between two shots is called a camera break (or video edit). Based on the transitional properties of video edits, there are two major types of camera breaks: *cut* and *gradual transition*. A camera cut is an instantaneous change from one shot to another. The gradual transition can further be categorized as *wipe* and *dissolve*. A wipe is the moving transition of a frame (or a pattern) across the screen that enables one shot to gradually replace another. A dissolve superimposes two shots, where one shot gradually appears while the other fades out slowly. Fade-in and fade-out can be considered as the special cases of dissolve, by replacing one of the shots as a constant image sequence (e.g., black image sequence). In contrast to cut, wipe and dissolve involve gradual transitions with no drastic changes between two consecutive frames, and hence, are relatively difficult to identify by computer.

Historical Background

Shot is regarded as the elementary unit of video structure, often serving as the basic component for video indexing, browsing and retrieval. Shot detection has been extensively studied since the emergence of the topic: content-based video retrieval in mid 1990. The early researches are unified under the theme of *video structuring and representation*, which targets for decomposing videos into smaller unit facilitating content analysis and understanding [15]. The scope of studies includes the detection of cut and gradual transition [15], detection efficiency [11], feasibility of thresholding [10], compressed versus uncompressed domain [11], and framework of detection [13].

Shot detection is mostly based on the analysis of visual information, or more precisely the changes of visual signal over time. For instance, an abrupt change of signal between frames can imply a camera cut. To date, there exist numerous approaches for shot detection which can be broadly classified according to the visual features being used and the type of camera breaks being analyzed. Based on feature, the existing approaches can be categorized as pixel-based, block-based, color-based, texture-based [14], motion-based [2], and slice-based [8]. These approaches can operate in either a compressed or uncompressed domain with slight modification. Uncompressed domain detection refers to the pixel-to-pixel processing of video frames to extract the desired features for analysis. Compressed domain detection refers to the direct processing of compressed information without explicitly decompressing video signals. The compressed features popularly used are DCT coefficients and MPEG motion vectors. Since the amount of information to be processed is relatively small, compressed domain detection has the advantage of speed efficiency. For instance, in [15], DC images which are the thumbnail version of original video frames are extracted directly from MPEG videos for rapid shot detection. In [15], motion vectors of P- and B- frames are extracted directly from MPEG videos for analysis.

Compared to cut, gradual transition (GT) is difficult to detect due to the absence of sharp signal changes. While cut normally involves only two frames, GT involves a sequence of frames ranging from several to even hundreds of frames. The first algorithm for GT detection is twin-comparison [15], which utilizes two thresholds to locate GTs. Another well-known algorithm is based on the video production model proposed in [1], which model the distribution of GT signals for detection by reversing the linear equation used to synthesized GTs in video production. GT detection has also recently been cast as a pattern analysis and learning problem. In [5–7,13], various classifiers are learned to recognize different types of GTs.

TRECVID has been the rendezvous where researchers meet and compete their shot detection algorithms since early 2000 [9]. The series of activities, over the past few years, have led to several sophisticated frameworks that perform reliably and excellently for the detection of camera cuts and GTs [9,13]. Based on the reported results, the best performing systems can always achieve more than 80% of recall and precision for cut and GT detection. Most systems also run faster than real-time.

Foundations

Cut

Camera cut is the most common type of shot boundary. In contrast to gradual transitions, most cuts involve no transitional frame. Thus, direct comparison between two frames is usually sufficient to attain satisfactory performance. A straightforward approach is to compute the mean absolute change of intensity between two frames at time t and t + 1. This simple approach, however, is sensitive to object and camera motion. An alternative version is to compute the energy changes of DC frames, which is the smaller and smoothed version of the original frames, extracted from MPEG videos to reduce the sensitivity [11]. In [8], the sequence of DC frames is further sub-sampled along the time dimension to form spatial-temporal slices. This approach detects cuts by analyzing the changes of temporal patterns in slices.

The most popular approach perhaps is via the comparison of intensity or color histograms between two contiguous frames. Color histogram quantizes the color space and captures the ratio of color components in a frame as a distribution. As a result, the histogram-based approaches are more invariant to local and global motion changes. To also capture spatial information in a color histogram, a more popular version is the block-based histogram: divide frames into blocks and compute a histogram for each block.

Other visual features being considered for cut detection include edge information, which captures the structural discontinuity of frames. In [14], edge change ratio is used to model the discontinuity by

computing the number of edge pixels entering and existing between two frames. In addition, motion discontinuity is also studied in [2]. The cut detection is conducted by computing the global dominant motion of frames and accounting the number of points that do not fit the computed motion model. This approach can also be used for detecting gradual transitions.

Dissolve

Some existing works on the detection of dissolve transitions are based on the video production model [1]:

$$f(x,y,t) = (1-\alpha(x,y,t))g(x,y,t) + \alpha(x,y,t)h(x,y,t) \quad (1)$$

where f is a dissolved frame superimposed by two frames g and h at time t. Typically, g and h are frames from two different shots. The transition function α characterizes, either linearly or non-linearly, how f is dissolved over time as a result of mixing g and h. Usually $0 < \alpha(x, y, t) < 1$ with the condition $\alpha(x, y, t) \leq \alpha(x, y, t+1)$. Since (1) is irreversible, apparently, detecting and classifying dissolves is a difficult task. To simplify the problem of detection, various assumptions have been made on (1). These assumptions can lead to plateau a effect [11] and parabolic curve of variance [1]. Take $\alpha(x, y, t) = \alpha(t)$, $g(x, y, t) = g(x, y)$ and $h(x, y, t) = h(x, y)$, (1) becomes

$$f(x,y,t) = (1-\alpha(t))g(x,y) + \alpha(t)h(x,y) \quad (2)$$

In other words, f is a dissolved sequence of two static shots g and h in $t = [t_1, t_2]$. Let $\mathcal{F}(t) = f(x, y, t)$, by taking the frame difference, the following can be derived:

$$\frac{\mathcal{F}(t) - \mathcal{F}(t+k)}{\mathcal{F}(t-k) - \mathcal{F}(t)} = \beta(t,k) \quad (3)$$

where $\beta(t,k) = \frac{\alpha(t+k)-\alpha(t)}{\alpha(t)-\alpha(t-k)} > 1$ and $t = [t_1 + k, t_2 - k]$. If $k > t_2 - t_1 + 1$, plateau effect will be exhibited and this effect can be exploited effectively for dissolve detection [11].

Equation(2) can be further simplified by assuming $\alpha(t)$ as a linear function, $\alpha(t) = \frac{t-t_1}{t_2-t_1}$ for instance. This terms leads to a formula in terms of variance:

$$\sigma_f(t) = (\sigma_g + \sigma_h)\alpha^2(t) - 2\sigma_g\alpha(t) + \sigma_g \quad (4)$$

where $\sigma_f(t)$, σ_g and σ_h are the variances of, $g(x, y)$ and $h(x, y)$. Since $\sigma_f(t)$ is a concave upward parabolic curve, dissolves can be detected simply by locating parabolic curves [1,8]. The limitations of (3) and (4) are mainly due to the linearity assumption of $\alpha(t)$ and the static sequence assumption of shots g and h. As a result, most detectors are generally very sensitive to noise, camera and object motions.

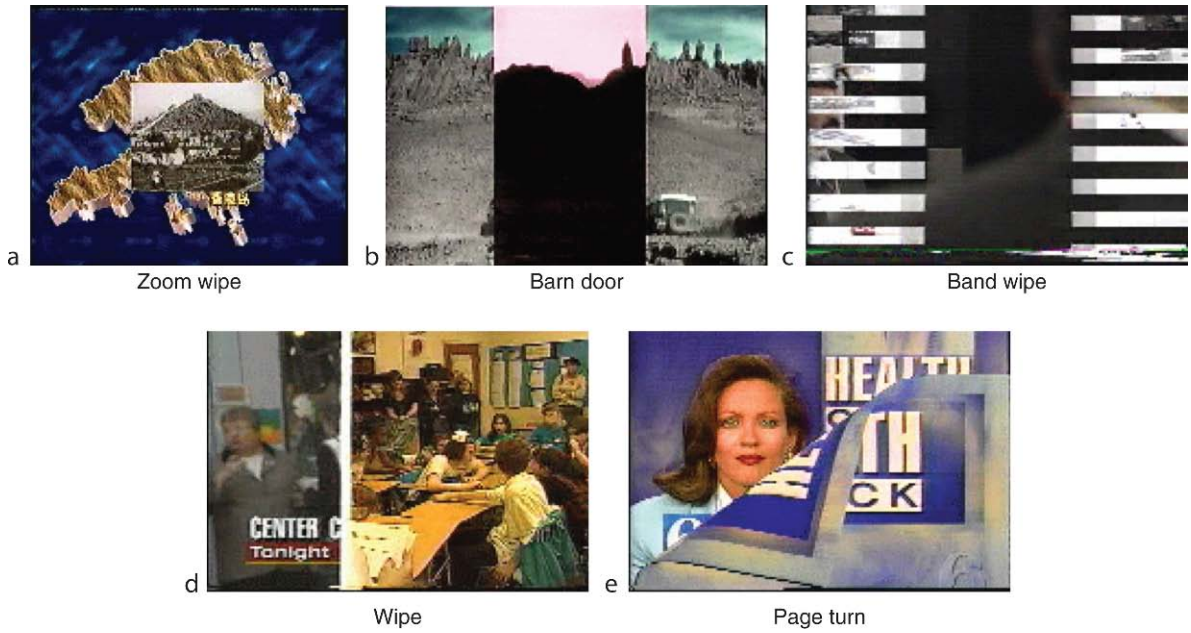
Recently, dissolve detection is also considered as a pattern classification problem, and various machine learning algorithms are brought in for this task [3]. For instance, in [6], multiple independent FSM (finite state machine) based detectors are used for detecting different types of shot boundaries. Together with classifiers, multi-resolution analysis is also adopted by [5,7,13]. In [7], dissolve detection is reduced to the cut detection problem by projecting the spatio-temporal slices to lower temporal resolution scale. The detected cuts at lower resolution are further classified in the higher resolution space by machine learning. In [13], multi-resolution features are extracted for active learning of SVM (support vector machines) to detect dissolves.

Wipe

Wipe is also a common shot boundary transition frequently found, for example, in sport and news videos. There are lots of fancy wipe transitions being used by video editors today. Figure 1 shows a few examples of wipe. As each transition exhibits its own unique transition patterns, wipe patterns, compared to dissolves, are relatively hard for machines to learn. Early works are mostly based on frame differencing and edge detection to trace the moving boundary lines created by wipes. For instance, the works in [8] detect the edge patterns formed by wipes in spatial-temporal slices. The recent work in [4] analyzes the independence and completeness properties of wipe transitions. A cost function utilizing the MPEG motion vectors is derived to model these two properties for reliable wipe detection.

General Challenges

As a fundamental building block to support other video applications, shot detection has been extensively studied in the literature. The challenges that are not completely solved for cut detection include abrupt illumination change (e.g., flashlight) and large object/camera movement, which could trigger abrupt changes of video signal and generate false alarms. For gradual transition (GT), on the other hand, the



Video Shot Detection. Figure 1. Various wipe transition.

major difficulties are the slow evolution of signal that is hard to observe. Furthermore, the temporal duration can vary greatly while the patterns could appear similar to slow camera/object movement, making GT detection a very challenging problem. For both cut and GT detection, a fundamental issue is the thresholding heuristic – mostly relying on empirical setting. Over the years, the thresholding techniques have also been evolved from simple global threshold setting, to the adaptive thresholding [10] and learning-based heuristics [3,5,13].

Key Applications

Shot detection is a fundamental task of video content manipulation. Shots are often served as the basic unit for video browsing, indexing and search. An interesting application of shot detection is that, by knowing the type, speed and frequency of shot transitions, the pace, mood, style and highlight of the underlying video events can be modeled for affective computing.

Experimental Results

The evaluations commonly used are recall and precision, defined as:

$$\text{Precision} = \frac{\text{Number of Transitions Correctly Detected}}{\text{Number of Transitions Detected}} \quad (5)$$

$$\text{Recall} = \frac{\text{Number of Transitions Correctly Detected}}{\text{Number of Transitionsarray}} \quad (6)$$

Precision and recall are in the interval of [0,1]. Low precision hints the frequent occurrence of false positives, while low recall indicates the frequent occurrence of false negatives. As gradual transition involves multiple frames, frame precision and recall are also used in order to evaluate the accurate localization of transitions,

$$\text{Frame Precision} = \frac{\text{Number of Frames Correctly Located in the Detected Transitions}}{\text{Number of Frames Located in the Detected Transitions}} \quad (7)$$

$$\text{Frame Recall} = \frac{\text{Number of Frames Correctly Located in the Detected Transitions}}{\text{Number of Frames in the Transitions}} \quad (8)$$

Data Sets

The popular datasets always experimented are TREC-VID benchmarks [9].

Cross-references

► Scene Boundary Detection
► Sub-Shot Detection

Recommended Reading

1. Alattar A.M. Detecting and compressing dissolve regions in video sequences with a DVI multimedia image compression algorithm. Int. Symp. Circuits Syst., 1:13–16, 1993.
2. Bouthemy P., Gelgon M., and Ganansia F. A unified approach to shot change detection and camera motion characterization, IEEE Trans. Circuits Syst. Video Tech., 9(7):1030–1044, 1999.
3. Hanjalic A. Shot boundary detection: unraveled and resolved. IEEE Trans. Circuits Syst. Video Tech., 12(2):90–105, February 2002.
4. Li S. and Lee M.-C. Effective detection of various wipe transitions. IEEE Trans. Circuits Syst. Video Tech., 17(6), June 2007.
5. Lienhart R. Reliable dissolve detection. In Proc. SPIE Storage Retrieval Media Database, 2001.
6. Liu Z., Gibbon D., Zavesky E., Shahrary B., and Haffner P. AT&T Research at TRECVID 2006. In Proc. TREC Video Retrieval Evaluation, 2006.
7. Ngo C.W. A robust dissolve detector by support vector machine. In Proc. 11th ACM Int. Conf. on Multimedia, 2003.
8. Ngo C.W., Pong T.C., and Chin R.T. Video partitioning by temporal slice coherency. IEEE Trans. Circuits Syst. Video Tech., 11(8):941–953, August 2001.
9. TREC-Video, http://www-nlpir.nist.gov/projects/trecvid/.
10. Vasconcelos N. and Lippman A. Statistical models of video structure for content analysis and characterization. IEEE Trans. Image Process., 9(1):3–19, January 2000.
11. Yeo B.L. and Liu B. Rapid scene analysis on compressed video. IEEE Trans. Circuits Syst. Video Tech., 5(6):533–544, 1995.
12. Yu H. and Wolf W. A multi-resolution video segmentation scheme for wipe transition identification. In Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing, 1998.
13. Yuan J., Wang H., Xiao L., Zheng W., Li J., Lin F., and Zhang B. A formal study of shot boundary detection. IEEE Trans. Circuits Syst. Video Tech., 17(2), 2007.
14. Zabih R., Miller J. Mai K., and Zabih R. et al., A feature-based algorithm for detecting and classifying scene break. In Proc. 3rd ACM Int. Conf. on Multimedia, 1995.
15. Zhang H.J., Kankanhalli A., Smoliar S., and Zhang H.J. et al. Automatic partitioning of full-motion video. ACM Multimedia Syst., 1(1):10–28, 1993.

Video Shot-Cut Detection

► Video Segmentation

Video Skimming

► Video Summarization

Video Structure Analysis

► Video Content Structure

Video Structuring

► Video Content Structure

Video Summarization

Chong-Wah Ngo, Feng Wang
City University of Hong Kong, Hong Kong, China

Synonyms

Video abstraction; Video skimming

Definition

Video summarization is to generate a short summary of the content of a longer video document by selecting and presenting the most informative or interesting materials for potential users. The output summary is usually composed of a set of keyframes or video clips extracted from the original video with some editing process. The aim of video summarization is to speed up browsing of a large collection of video data, and achieve efficient access and representation of the video content. By watching the summary, users can make quick decisions on the usefulness of the video. Dependent on applications and target users, the evaluation of summary often involves usability studies to measure the content informativeness and quality of a summary.

Historical Background

Due to the advance of web technologies and the popularity of video capture devices in the past few decades, the amount of video data is dramatically increasing. This creates a strong demand for efficient tools to

browse, access, and manipulate large video collections. The most straightforward and simplest way for quick browsing of video content is to speed up the play of video by frame dropping or re-encoding. As studied in [6], the entire video could be watched in a shorter amount of time by fast playback with almost no pitch distortion using the time compression technology. This kind of approach is easy to implement and preserves most information while shortening the time of watching videos. However, without comprehensive understanding of the original video content, the selection of video materials is impossible, and thus almost all redundant information still has to be watched. This limits further shortening of summary duration and its enjoyability. Meanwhile, the ratio of time compression is also limited.

Since the 1990s, *video content analysis* has attracted a lot of research attention. Different approaches have been proposed for automatic structuring and understanding of video content. This enables video summarization by measuring and selecting the most informative materials based on more comprehension of the video content. Video summarization can be seen as a general term of video abstraction and video skimming. In some works, they are distinguished by little difference in the composition of the output. For instance, video skims are mainly composed of the short clips skimmed from the original video. Video summary and abstraction could be a set of keyframes, clips or even other media (e.g., texts) created to describe the video content. Nowadays, video summarization has become the key tool to facilitate efficient browsing and indexing of large video collections.

Foundations

In the past two decades, a lot of research efforts have been devoted to video summarization. A systematic review of these works can be found in [10]. According to the definition, a summary should be as short or concise as possible to support efficient video browsing. On the other hand, it should be informative so that as much interesting and useful materials as possible will be included. Thus, the key of video summarization is how to measure the importance of different video clips and select the most useful ones so as to achieve both elegancy and informativeness. For this purpose, there are mainly two issues concerned: *video content analysis* and summary generation by selecting appropriate materials. *Video content analysis* enables the understanding and comprehension of the original video content. Different features are then extracted to structure and describe the video. Based on *video content analysis*, a shorter summary is generated by selecting and presenting the most important materials to users. Meanwhile, to cope with the characteristics of different video domains, many algorithms are proposed to achieve better results for the summarization of specific video domains by employing domain-specific knowledge.

Features for Content Analysis

Along with the advances in *video content analysis*, more and more features are employed in video summarization. According to the features adopted for video content analysis, the algorithms can be categorized as shot detection based [15], motion based [3], audio based [9], and multi-modality based [4,11]. In recent years, most algorithms tend to integrate multiple features to better describe the video content. Besides the visual-audio features, especially with the dramatic increasing of web videos, the metadata, context, social, and log information are being explored for video summarization of video corpus, e.g., the video clips that are watched by the most users may be more important and interesting. What features to use is usually dependent on the requirements of specific applications.

Importance Measure for Summary Generation

How to decide the importance and usefulness of video segments are essential for summary generation. During the past two decades, different approaches have been proposed. The first kind of approach is based on *video shot detection*. Shot detection is one of the fundamental techniques for video structuring. In each segmented shot, one or several keyframes are selected and presented for users to effiently browse and index the video content in order to find the interesting video segments for further exploration. In [15], several keyframes are extracted by detecting visual changes using color histogram inside each shot. In [2], to eliminate the possible redundancy among extracted keyframes, fuzzy clustering and data pruning methods are employed based on color information to obtain a set of non-redundant keyframes as the summary of the video. These approaches just employ limited low-level visual features. For the selection of keyframes or video segments, the main idea is to reduce the redundancy and maximize the visual difference in the generated

summary to show more information measure by low-level features.

In the second kind of approach, importance measure is based on some pre-defined rules. In [8], a set of rules are defined to select the most useful frames or video segments. For instance, images between similar scenes that are less than 5 seconds apart, are used for summarization; short successive shots are considered to introduce a more important topic. With prioritized video frame from each shot, another set of high order rules are then used to compose the selected clips for summary generation. For instance, the duration of each clip is at least 60 frames based on empirical and user studies of visual comprehension in short video sequences. Similar rules can also be used in query-based or domain-specific video summarization. For instance, given a soccer video, the users may be just interested in the shots of goal. In this case, to make a summary, the events of goal should be first detected and then shown to the users. In this kind rule-based approaches, the importance of video segment is manually assigned by the predefined rules. This may be useful for some specific video domains and applications. However, the rules are usually subjective. Considering the variations of video content, to make desirable summaries, more and more detailed rules are required to cope with different scenarios. This becomes almost impossible in general videos.

Today many approaches try to numerically measure the importance of video segments and model video summarization in a computational way. In [4], a user attention model utilizing a set of audio-visual features is proposed. Attention is a neurobiological conception. It implies the concentration of mental powers upon an object by close or careful observing or listening, which is the ability or power to concentrate mentally. When watching a video, human attention is always attracted by different information elements such as face, text, object and camera motion, and so on. In [4], attention models are first computed based on different visual/audio features. A user attention curve is then generated by linear combination fusion scheme. The crests of the curve are more likely to attract the viewers' attentions. Keyframes or video skims are extracted from these crests. Similarly, in [14], a perception curve that corresponds to human perception changes is constructed based on a number of visual features, including motion, contrast, special scenes, and statistical rhythm. The frames corresponding to the peak points of the perception curve are extracted for summarization. This kind of approach usually integrates multiple modalities from video documents and human perception cue to depict the curve of video importance numerically for summarization. However, there are still manually defined rules used such as the selection and the weights of different modalities.

Domain-Specific Video Summarization

According to the target video domains, video summarization can be categorized into general videos and domain-specific videos, such as music video, news video, sports video, and rushes video.

General video: The algorithms for general video summarization employ features available in all types of videos, and usually do not consider any domain-specific knowledge. In [5], a video is represented as a complete undirected graph and the normalized cut algorithm is employed to partition the graph into video clusters. The resulting clusters form a directed temporal graph and a shortest path algorithm is proposed to detect video scenes. The attention values are computed and attached to the scenes, clusters, shots and subshots. Thus, the temporal graph can describe the evolution and perceptual importance of a video. A video summary is generated from the temporal graph to emphasize both content balance and perceptual quality. The advantage of this kind of approach is that no prior knowledge is used and the algorithms can work for most videos. However, when watching videos of different domains, the users' attentions are quite different. By treating all videos in the same way, these algorithms usually cannot produce satisfactory results for videos of specific interest. Thus, a lot of works focus on specific video domains and make use of domain-specific knowledge in order to improve the summary qualities.

Music video: In music video, the music plays the dominant role. Thus, music video summarization is mainly based on music analysis. In [13], the chorus is detected and used as a thumbnail for music content. The analysis of visual content such as shot classification and text (lyrics) recognition are employed for music-visual-text alignment so as to make a meaningful and smooth music video summary.

Sports video: When watching sports videos, people are more interested in the exciting moments and great plays. Thus, the highlight detection and event

classification are usually involved in order to make a good summary. In [1], a multi-level representation for sports video is proposed. Based on low-level features, different semantic shots (e.g., Audience, Player Close-up, and Player Following) are classified at mid-level. For different sports games, the corresponding interesting objects and events (such as the football and a goal event in a soccer video) are detected at high-level. The interesting events can then be selected for summary generation.

News video: In daily news videos, a topic is usually composed of a story chain during the topic evolution. The dependencies among related stories are important to summarize a news topic. A feasible solution for rapid browsing of news topics is by threading and autodocumenting news videos [12]. In [12], the duality between stories and textual-visual concepts is first exploited to cluster the stories of the same topic. The topic structure is then presented by exploring the textual-visual novelty and redundancy of stories. By pruning the peripheral and redundant news tories in the topic structure, a main thread is extracted for autodocumentary.

Rushes video: Rushes are the raw materials captured during video production. Being unedited, rushes contain a lot of redundant and junk information which is intertwined with useful stock footage. In [11], stock footage is first located by motion analysis, repetitive shot detection, and shot classification. The most representative video clips are then selected to compose a summary based on object and event understanding.

Key Applications

Video summarization can be used in many applications.

Multimedia Archives Indexing and Retrieval

Nowadays more and more video documents are being digitized and archived worldwide. Video summarization can be used to index and retrieve a large video collection, and thus facilitate efficient access of video content. For instance, the online summarization can support journalists when searching old video material, or when producing documentaries. Another example is the wide use of web videos, where the index could easily be realized by automatically generated video summaries.

Movie Marketing

In the broadcasting and filmmaking industries, a lot of raw footage (extra video, B-rolls footage) is used to generate the final products such as TV programs and movies. Twenty to forty times as much material may be shot as actually becomes part of the finished product. The "shoot-to-show" ratio, such as in BBC TV, ranges from 20 to 40. The producers see these large amount of raw footage as cheap gold mine. Video summarization can be used to find the stock footage from the heavily redundant materials for potential reuse.

Home Entertainment

For instance, one can easily have a brief overview of what happened in a television series that may have been missed.

Future Directions

Performance evaluation

Despite the advance achieved in video summarization, a fundamental problem, i.e., performance evaluation, is yet to be solved. Today, the evaluation is mainly carried out by subjective scoring based on the users' personal judgment or few predefined criterions. This limits effective comparison between different approaches.

Comprehensive understanding and modeling of video content

In the existing approaches, neither the rule-based nor computational models can fully grasp the content and the storyline of the video. Besides more powerful approaches for *video content analysis*, the modeling of multimedia information is also essential for video summarization.

From large-scale video summarization to search and browsing

The existing works mainly focus on the summarization of a single video document or a small video cluster, where the information inside a video or dependency between videos can be relatively easily modeled. With the dramatic increasing of video data (e.g., web videos), how to summarize a huge group of videos, for instance, by structuring the relationship among the diversity of videos is a big challenge. Besides video content dependency, context and social information are cues that could be explored for this type of summarization in large-scale. In addition, the exploitation of summarization for large-scale video search and browsing remains an area yet to be studied.

Experimental Results

In TRECVID Workshop 2007 on Rushes Summarization [7], there were 22 runs submitted for evaluation on the same benchmark. Different criteria such as IN (inclusion of groundtruth objects/events), EA (easy to understand), and RE (redundancy of the summary) were assessed. Runs generated by CityUHK, NII, and LIP6 were verified by [7] as significantly better than two baselines from CMU. The baselines were generated by evenly sampling frames and shot clustering, respectively.

Data Sets

Rushes Videos by TRECVID Workshop.
Since 2004, the annual TRECVID Workshop provides a benchmark for rushes video exploitation and summarization. The dataset is composed of 100 h's raw video materials produced during news video and film production.

Cross-references

► Video Content Analysis

References

1. Duan L.Y., Xu M., Chua T.S., Tian Q., and Xu C. A Mid-Level Representation Framework for Semantic Sports Video Analysis. In Proc. 11th ACM Int. Conf. on Multimedia, 2003.
2. Ferman A.M. and Tekalp A.M. Two-stage hierarchical video summary extraction to match low-level user browsing preerences. IEEE Trans. Multimedia, 5(2):244–256, 2003.
3. Liu T., Zhang H.J., and Qi F. A Novel Video Keyframe Extraction Algorithm based on Perceived Motion Energy Model. IEEE Trans. Circuits Syst. Video Tech., 13(10):1006–1013, 2003.
4. Ma Y.F., Lu L., Zhang H.J., and Li M. A user attention model for video summarization. In Proc. 10th ACM Int. Conf. on Multimedia, 2002.
5. Ngo C.W., Ma Y.F., and Zhang H.J. Video summarization and scene detection by graph modeling. IEEE Trans. Circuits Syst. Video Tech., 15(2):296–305, 2005.
6. Omoigui N., He L., Gupta A., Grudin J., and Sanocki E. Time-compression: system concerns, usage, and benefits. In Proc. SIGCHI Conf. on Human Factors in Computing Systems, 1999.
7. Over P., Smeaton A.F., and Kelly P. The TRECVID 2007 BBC Rushes Summarization Evaluation Pilot. In TRECVID BBC Rushes Summarization Workshop in ACM Multimedia, 2007.
8. Smith M.A. and Kanade T. Video skimming and characterization through the combination of image and language understanding. In Proc. IEEE Int. Workshop on Content-Based Access of Image and Video Database, 1998.
9. Taskiran C.M., Pizlo Z., Amir A., Ponceleon D., and Delp E. Automated video program summarization using speech transcripts. IEEE Trans. Multimedia. 8(4):775–791, 2006.
10. Truong B.T. and Venkatesh S. Video abstraction: a systematic review and classification. ACM Trans. Multimedia Comput. Commun. Appl., 3(1), 2007.
11. Wang F. and Ngo C.W. Rushes video summarization by object and event understanding. In TRECVID Workshop on Rushes Summarization in ACM Multimedia Conference September 2007.
12. Wu X., Ngo C.W., and Li Q. Threading and autodocumenting in news videos. IEEE Signal Process. mag., 23(2):59–68, 2006.
13. Xu C., Shao X., Maddage N.C., and Kankanhalli M.S. Automatic music video summarization based on audio-visual-text analysis and alignment. In Proc. 31st Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 2005, pp. 361–368.
14. You J., Liu G., Sun L., and Li H. A multiple visual models based perceptive analysis framework for multilevel video summarization. IEEE Trans. Circuits Syst. Video Tech., 17(3), 2007.
15. Zhang H.J., Wu J., Zhong D., and Smoliar S.W. An integrated system for content-based video retrieval and browsing. Pattern Recogn., 30(4):643–658, 1997.

View Adaptation

Kenneth A. Ross
Columbia University, New York, NY, USA

Synonyms

Materialized view redefinition

Definition

Small changes to the definition of a materialized view are often needed in database systems. View adaptation is the process of keeping a materialized view up-to-date when the definition of the view is changed. View adaptation aims to leverage the previously materialized view to generate the new view, since the cost of rebuilding the materialized view from scratch may be expensive.

Key Points

View adaptation is related to the question of *answering queries using views*. The new view can be thought of as a query, with the old view available to help compute it. However, view adaptation also admits in-place changes that are not possible using a query-answering approach. For example, if a redefined view contains most but not all of the records from the original view, then view adaptation could be achieved by deleting the records that no longer qualify.

A variety of adaptation techniques are presented in [2,3], allowing changes to the SELECT, FROM,

WHERE, GROUPBY and HAVING clauses. UNION and EXCEPT views are also considered. In some cases, multiple changes to a view definition can be handled in a single pass, without materializing intermediate results for each change. Experimental results show the value of adaptation, particularly the in-place methods, relative to recomputation. Extensions of view adaptation to more general contexts, such as distributed databases, have also been proposed [1,4].

Data warehouses and decision support systems often employ materialized views to speed up query processing. Applications in which a user can change queries dynamically and see the results fast, such as data visualization, data archeology, and dynamic query processing can also benefit from view adaptation.

Cross-references

- ▶ Answering Queries using Views
- ▶ Incremental View Maintenance
- ▶ Materialized Views

Recommended Reading

1. Bellahsene Z. View adaptation in the fragment-based approach. IEEE Trans. Knowl. Data Eng., 16(11):1441–1455, 2004.
2. Gupta A., Mumick I.S., Rao J., and Ross K.A. Adapting materialized views after redefinitions: techniques and a performance study. Inf. Syst., 26(5):323–362, 2001.
3. Gupta A., Mumick I.S., and Ross K.A. Adapting materialized views after redefinitions. In Proc. SIGMOD Conf. on Management of Data. San Jose, CA, 1995, pp. 211–222.
4. Mohania M.K. and Dong G. Algorithms for adapting materialized views in data warehouses. In Proc. Int. Symp. on Cooperative Database Systems for Advanced Applications, 1996, pp. 309–316.

View Definition

Yannis Kotidis
Athens University of Economics and Business, Athens, Greece

Synonyms

View expression

Definition

The definition of a view consists of the name of the view and of a query, whose result is used to determine the content of the view.

Key Points

A view is a virtual relation. Its content depends on the evaluation of a query over a set of base tables or other views in the database. This query is part of the view definition and is, typically, recomputed every time the view is referenced. In some cases, for efficiency, the tuples of a view may be materialized as a separate table in the database.

In relational systems, a view is defined using the **create view** command:

create view $<$ v $>$ **as** $<$ *query expression* $>$

The name of the view in the above example is $<$ v $>$ and the schema and content of the view are derived on-demand by the evaluation of $<$ query expression $>$, which should be a legal expression supported by the database management system. Different vendor systems may impose some constraints on the form of $<$ query expression $>$, for instance they may disallow references to temporary tables. When the data in the table(s) mentioned in $<$ query expression $>$ changes, the data in view $<$ v $>$ changes also.

Consider the following view definitions:

create view v1 as select Name, Age from Personnel where Department = "Sales"
create view v2 as select Wages.Name,Wages.Salary from Personnel, Wages where Personnel.Name = Wages.Name and Personnel.Department = "Sales"

The first expression defines a view termed *v1* that contains the name and age attributes from database table *Personnel*. The instance of view *v1* consists of the subset of personnel data restricted to those working at department *Sales*. View *v2* defined by the second expression, contains the result of the join between tables *Personnel* and *Wages* for employees working at the same department.

Cross-references

- ▶ View Maintenance Aspects
- ▶ View Unfolding
- ▶ Views

Recommended Reading

1. Adiba M.E. and Lindsay B.G. Database snapshots. In Proc. 6th Int. Conf. on Very Data Bases, 1980, pp. 86–91.

2. Dayal U. and Bernstein P. On the correct translation of update operations on relational views. ACM Trans. Database Syst., 8(3):381–416, 1982.
3. Gupta A., Jagadish H.V., and Mumick I.S. Data integration using self-maintainable views. In Advances in Database Technology, Proc. 5th Int. Conf. on Extending Database Technology, 1996, pp. 140–144.
4. Gupta H., Harinarayan V., Rajaraman A., and Jeffrey D.U. Index selection for OLAP. In Proc. 13th Int. Conf. on Data Engineering, 1997, pp. 208–219.
5. Kotidis Y. and Roussopoulos N. DynaMat: a dynamic view management system for data warehouses. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1999, pp. 371–382.
6. Roussopoulos N. View indexing in relational databases. ACM Trans. Database Syst., 7(2):258–290, 1982.
7. Roussopoulos N. An incremental access method for viewCache: concept, algorithms, and cost analysis. ACM Trans. Database Syst., 16(3):535–563, 1991.

View Maintenance

Alexandros Labrinidis[1], Yannis Sismanis[2]
[1]Department of Computer Science, University of Pittsburgh, Pittsburgh, PA, USA
[2]IBM Almaden Research Center, Almaden, CA, USA

Synonyms

View update; Materialized view maintenance

Definition

View maintenance typically refers to the updating of a *materialized view* (also known as a derived relation) to make it consistent with the base relations it is derived from. Such an update typically happens *immediately*, with the transaction that updates the base relations also updating the materialized views. However, such immediate updates impose significant overheads on update transactions that cannot be tolerated by many applications. *Deferred view maintenance*, on the other hand, allows the view to become inconsistent with its definition, and a *refresh operation* is required to establish consistency. Typically, under deferred maintenance, a view is *incrementally* updated only just before data is retrieved from it (i.e., on-demand, just before a query is performed on the view).

Historical Background

Early systems that supported views did so in their "pure form," i.e., by storing just the view definition and using query rewriting to take advantage of views in other queries [11].

Incremental view maintenance is introduced in [1] through a technique to efficiently detect relevant updates to materialized views, thus streamlining their maintenance.

Deferred view maintenance is introduced in [10] as a scheme for materializing copies of views on workstations attached to a mainframe that maintains a shared global database. The workstations update local copies of the views while processing queries. In [7], deferred view maintenance is defined as the application of incremental view maintenance whenever desired, unlike the immediate view maintenance, where any database update triggers the incremental view maintenance algorithm. [5] has a nice survey of view maintenance techniques.

Foundations

Algorithms and techniques for maintenance of materialized views can be classified according to three different criteria:

- Whether the view is recomputed from scratch or not: *recomputation* versus *incremental* maintenance.
- Whether the view is updated whenever the base data change or not: *immediate* versus *deferred* maintenance.
- Whether queries can be executed while the view is being updated or not: *online* versus *offline* maintenance.

All the above dimensions are typically orthogonal. We explain the different options below.

View Recomputation

Recomputing a materialized view from the base relations it is derived from is the most general technique of updating. As such, it can be applied on any type of view, regardless of the complexity of the query definition. The disadvantage is that, in most cases, such recomputation is costly, and, in many cases, the view can be updated *incrementally* instead, at a fraction of the cost.

Incremental View Maintenance

It is possible to update a materialized view *incrementally* for many types of view definitions (i.e., queries). One such class is the general case of SPJ views (i.e., views whose definition is just a select-project-join query).

For example, assume that we have a view V defined over two relations R and S through a natural join (i.e., $V = R \bowtie S$; for simplicity of the presentation we ignore the selection and projection operators). Further, let us assume that we have a set of deleted tuples from relation R, denoted as R_D; a set of inserted tuples into relation R, denoted as R_I (i.e., $R' = R \cup R_I - R_D$). Also, assume a set of deleted tuples from relation S, denoted as S_D; and a set of inserted tuples into relation S, denoted as S_I (i.e., $S' = S \cup S_I - S_D$). We trivially represent base relation updates as pairs of deletions and insertions.

Given the above, the updated version of V, i.e., V', should be $V' = R' \bowtie S' = (R \cup R_I - R_D) \bowtie (S \cup S_I - S_D)$. By expanding this further, and grouping all the deletions from V as V_D and all the insertions to V as V_I, we have that: $V_D = (R_D \bowtie (S \cup S_I)) \bowtie ((R \cup R_I) \bowtie S_D)$, and $V_I = (R_I \bowtie S) \cup (R \bowtie S_I) \cup (R_I \bowtie S_I)$, so that $V' = V \cup V_I - V_D$. This, incrementally computed formula, should be less costly to compute than recomputing the entire join from scratch.

The problem of incrementally updating materialized views is difficult in the general case, but there are additional classes of queries (i.e., besides SPJ views) that it can be solved for [6].

Immediate View Maintenance

The default way of updating materialized views is to do so *immediately*, i.e., batch together, in a single transaction, the updating of the base relations and the updating of the materialized views that are derived from these relations. However, many applications cannot tolerate this delay, especially if they are interactive and users are expecting an answer at transaction commit.

Deferred View Maintenance

Incremental deferred view maintenance requires (i) techniques for checking what views are affected by an update to the basic tables, (ii)auxiliary tables that maintain certain information like updates and deletes since the last view refresh and finally (iii) techniques for propagating the changes from the base tuples to the view tuples without fully recomputing the view relation.

First, Buneman in [2], proposes a technique for the efficient implementation of alerters and triggers that checks each update operation prior to execution to see whether it can cause a view to change. In [1], an efficient method for identifying updates that cannot possibly affect the views is described. Such *irrelevant updates* are then removed from consideration while differentially updating the views.

In [7], the hypothetical relations technique developed in [12] is adapted to the purpose of storing and indexing the deltas to the base tables. The main idea is to use a single table AD that stores deletions and insertions for the base tables (updates can be modeled as a deletion followed by an update). Whenever a view is accessed, the base tables and the AD table need to be accessed (in order to check for new or deleted tuples). A bloom filter however, is used to check if a tuple from the base relation exists in AD significantly reducing irrelevant accesses to AD.

In [4], the authors demonstrate that the ordering of the updates from the base tuples to the view tuples is critical and call this phenomenon *state bug*. Typically, an "incremental query" – during the refresh operation – avoids recomputing the full view and only incrementally computes the delta view to bring it up to date, based on updates/deletes made to the base tables. Such incremental queries can evaluated in two states: The *pre-update state*, where the base table updates have not been applied yet or the *post-update state* where changes have been applied. In most techniques a pre-update state is assumed which severely limits the class of updates and views considered. The post-update state allows for a much larger class of view to be deferred maintained, however direct application of pre-update techniques results in incorrect answers (state bug) and new techniques are proposed.

Offline View Maintenance

Typically, maintaining materialized views is done *offline*, without allowing queries to the materialized view to execute concurrently with the processing of the materialized view updates. This simplifies the view maintenance algorithms significantly, at the expense of delaying queries. Traditionally, in data warehousing environments [3], updates of materialized views are performed at night, thus minimizing the possibility of delaying user queries.

Online View Maintenance

The need of most companies for continuous operation (especially in the presence of the Web), has precipitated the need for *online view maintenance*, where

queries can be answered while the materialized views are being updated.

In a centralized setting, this is typically achieved through some sort of multi-versioning, either as *horizontal redundancy*, where extra columns are added to hold the different versions [9], or as *vertical redundancy*, where extra rows are needed to hold the different versions [8]. In a distributed setting, this is typically achieved through determination of additional queries to ask of the data sources [13].

Key Applications

Materialized views help speed up the execution of frequently accessed queries, giving interactive response times to even the most complex queries. The cost of maintaining materialized views is typically amortized over multiple accesses (i.e., queries to the view). This has been utilized/transferred in many different application domains, from data warehousing to web data management. Beyond efficient algorithms and techniques to update materialized views, special attention has also been given to the *view selection* problem: how to identify which views should be materialized, and also to the issue of how to effectively use materialized views to answer other queries (i.e., by utilizing subsumption or caching).

Cross-references

► Recursive View Maintenance
► View Selection

Recommended Reading

1. Blakeley J.A., Larson P.Å., and Tompa F.W. Efficiently Updating Materialized Views. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1986, pp. 61–71.
2. Buneman P. and Clemons E.K. Efficient Monitoring Relational Databases. ACM Trans. Database Syst., 4(3):368–382, 1979.
3. Chaudhuri S. and Dayal U. An overview of data warehousing and OLAP technology. ACM SIGMOD Rec., 26(1):65–74, 1997.
4. Colby L.S., Griffin T., Libkin L., Mumick I.S., and Trickey H. Algorithms for deferred view maintenance. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1996, pp. 469–480.
5. Gupta A. and Mumick I.S. Maintenance of materialized views: problems, techniques, and applications. IEEE Data Eng. Bull., 18(2):3–18, 1995.
6. Gupta A., Mumick I.S., and Subrahmanian V.S. Maintaining views incrementally. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1993, pp. 157–166.
7. Hanson E.N. A performance analysis of view materialization strategies. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1987, pp. 440–453.
8. Labrinidis A. and Roussopoulos N. A performance evaluation of online warehouse update algorithms. Tech. Rep. CS-TR-3954, Department of Computer Science, University of Maryland, 1998.
9. Quass D. and Widom J. On-line warehouse view maintenance. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1997, pp. 393–404.
10. Roussopoulos N. and Kang H. Principles and Techniques in the Design of ADMS±. IEEE Comp., 19(12):19–25, 1986.
11. Stonebraker M. Implementation of integrity constraints and views by query modification. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1975, pp. 65–78.
12. Woodfill J. and Stonebraker M. An implementation of hypothetical relations. In Proc. 9th Int. Conf. on Very Data Bases, 1983, pp. 157–166.
13. Zhuge Y., Garcia-Molina H., Hammer J., and Widom J. View maintenance in a warehousing environment. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1995, pp. 316–327.

View Maintenance Aspects

Antonios Deligiannakis
University of Athens, Athens, Greece

Definition

Database systems often define *views* in order to provide conceptual subsets of the data to different users. Each view may be very complex and require joining information from multiple *base relations*, or other views. A view can simply be used as a query modification mechanism, where user queries referring to a particular view are appropriately modified based on the definition of the view. However, in applications where fast response times to user queries are essential, views are often materialized by storing their tuples inside the database. This is extremely useful when recomputing the view from the base relations is very expensive. When changes occur to their base relations, materialized views need to be updated, with a process known as view maintenance, in order to provide fresh data to the user.

Historical Background

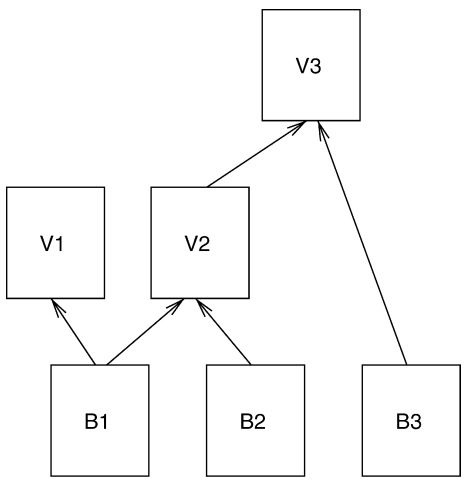
The use of relational views has long been proposed in relational database systems. The notion of materialized views, or *snapshots*, was first proposed in [1]. A snapshot represents the state of some portion of the database at the time when the snapshot was computed. Since the publication of [1], a large number of mechanisms for refreshing materialized views has been proposed. These

mechanisms mainly involve either the time of refreshing the materialized views, or how such a refresh operation can be performed (i.e., through a re-computation of the view, or in an incremental manner). While each policy has its own advantages and drawbacks, ADMS was the first system that realized the importance of having multiple maintenance policies within the same database [11,12]. An excellent classification of the view maintenance problem was presented in [4].

Foundations

Materialized views are used in order to provide faster response times to user queries. Figure 1 depicts a sample *view-dependency graph* of three materialized views (namely, V1, V2 and V3) defined over three base relations (denoted as B1, B2 and B3). If a view V contains in its definition a reference to a base relation (or another view) B, then there exists a directed edge in the view-dependency graph from B to V. Given this explanation, please note that V3 in the sample view-dependency graph is expressed in terms of a base relation (B3) and another materialized view (V2). However, the ultimate set of base relations that are used to derive V3 are B1, B2 and B3.

When the underlying data of the base relations changes, the materialized views contain stale data, until the materialized view is refreshed. The view maintenance can be performed either through a re-computation of the view or, if possible, through an incremental procedure. An incremental update policy is often faster, and is thus in many cases desirable, if the base relations have been modified only by a small portion. However, when a large portion of some base relations is updated (i.e., most tuples of a base relation are deleted), then a complete re-computation may actually be faster. The process of maintaining a materialized view can be classified based on five main dimensions.



View Maintenance Aspects. Figure 1. An example of a view-dependency graph.

Time Dimension

Every update transaction incurs a time penalty, thus slowing down queries performed during the update process. On the other hand, some applications (i.e., applications regarding stock data) cannot tolerate viewing stale data.

Depending on when a materialized view is updated, three main policies for view maintenance can be defined [2]:

1. Immediate Views: Upon an update to a base relation, the materialized view is updated immediately. This is often accomplished using database triggers.
2. Deferred Views: Updating the materialized view is deferred until the first time when the view is queried.
3. Snapshot Views: The view is updated periodically (i.e., at daily intervals) by an asynchronous process.

Each of the aforementioned policies has some advantages and drawbacks, and each may be the policy of choice, depending on the characteristics of the targeted application. Immediate views incur an update penalty even if the updated tuples are never queried. Deferred views incur this update cost only at the first time when the view is queried after its base relations have been updated. Thus, deferred views incur a lower update overhead, but also worse query performance, since the query evaluation process may have to wait for some views to be updated, when compared to the immediate views policy. The snapshot views policy leads to faster query performance, since typically no updates are performed in parallel to the user queries, and to better update performance, since the updates are batched. Thus, snapshot views represent an attractive choice when the application may tolerate reading stale data.

Expressiveness of View Definition Language

Algorithms proposed for the maintenance of a materialized view can often handle only views expressed as a subset of SQL. For example, some techniques may only be able to handle views defined as select-project-join (SPJ) queries. However, a view definition may be much more complex, as it may contain (amongst others)

duplicates, arithmetic operations, aggregate or ranking functions, nested subqueries, set operations such as calculating the union or difference amongst two sets, outer-joins, recursion etc.

View maintenance algorithms often seek to derive formulas for determining the update to a view, and thus avoid recalculating the entire view from scratch, based on the updates to its base relations. For example, for the sample dependency graph presented in Fig. 1, let view $V2$ be calculated as: $V2 = B1 \bowtie_{JC} B2$, where JC denotes the join condition in the definition of $V2$ between relations $B1$ and $B2$. If $\Delta B1$ and $\Delta B2$ represent the inserted (deleted) tuples to relations $B1$ and $B2$, then the inserted (deleted) tuples $\Delta V2$ to view $V2$ can be calculated as:

$$\Delta V2 = (\Delta B1 \bowtie_{JC} B2) \cup (B1 \bowtie_{JC} \Delta B2) \cup (\Delta B1 \bowtie_{JC} \Delta B2)$$

Updates can be modeled as deletions followed by insertions. Deriving such formulas for computing the incremental update operations on a view typically becomes harder, or even impossible, as the expressiveness of the view definition language is increased. Thus, maintenance algorithms often first check whether a view can be incrementally maintained, and then proceed to decide how to actually maintain it.

Available Information

When maintaining a materialized view, we do not always have access to both the materialized view and its base relations. For example, when one of the base relations represents a data stream, as in the case of Chronicle Views [7], the entire relation cannot be stored and is, thus, unavailable during the maintenance process. Thus, during the view maintenance process one may, or may not, have access to the materialized view itself, to its base relations, or to information regarding the presence of keys and referential integrity constraints. Depending on the amount of information available, different algorithms can be used for maintaining a view.

Supported Modifications

The view maintenance algorithms can also be classified based on types of modifications to the base relations that they can handle. Not all algorithms handle both insertions and deletions to base relations. Some algorithms handle sets of modifications (i.e., insertions and deletions) in a single pass, as in [6], while others require one pass for each different modification set. Some algorithms handle updates directly, while other algorithms model them as deletions followed by insertions. Finally, not all view maintenance algorithms can handle more complex modifications, such as modifications to the view definition. Even when such modifications are handled, the view maintenance can be achieved by different techniques that either recompute the view, or try to *adapt* the view, so that the old view can be used to materialize the new view [5].

Algorithm Applicability

View maintenance algorithms should be able to be applied to all possible data stored in the database tables, and to all instances of a particular modification (i.e., insertion, deletion, update etc). Techniques that operate correctly but only on specific database instances, or on specific only modification instances are less desirable.

Figure 2, originally presented in [4], summarizes some areas of the problem space for three of the five dimensions described above, namely for the expressiveness of the view definition language, for the available information and for the supported modifications dimensions. The remaining two dimensions have been omitted for ease of presentation.

Key Applications

Data Warehousing

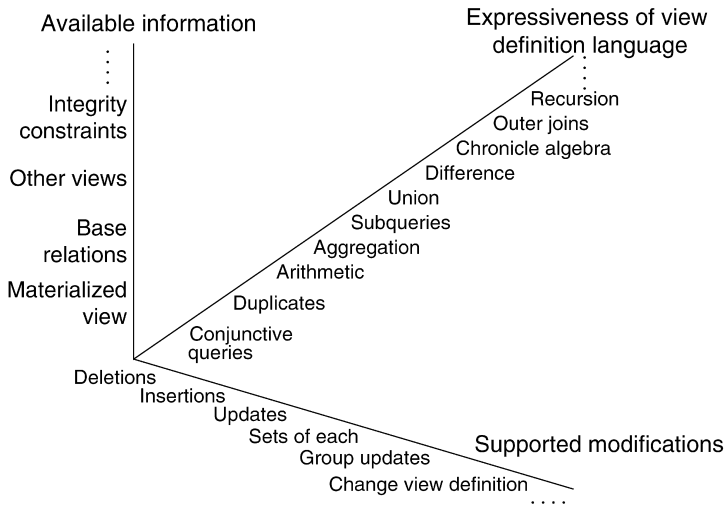
Materialized views are frequently used in data warehouses to provide personalized behavior to users, store frequently queried data derived from multiple relations, and to encapsulate different views of local or even remote data and databases. Materialization allows for significantly better query performance [8], while view maintenance, in the case of Immediate or Deferred Views, allows users to access fresh data.

Data Streams

Banking, billing, networking and stock applications often generate infinite streams of data. View maintenance algorithms can help answer complex queries over such infinite data streams without requiring access to the entire stream.

Caching

Cached data can be refreshed quickly using techniques developed for view maintenance when only a small portion of the cache has become stale. The cached



View Maintenance Aspects. Figure 2. The problem space, as presented in [4].

content may also involve dynamically generated web page fragments. In this case, the notion of WebViews [9,10] can be used to decide which such fragments to materialize, and how to determine a schedule for updating them.

Mobile Applications

If data needs to be transmitted in cases of bandwidth constraints, as in applications of mobile clients holding a cell phone or a GPS system, then only the data altered between the last transmission needs to be transmitted. Similar techniques that transmit only the changes in the computed data can also be used in other bandwidth-constrained applications as well, such as in transmitting smaller amounts of data to/from sensor nodes [3].

Cross-references

- ▶ Database Tuning and Performance
- ▶ Deferred View Maintenance
- ▶ Incremental View Maintenance
- ▶ Maintenance of Recursive Views
- ▶ Maintenance of Materialized Views with Outer-Joins
- ▶ Query Processing and Optimization in Object Relational Databases
- ▶ View Maintenance
- ▶ Viewcaches

Recommended Reading

1. Adiba B. and Lindsay B. Database snapshots. In Proc. 6th Int. Conf. on Very Data Bases, 1980, pp. 86–91.
2. Colby L., Kawaguchi A., Lieuwen D., Mumick I.S., and Ross K.A. Supporting multiple view maintenance policies. In Proc. ACM SIGMOD Conf. on Management of Data, 1997, pp. 405–416.
3. Deligiannakis A., Kotidis Y., and Roussopoulos N. Processing approximate aggregate queries in wireless sensor networks. Inf. Syst., 31(8):770–792, December 2006.
4. Gupta A. and Mumick I.S. Maintenance of materialized views: problems, techniques, and applications. IEEE Data Eng. Bull., Special Issue on Materialized Views and Data Warehousing, 18(2):3–19, June 1995.
5. Gupta A., Mumick I.S., Rao J., and Ross K.A. Adapting materialized views after redefinitions: techniques and a performance study. Inf. Syst., Special Issue on Data Warehousing, 16(5):323–362, July 2001.
6. Gupta A., Mumick I.S., and Subrahmanian V.S. Maintaining views incrementally. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1993, pp. 157–166.
7. Jagadish H.V., Mumick I.S., and Silberschatz A. View maintenance issues in the chronicle data model. In Proc. 14th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, 1995, pp. 113–124.
8. Kotidis Y. and Roussopoulos N. DynaMat: a dynamic view management system for data warehouses. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1999, pp. 371–382.
9. Labrinidis A. and Roussopoulos N. WebView materialization. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2000, pp. 367–378.
10. Labrinidis A. and Roussopoulos N. Balancing performance and data freshness in Web database servers. In Proc. 29th Int. Conf. on Very Large Data Bases, 2003, pp. 393–404.
11. Roussopoulos N. The incremental access method of view cache: concept, algorithms, and cost analysis. ACM Trans. Database Syst., 16(3):535–563, September 1991.
12. Roussopoulos N. and Kang H. Principles and techniques in the design of ADMS±. IEEE Comput., 19(2):19–25, December 1986.

View Expression

► View Definition

View Update

► View Maintenance

View-based Data Integration

Yannis Katsis, Yannis Papakonstantinou
University of California-San Diego, La Jolla, CA, USA

Definition

Data Integration (or *Information Integration*) is the problem of finding and combining data from different sources. *View-based Data Integration* is a framework that solves the data integration problem for *structured* data by integrating sources into a single unified view. This integration is facilitated by a declarative *mapping language* that allows the specification of how each source relates to the unified view. Depending on the type of view specification language used, view-based data integration systems (VDISs) are said to follow the *Global as View (GAV)*, *Local as View (LAV)* or *Global and Local as View (GLAV)* approach.

Historical Background

Data needed by an application are often provided by a multitude of data sources. The sources often employ heterogeneous data formats (e.g., text files, web pages, XML documents, relational databases), structure the data in different ways and can be accessed through different methods (e.g., web forms, database clients). This makes the task of combining information from multiple sources particularly challenging. To carry it out, one has to retrieve data from each source individually, understand how the data of the sources relate to each other and merge them, while accounting for discrepancies in the structure and the values, as well as for potential inconsistencies.

The first to realize this problem were companies willing to integrate their structured data within or across organizations. Soon the idea of integrating the data into a single unified view emerged. These systems, to be referred to as view-based integration systems (VDISs) would provide a single point of access to all underlying data sources. Users of a VDIS (or applications) would query the unified view and get back integrated results from all sources, whereas the task of combining data from the sources and resolving inconsistencies would be handled by the system transparently to the applications.

The VDISs made their first appearance in the form of multidatabases and federated systems [11]. Subsequently, the research community dived into the problem of specifying the correspondence between the sources and the unified view. The outcome were three categories of languages to express the correspondence (GAV, LAV and GLAV), together with several related theoretical and system results. The industry also embraced the view-based integration framework by creating many successful VDISs (e.g., BEA AquaLogic, IBM WebSphere).

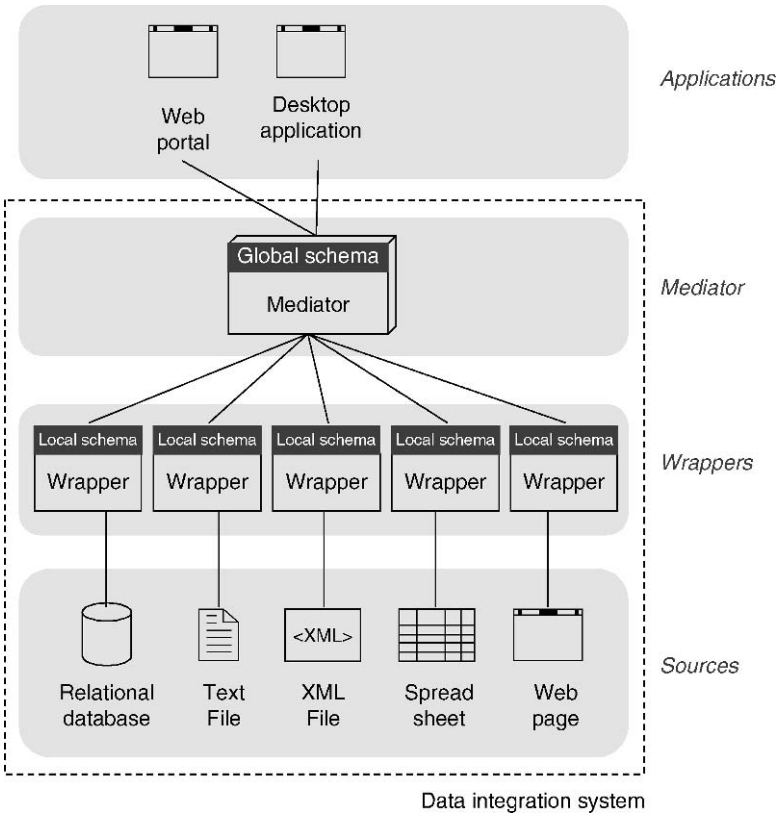
Foundations

Abstracting out the differences between individual systems, a typical VDIS conforms to the architecture shown in Fig. 1. Sources store the data in a variety of formats (relational databases, text files, etc.). Wrappers solve the heterogeneity in the formats by transforming each source's data model into a common data model used by the integration system. The wrapped data sources are usually referred to as local or source databases, the structure of which is described by corresponding local/source schemas. This is in contrast to the unified view exported by the mediator, also called global/target database. Finally, mappings expressed in a certain mapping language (depicted as lines between the wrapped sources and the mediator) specify the relationship between the wrapped data sources (i.e., the local schemas) and the unified view exported by the mediator (i.e., the global schema).

Given a VDIS, applications or users retrieve data from the sources indirectly by querying the global schema. It is the task of the mediator to consult the mappings to decide which data to retrieve from the sources, and how to combine them appropriately in order to form the answer to the user's query.

VDISs can be categorized according to the following three main axes:

1. *Common data model and query language.* The data model and query language that is exposed by the



View-based Data Integration. Figure 1. View-based Data Integration System (VDIS) architecture.

wrappers to the mediator and by the mediator to the applications. Commonly used data models include the relational, XML and object-oriented data model.

2. *Mapping language.* The language for specifying the relationship of sources with the view. Languages proposed in the literature fall into three categories; *Global as View (GAV)*, *Local as View (LAV)* and *Global and Local as View (GLAV)*. Being one of the most important components in a VDIS, these are discussed in detail below.
3. *Data storage method.* The decision on the place where the data are actually stored. The two extremes are the materialized and the virtual approach (see [14] for a comparison). In the materialized integration (also known as eager, in-advance or warehousing approach), all source data are replicated on the mediator. On the other hand, in the virtual mediation (e.g., Infomaster [6]) (or lazy approach), the data are kept in the sources and the global database is virtual. Consequently, a query against the global database cannot be answered directly, but has to be translated to queries against the actual sources. Finally, some systems employ hybrid policies, such as virtualization accompanied by a cache.

Specifying the Relationship of the Sources with the Unified View

To allow the mediator to decide which data to retrieve from each source and how to combine them into the unified view, the administrator of the VDIS has to specify the correspondence between the local schema of each source and the global schema through mappings.

The mappings are expressed in a language, corresponding to some class of logic formulas. Languages proposed in the literature fall into three categories: *Global as View (GAV)*, *Local as View (LAV)* and *Global and Local as View (GLAV)*. In GAV the global database (schema) is expressed as a function of the local databases (schemas). LAV on the other hand follows the

opposite direction, with each local schema being described as a function over the global schema. Therefore, LAV allows to add a source to the system independently of other sources. Finally, GLAV is a generalization of the two. This section presents each of these approaches in detail and explains their implications on the query answering algorithms. Essentially, each of them represents a different trade-off between expressivity and hardness in query answering.

Running example. For ease of exposition, the following discussion employs a running example of integrating information about books. The example employs the relational data model for both the sources and the global database. Moreover, the query language used by the users to extract information from the global database, and in turn by the mediator to retrieve information from the sources, is the language of conjunctive queries with equalities ($CQ^{=}$); a subset of SQL widely adopted in database research.

Figure 2 shows the employed local and global schemas. Relations are depicted in italics and their attributes appear nested in them. For instance, the global schema $\mathcal{G}$ in Fig. 2b consists of two relations Book and Book_Price. Relation Book has attributes ISBN, title, suggested retail price, author and publisher, while Book_Price stores the book price and stock information for different sellers. Data is fueled into the system by two sources; the databases of the bookstore Barnes & Noble (B&N) and the publisher Prentice Hall (PH), with the schemas shown in Fig. 2a. Note that there are two versions of the PH schema, used in different examples.

Global as View (GAV)

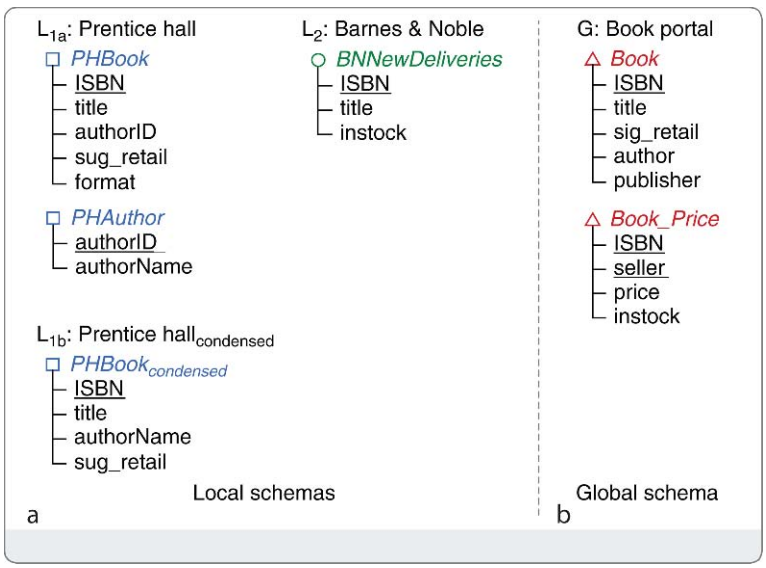
Historically, the first VDISs followed the Global as View (GAV) approach [7,12], in which the global schema is described in terms of the local schemas. In such systems the contents of each relation R in the target schema $\mathcal{G}$ are specified through a query (view) V over the combined schemas of the sources.

Thus, in GAV the correspondence between the local schemas and the global schema can be described through a set of mappings of the form:

$$V_i \rightarrow I(R_i)$$

one for each relation R_i of the global schema, where V_i is a query over the combined source schemas. $I(R_i)$ is the identity query over R_i (i.e., a query that returns all attributes of R_i). The symbol $\rightarrow$ can represent either query containment ($\subseteq$) or query equality ($=$). This leads to two different semantics, referred to in the literature as the open-world and the closed-world assumption, respectively.

Example: Consider the following two GAV mappings (For the examples, the identity query I over some relation is considered to return the attributes of that relation in the same order as they appear on the schema in Fig. 2).



View-based Data Integration. Figure 2. Local and global schemas of the running example.

$M_1 : V_1 \rightarrow I(Book)$
$M_2 : V_2 \rightarrow I(Book\_Price)$

where

$V_1(ISBN, title, sug\_retail, authorName, \text{“}PH\text{”}):-$
$PHBook(ISBN, title, authorID, sug\_retail, format),$
$PHAuthor(authorID, authorName)$

and

$V_2(ISBN, \text{“}B\&N\text{”}, sug\_retail, instock):-$
$PHBook(ISBN, title1, authorID, sug\_retail, format),$
$BNNewDeliveries(ISBN, title2, instock)$

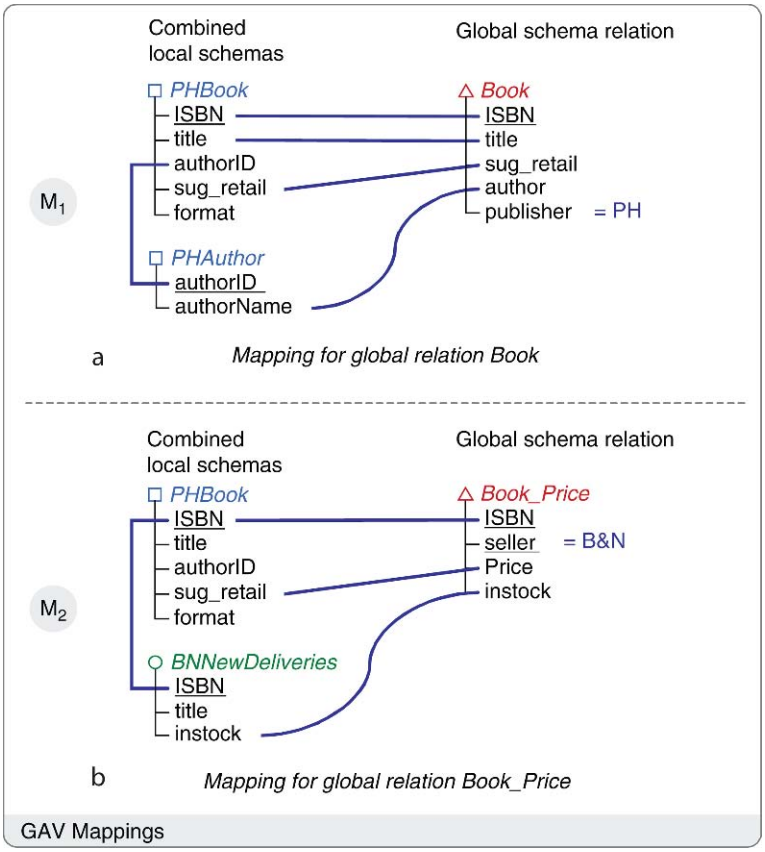
The mappings are graphically depicted in Fig. 3a and b, as described in [9]. This is similar to the way most *mapping tools* (i.e., tools that allow a visual specification of mappings), such as IBM Clio and MS BizTalk Mapper, display mappings.

Mapping M_1 intuitively describes how Book tuples in the global database are created. This is done by retrieving the ISBN, title and sug_retail from a PHBook tuple, the author from the corresponding PHAuthor tuple (i.e., the PHAuthor tuple with the same authorID as the PHBook tuple), and finally setting the publisher to “PH” (since the extracted books are published by PH).

Similarly, mapping M_2 describes the construction of the global relation Book_Price. This involves combining information from multiple sources: the price from the suggested retail price information provided by PH and the inventory information from B&N, because B&N's administrator knows that B&N's sells its books at the suggested retail price.

Query Answering in GAV. GAV mappings have a *procedural* flavor, since they describe how the global database can be constructed from the local databases. For this reason, query answering in GAV is straightforward, both in the materialized and in the virtual approach.

In the materialized approach, the source data are replicated in the global database by executing for each mapping $V_i \rightarrow I(R_i)$ the query V_i against the local databases and populating R_i with the query results.



View-based Data Integration. Figure 3. Example of GAV mappings.

Subsequently, an application query Q against the global schema is answered by simply running Q over the materialized global database.

On the other hand, in the virtual approach, data are kept in the sources and thus a query against the global schema has to be translated to corresponding queries against the local schemas. Due to the procedural flavor of GAV, this can be done through view unfolding (i.e., replacing each relational atom of the global schema in the query by the corresponding view definition). Intuitively, whenever a query asks for a global relation R_i, it will instead run the subquery V_i over the local schemas, which, according to the mapping $V_i \rightarrow I(R_i)$, provides the contents of R_i.

Advantages. The simplicity of the GAV rules together with the straight-forward implementation of query answering, led to the wide adoption of GAV by industrial systems. From the research sector representative GAV-based VDISs systems are MULTIBASE [11], TSIMMIS [5] and Garlic [2].

Disadvantages. GAV also has several drawbacks:

First, since the global schema is expressed in terms of the sources, *global relations cannot model any information not present in at least one source.* For instance, the Book relation in the example could not contain an attribute for the book weight, since no source currently provides it. In other words, the value of each global attribute has to be explicitly specified (i.e., in the visual representation all global attributes must have an incoming line or an equality with a constant).

Second, as observed in mapping M_2 of the running example, *a mapping has to explicitly specify how data from multiple sources are combined to form global relation tuples.* Therefore, GAV-based systems do not facilitate adding a source to the system independently of other sources. Instead, when a new source wants to join the system, the system administrator has to inspect how its data can be merged with those of the other sources currently in the system and modify the corresponding mappings.

Local as View (LAV)

To overcome the shortcomings of GAV, researchers came up with the Local as View (LAV) approach [7,12]. While in GAV the global schema is described in terms of the local schemas, LAV follows the opposite direction expressing each local schema as a function of the global schema. LAV essentially corresponds to the "source owners view" of the system by describing which data of the global database are present in the source. Using the same notation as in GAV, local-to-global correspondences can be written in LAV as a set of mappings:

$$I(R_i) \rightarrow U_i$$

one for every relation R_i in the local schemas, where U_i is a query over the global schema and I the identity query.

Example: Figure 4 shows the following two LAV mappings for the running example:

$M'_1 : I(PHBook_{condensed}) \rightarrow U_1$
$M'_2 : I(BNNewDeliveries) \rightarrow U_2$

where

$U_1(ISBN, title, author, sug\_retail)$:-
  $Book(ISBN, title, sug\_retail, author, \text{“}PH\text{”})$

and

$U_2(ISBN, title, instock)$:-
  $Book(ISBN, title, sug\_retail, author, publisher)$,
  $Book\_Price(ISBN, \text{“}B\&N\text{”}, sug\_retail, instock)$

For instance, M'_1 specifies that $PHBook_{condensed}$ contains information about books published by PH. Similarly, M'_2 declares that *BNNewDeliveries* contains the ISBN and title of books sold by B&N at their suggested retail price and whether B&N has them in stock.

In contrast to GAV mappings, LAV mappings have a *declarative* flavor, since, instead of explaining how the global database can be created, they describe what information of the global database is contained in each local database.

Advantages. LAV addresses many of GAV problems with the most important being that sources can register independently of each other, since a source's mappings do not refer to other sources in the system.

Disadvantages. LAV suffers from the symmetric drawbacks of GAV. In particular, it cannot model sources that have information not present in the global schema (this is the reason why the example above used the condensed version of PH's schema that did not contain the attribute format, which is not present in the global schema). Furthermore, due to LAV's declarative nature, query answering is non-trivial any more, as described next. Mainly because of its technical implications to query answering, LAV has been extensively studied in the literature. Representative LAV-based systems include Information Manifold [10] and the system described in [13].

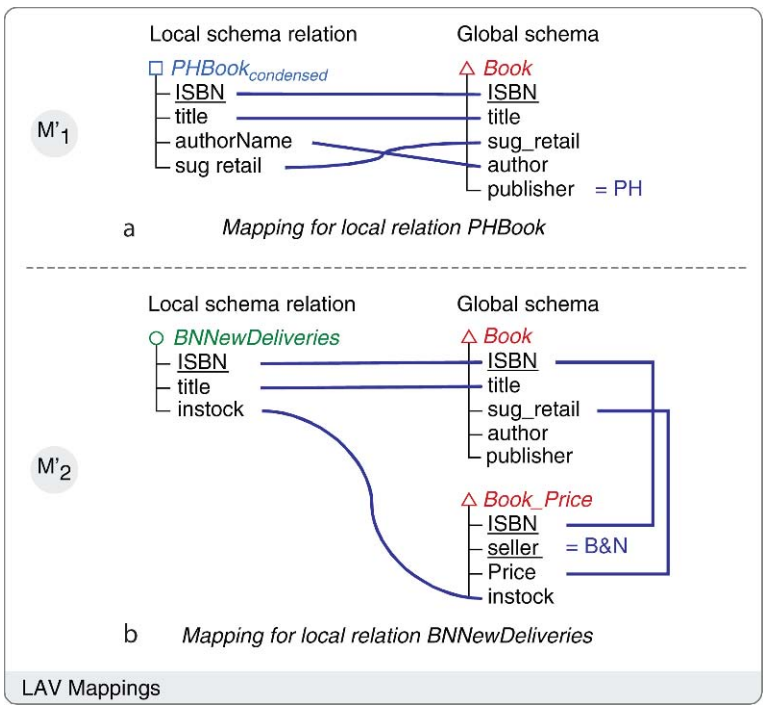
Query Answering in LAV. Since LAV mappings consist of an arbitrary query over the global schema, they may leave some information of the global database unspecified. For instance, mapping M'_2 above only states that B&N sells its books at the suggested retail price, without specifying the exact price. Thus, there might be infinitely many global databases that could be inferred from the sources through the mappings (each of them would make sure that each pair of Book and Book_Price tuples created from a BNNewDeliveries tuple share the same value for price, but each such global database might choose a different constant for the value). These databases are called *possible worlds*. Their existence has two important implications:

First, since a unique global database does not exist, it cannot be materialized and therefore LAV lends itself better to virtual mediation. However, there is still a way of replicating source information in a centralized place. This involves creating a "special" database that intuitively stores the general shape of all possible worlds. This "special" database is called *canonical universal solution* and can be built through procedures employed in data exchange [3].

Second, since many global databases exist, the query answering semantics need to be redefined. The standard semantics adopted in the literature for query answering in LAV-based systems are based on the notion of *certain answers* [1,8]. The certain answers to a query are the answers to the query, which will always appear regardless of which possible world the query is executed against (i.e., the tuples that appear in the intersection of the sets of query answers against each possible world). Intuitively, certain answers return information that *is guaranteed* to exist in *any* possible world.

Example: If in the integration system of Fig. 4 a query asks for all ISBNs that are sold by some seller at their suggested retail price, it will get back all ISBNs stored in relation BNNewDeliveries, because for each of them, any possible world will contain a pair of Book and Book_Price tuples with the same ISBN and the same prices (although these prices will have different values between possible worlds). On the other hand, if the query asks for all books sold at a specific price, it will not get back the ISBNs from BNNewDeliveries, because their exact prices are left unspecified by the mapping M'_2 and will therefore differ among possible worlds.

In order to compute the certain answers to a query in a virtual integration system following the LAV approach, the query against the global schema has to be translated to corresponding queries against the local schemas. This problem is called *rewriting queries using*



View-based Data Integration. Figure 4. Example of LAV mappings.

views (because the query over the global database has to be answered by using the sources which are expressed as views over it), and is also of interest to other areas of database research, such as query optimization and physical data independence (see [8] for a survey). In contrast to GAV, it is a non-trivial problem studied extensively by researchers.

Global and Local as View (GLAV)

To overcome the limitations of both GAV and LAV, [4] proposed a new category of mapping languages, called Global and Local as View (GLAV), which is a generalization of both GAV and LAV. GLAV mappings are of the form:

$$V_i \rightarrow U_i$$

where V_i, U_i are queries over the local and global schemas, respectively.

GLAV languages can trivially express both GAV mappings and LAV mappings by assigning to U_i a query returning a single global relation or to V_i a query asking for a single local relation, respectively. However, GLAV is a strict superset of both GAV and LAV by allowing the formulation of mappings that do not fall either under GAV or under LAV (i.e., mappings in which V_i and U_i both do not return just a single local or global relation).

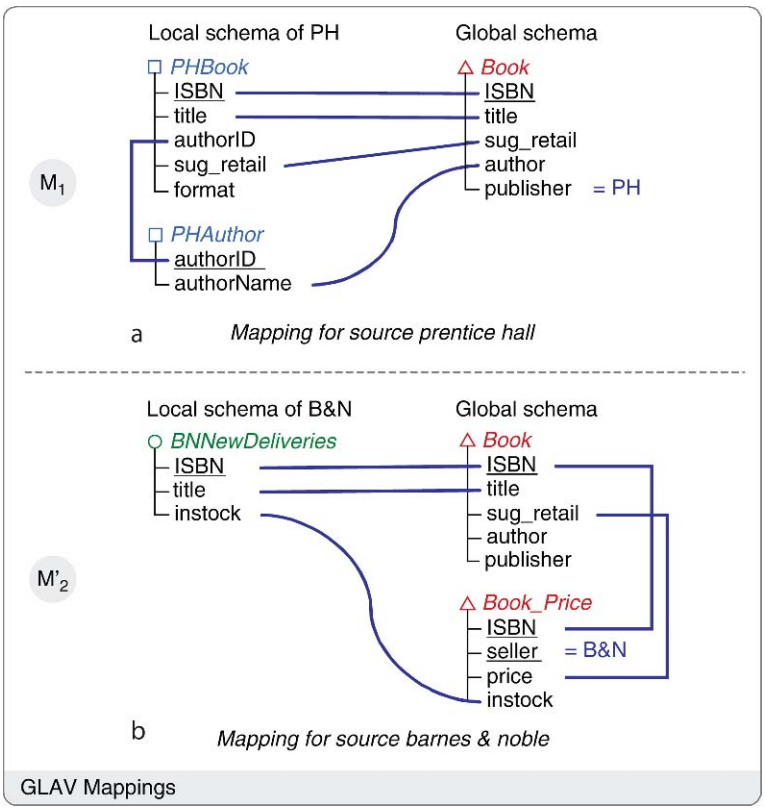
Example: Figure 5 shows two GLAV mappings. The first mapping is the GAV mapping M_1 first presented in Fig. 3a, while the second mapping is the LAV mapping M'_2 used in Fig. 4b.

Since U_i (a.k.a. the conclusion of the mapping) can be an arbitrary query over the global schema, GLAV allows the independent registration of sources in the same way as LAV. However, this also implies that the global database is incomplete. Consequently, query answering in GLAV is usually done under certain answer semantics, by extending query rewriting algorithms for LAV [15].

Alternatives to VDISs

Apart from the VDISs, research and industrial work led to many alternative approaches to data integration of structured data:

1. *Vertical Integration Systems* are specialized applications that solve the problem of data



View-based Data Integration. Figure 5. Example of GLAV mappings.

integration for a specific domain. For example, http://www.mySimon.com or http://www.rottentomatoes.com integrate price and movie information, respectively.

2. *Extract Transform Load (ETL) Tools* generally facilitate the actual migration of data from one system to another. When used for data integration they are closely tied to the problem of materializing the integrated database in a central data warehouse.

Compared to these solutions, VDISs offer a more general approach to data integration with the following advantages: (i) The relationships between the sources and the unified view are explicitly stated and not hidden inside a particular implementation, (ii) a general VDIS implementation can be used in many different domains, and (iii) a VDIS deployment can be easily utilized by many applications, as shown in Fig. 1. In a way VDISs are analogous to Database Management Systems, offering a general way of managing the data (which in VDISs are heterogeneous and distributed), independently of the applications that need those data.

Finally, another alternative to VDISs are Peer-to-Peer (P2P) Integration Systems that drop the requirement for a single unified view, allowing queries to be posed over any source schema. Although it is an active area of research, P2P systems have not yet been widely adopted in industry.

Key Applications

VDISs are used for integration of structured data in many different settings, including among others enterprises, government agencies and scientific communities.

Cross-references

- ▶ Information Integration
- ▶ Peer-to-Peer Data Integration
- ▶ Query Translation
- ▶ Query Rewriting Using Views

Recommended Reading

1. Abiteboul S. and Duschka O.M. Complexity of answering queries using materialized views. In Proc. 17th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, 1998, pp. 254–263.
2. Carey M.J., Haas L.M., Schwarz P.M., Arya M., Cody W.F., Fagin R., Flickner M., Luniewski A., Niblack W., Petkovic D., Thomas II J., Williams J.H., and Wimmers E.L. Towards heterogeneous multimedia information systems: The Garlic approach. In Proc. 5th Int. Workshop on Research Issues on Data Eng., 1995, pp. 124–131.
3. Fagin R., Kolaitis P.G., Miller R.J., and Popa L. Data exchange: Semantics and query answering. In Proc. Int. Conf. on Database Theory, 2002. pp. 207–224.
4. Friedman M., Levy A., and Millstein T. Navigational plans for data integration. In Proc. 16th National Conf. on AI and 11th Innovative Applications of AI Conf., 1999.
5. Garcia-Molina H.K., Papakonstantinou Y.K., Quass D.K., Rajaraman A.K., Sagiv Y.K., Ullman J.K., Vassalos V.K., and Widom J.K. The TSIMMIS approach to mediation: data models and languages. J. Intell. Inf. Syst., 8(2):117–132, 1997.
6. Genesereth M.R., Keller A.M., and Duschka O.M. Infomaster: An information integration system. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1997.
7. Halevy A. Logic-based techniques in data integration. In Logic Based Artif. Intell., 2000.
8. Halevy A.Y. Answering queries using views: A survey. VLDB J., 10(4):270–294, 2001.
9. Katsis Y., Deutsch A., and Papakonstantinou Y. interactive source registration in community-oriented information integration. In Proc. 34th Int. Conf. on Very Large Data Bases, 2008.
10. Kirk T., Levy A.Y., Sagiv Y., and Srivastava D. The information manifold. In Information Gathering from Heterogeneous, Distributed Environments, 1995.
11. Landers T. and Rosenberg R.L. An overview of MULTIBASE. Distributed systems, Vol. II: distributed data base systems table of contents, 1986, pp. 391–421.
12. Lenzerini M. Data integration: A theoretical perspective. In Proc. 21st ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, 2002.
13. Manolescu I., Florescu D., and Kossmann D. Answering XML queries over heterogeneous data sources. In Proc. 27th Int. Conf. on Very Large Data Bases, 2001.
14. Widom J. Research problems in data warehousing. In Proc. 27th Int. Conf. on Very Large Data Bases, 1995.
15. Yu C. and Popa L. Constraint-based XML query rewriting for data integration. In Proc. 27th Int. Conf. on Very Large Data Bases, 2004.

Views

Yannis Kotidis
Athens University of Economics and Business, Athens, Greece

Definition

Views are virtual relations without independent existence in a database. The contents of the instance of a view are determined by the result of a query on a set of database tables or other views in the system.

Key Points

Over the years, there have been several definitions and uses for the term view. From one perspective, a view is a query (or a macro) that generates data every time it is invoked. The term view is also used in association with the derived data that is produced by the execution of the view query. Views have also been used as indexes, summary tables or combinations of both. They provide physical data independence, by decoupling the logical schema from the physical design, which is usually driven by performance reasons. By controlling access to the views, one can control access to different parts of the data and implement different security policies. Views are also used in integration systems in order to provide unified access to physically remote databases.

When the result of the view is stored as an independent table, the view is called materialized. Materialized views can be of two forms. In the first, the materialized table is treated as pure data, detached from the view definition that was used to derive its content. In the second, the view content is treated as derived data that needs to be properly maintained when update statements alter the state of the database, in a way that affects the instance of the view. This is called the view maintenance problem. In a symmetrical way, when a user or a program modifies a view (materialized or not) through an update statement, an implementation is required for translating the view update command into a series of updates on the base relations so that the requested update is observed in the view instance. This process is often termed update through views.

Cross-references

- ▶ Answering Queries Using Views
- ▶ Updates Through Views
- ▶ View Definition
- ▶ View Maintenance

Recommended Reading

1. Adiba M.E. and Lindsay B.G. Database snapshots. In Proc. 6th Int. Conf. on Very Data Bases, 1980, pp. 86–91.
2. Dayal U. and Bernstein P. On the correct translation of update operations on relational views. ACM Trans. Database Syst., 8(3):381–416, 1982.
3. Gupta A., Jagadish H.V., and Singh Mumick I. Data integration using self-maintainable views. In Advances in Database Technology, Proc. 5th Int. Conf. on Extending Database Technology, 1996, pp. 140–144.
4. Gupta H., Harinarayan V., Rajaraman A., and Ullman J.D. Index selection for OLAP. In Proc. 13th Int. Conf. on Data Engineering, 1997, pp. 208–219.
5. Kotidis Y. and Roussopoulos N. DynaMat: a dynamic view management system for data warehouses. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1999, pp. 371–382.
6. Roussopoulos N. View indexing in relational databases. ACM Trans. Database Syst., 7(2):258–290, 1982.
7. Roussopoulos N. An incremental access method for ViewCache: concept, algorithms, and cost analysis. ACM Trans. Database Syst., 16:535–563, 1991.

Virtual Disk Manager

- ▶ Logical Volume Manager (LVM)

Virtual Health Record

- ▶ Electronic Health Records (EHR)

Virtual Partitioning

Marta Mattoso
Federal University of Rio de Janeiro, Rio de Janeiro, Brazil

Synonyms

VP

Definition

Virtual Partitioning (VP) is a distribution database design technique [2] that avoids physical partitioning table design. VP is adopted in Database Clusters to implement intra-query parallelism [3]. VP is based on database replication and aims at designing a table partition dynamically, according to each query specification. The idea is to take advantage of the current configuration of the parallel execution environment, such as the number of nodes available, the tables of the query, the current load, and the available table replicas. Intra-query parallelism can be obtained through VP by rewriting a query into a set of sub-queries to be sent to different virtual partitions. At each node, the local sequential DBMS processes the sub-query on the specified virtual partition of the table. VP requires an extra query processing phase to compose the final result

from the partial ones produced by the sub-queries. This way, a database that is not physically partitioned can still be processed transparently through intra-query parallelism.

Key Points

VP is implemented through query rewriting [1]. Basically, a DBC rewrites the original query by adding range predicates to it, which originates as many sub-queries as the number of table replicas and nodes available. Then, each node receives a different sub-query. For example, let **Q** be a query on table `orders`:

```
Q: select sum (price)
   from orders
   where category = 'bolt';
```

A generic sub-query on a virtual partition is obtained by adding to Q's `where` clause the predicate "and `order_id` >=v1 and `order_id` < v2." By binding [v1, v2] to n subsequent ranges of `order_id` values, n sub-queries are generated, each for a different node on a different virtual partition of `orders`. These virtual partitions correspond to horizontally partitioned data. After the execution of all sub-queries each partial result would then be inserted into a temporary table, e.g., `tempResult`. Finally, to compose the result, the partitioned results need to be combined by an aggregate query. In this example: `select sum (sprice) from tempResult;`

For VP to be effective, the tuples of the virtual partition must be physically clustered according to the added range predicate. Ideally, there should be a clustered ordered index on table `orders` based on `order_id`, so that the DBMS will not scan tuples outside the VP.

Cross-references

- ▶ Clustering Index
- ▶ Data Partitioning
- ▶ Data Replication
- ▶ Distributed Database Design
- ▶ Horizontally Partitioned Data
- ▶ Intra-query Parallelism

Recommended Reading

1. Lima A.A.B., Mattoso M., and Valduriez P. Adaptive virtual partitioning for OLAP query processing in a database cluster. In Proc. 19th Brazilian Symp. on Database Systems, 2004, pp. 92–105.
2. Özsu T. and Valduriez P. Principles of Distributed Database Systems (2nd edn.). Prentice Hall, Upper Saddle River, NJ, 1999.
3. Röhm U., Böhm K., Scheck H.-J., and Schuldt H. FAS – A freshness-sensitive coordination middleware for a cluster of OLAP components. In Proc. 28th Int. Conf. on Very Large Data Bases, 2002, pp. 754–768.

Vision

- ▶ Visual Perception

Visual Analysis

- ▶ Visual Analytics
- ▶ Visual Data Mining

Visual Analytics

Daniel A. Keim, Florian Mansmann, Andreas Stoffel, Hartmut Ziegler
University of Konstanz, Konstanz, Germany

Synonyms

Visual analysis; Visual data analysis; Visual data mining

Definition

Visual analytics is the science of analytical reasoning supported by interactive visual interfaces. Over the last decades, data was produced at an incredible rate. However, the ability to collect and store this data is increasing at a faster rate than the ability to analyze it. While purely automatic or purely visual analysis methods were developed in the last decades, the complex nature of many problems makes it indispensable to include humans at an early stage in the data analysis process. Visual analytics methods allow decision makers to combine their flexibility, creativity, and background knowledge with the enormous storage and processing capacities of today's computers to

gain insight into complex problems. The goal of visual analytics research is thus to turn the information overload into an opportunity: Decision-makers should be enabled to examine this massive, multi-dimensional, multi-source, time-varying, and often conflicting information stream through interactive visual representations to make effective decisions in critical situations.

Historical Background

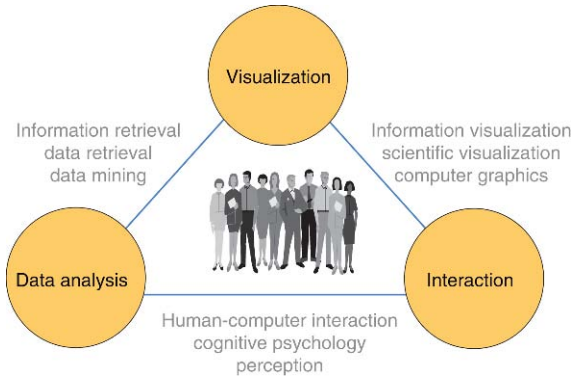
Automatic analysis techniques such as statistics and data mining developed independently from visualization and interaction techniques. However, some key thoughts changed the rather limited scope of the fields into what is today called visual analytics research. One of the most important steps in this direction was the need to move from confirmatory data analysis to exploratory data analysis, which was first stated in the statistics research community by John W. Tukey in his book "Exploratory data analysis" [7].

Later, with the availability of graphical user interfaces with proper interaction devices, a whole research community devoted their efforts to information visualization [1,2,5,8]. At some stage, this community recognized the potential of integrating the user in the KDD process through effective and efficient visualization techniques, interaction capabilities and knowledge transfer leading to *visual data exploration* or *visual data mining* [3]. This integration considerably widened the scope of both the information visualization and the data mining fields, resulting in new techniques and plenty of interesting and important research opportunities.

The term *visual analytics* was coined by Jim Thomas in the research and development agenda "Illuminating the Path" [6], which had a strong focus on Homeland Security in the United States. Meanwhile, the term is used in a wider context, describing a new multidisciplinary field that combines various research areas including visualization, human-computer interaction, data analysis, data management, geo-spatial and temporal data processing, and statistics [4].

Foundations

Visual analytics evolved from information visualization and automatic data analysis. It combines both former independent fields and strongly encourages human interaction in the analysis process as illustrated in Fig. 1. The focus of this section is to differentiate between visualization and visual analytics and thereby motivating its necessity. Thereafter, the visual analytics process is described and technical as well as social challenges of visual analytics are discussed.



Visual Analytics. Figure 1. Visual analytics as the interplay between data analysis, visualization, and interaction methods.

Visualization is the communication of data through the use of interactive interfaces and has three major goals: (i) presentation to efficiently and effectively communicate the results of an analysis, (ii) confirmatory analysis as a goal-oriented examination of hypotheses, and (iii) exploratory data analysis as an interactive and usually undirected search for structures and trends.

Visual analytics is more than only visualization. It can rather be seen as an integral approach combining visualization, human factors, and data analysis. Visualization and visual analytics both integrate methodology from information analytics, geospatial analytics, and scientific analytics. Especially human factors (e.g., interaction, cognition, perception, collaboration, presentation, and dissemination) play a key role in the communication between human and computer, as well as in the decision-making process. In matters of data analysis, visual analytics furthermore profits from methodologies developed in the fields of statistical analytics, data management, knowledge representation, and knowledge discovery. Note that visual analytics is not likely to become a separate field of study, but its influence will spread over the research areas it comprises [9].

Overlooking a large information space is a typical visual analytics problem. In many cases, the information at hand is conflicting and needs to be integrated from heterogeneous data sources. Often the computer system lacks knowledge that is still hidden in the expert's mind. By applying analytical reasoning, hypotheses about the

data can be either affirmed or discarded and eventually lead to a better understanding of the data. Visualization is used to explore the information space when automatic methods fail and to efficiently communicate results. Thereby human background knowledge, intuition, and decision-making either cannot be automated or serve as input for the future development of automated processes. In contrast to this, a well-defined problem where the optimum or a good estimation can be calculated by non-interactive analytical means would rather not be described as a visual analytics problem. In such a scenario, the non-interactive analysis should be clearly preferred due to efficiency reasons. Likewise, visualization problems not involving methods for automatic data analysis do not fall into the field of visual analytics.

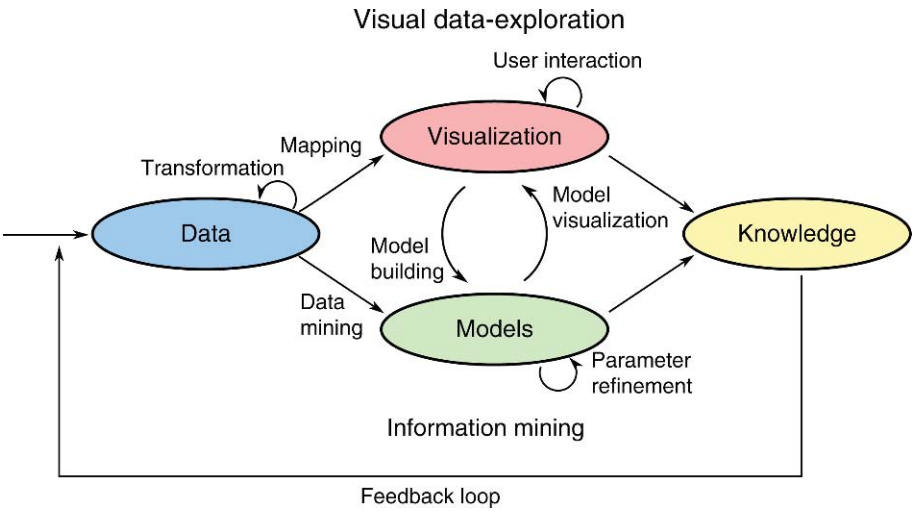
Visual Analytics Process

The visual analytics process is a combination of automatic and visual analysis methods with a tight coupling through human interaction in order to gain knowledge from data. Figure 2 shows an abstract overview of the different stages (represented through ovals) and their transitions (arrows) in the visual analytics process.

In many visual analytics scenarios, heterogeneous data sources need to be integrated before visual or automatic analysis methods can be applied. Therefore, the first step is often to preprocess and transform the data in order to extract meaningful units of data for further processing. Typical preprocessing tasks are data cleaning, normalization, grouping, or integration of heterogeneous data into a common schema.

Continuing with this meaningful data, the analyst can select between visual or automatic analysis methods. After mapping the data the analyst may obtain the desired knowledge directly, but more likely is the case that an initial visualization is not sufficient for the analysis. User interaction with the visualization is needed to reveal insightful information, for instance by zooming in different data areas or by considering different visual views on the data. In contrast to traditional information visualization, findings from the visualization can be reused to build a model for automatic analysis. As a matter of course these models can also be built from the original data using data mining methods. Once a model is created the analyst has the ability to interact with the automatic methods by modifying parameters or selecting other types of analysis algorithms. Model visualization can then be used to verify the findings of these models. Alternating between visual and automatic methods is characteristic for the visual analytics process and leads to a continuous refinement and verification of preliminary results. Misleading results in an intermediate step can thus be discovered at an early stage, which leads to more confidence in the final results.

In the visual analytics process, knowledge can be gained from visualization, automatic analysis, as well as the preceding interactions between visualizations, models, and the human analysts. The feedback loop



Visual Analytics. Figure 2. The *Visual Analytics Process* is characterized through interaction between data, visualizations, models about the data, and the users in order to discover knowledge.

stores this knowledge of insightful analyses in the system, and contributes to enable the analyst to draw faster and better conclusions in the future.

Technical and Social Challenges

While visual analytics profits from the increasing computational power of computer systems, faster networks, high-resolution displays, as well as novel interaction devices, it must be kept in mind that new technologies are always accompanied with a variety of technical challenges that have to be solved.

Dynamic processes in scientific or business applications often generate large streams of real-time data, such as sensor logs, web statistics, network traffic logs, or atmospheric and meteorological data. The analysis of such *large data streams* which can consist of terabytes or petabytes of data is one of the technical challenges since advances in many areas of science and technology are dependent upon the capability to analyze these data streams. As the sheer amount of data does often not allow to store all data at full detail, effective compression and feature extraction methods are needed to manage the data. Visual analytics aims at providing techniques that make humans capable of analyzing real time data streams by presenting results in a meaningful and intuitive way while allowing interaction with the data. These techniques enable quick identification of important information and timely reaction on critical process states or alarming incidents.

Synthesis of heterogeneous data sources is another challenge that is closely related to data streams, because real-world applications often access information from a large number of different information sources including collections of vector data, strings and text documents, graphs or sets of objects. Integrating these data sources includes many fundamental problems in statistics, machine learning, decision theory, and information theory. Therefore, the focus on scalable and robust methods for fusing complex and heterogeneous data sources is key to a more effective analysis process.

One step further in the analysis process, *interpretability and trustworthiness* or the ability to recognize and understand the data can be seen as one of the biggest challenges in visual analytics. Generating a visually correct output from raw data and drawing the right conclusions largely depends upon the quality of the used data and methods. A lot of possible quality problems (e.g., data capture errors, noise, outliers, low precision, missing values, coverage errors, double counts) can already be contained in the raw data. Furthermore, pre-processing of data in order to use it for visual analysis bears many potential quality problems (i.e., data migration and parsing, data cleaning, data reduction, data enrichment, up-/down-sampling, rounding and weighting, aggregation and combination). The concrete challenges are on the one hand to determine and to minimize these errors on the pre-processing side, and a flexible yet stable design of the visual analytics applications to cope with data quality problems on the other hand. From a technical point of view, the design of such applications should either be insensitive to data quality issues through usage of data cleaning methods or explicitly visualize errors and uncertainty to raise awareness for data quality issues.

In many scenarios, interpreting the raw data only makes little or no sense at all if it cannot be embedded in context. Research on *semantics* may derive this context from meta data by capturing associations and complex relationships. Ontology-driven techniques and systems have already started to enable new semantic applications in a wide span of fields such as bioinformatics, web services, financial services, business intelligence, and national security. Research challenges thereby arise from the size of ontologies, content diversity, heterogeneity as well as from computation of complex queries and link analysis over ontology instances and meta data.

Scalability in general is a key challenge of visual analytics, as it determines the ability to process large datasets by means of computational overhead as well as appropriate rendering techniques. Often, the huge amount of data that has to be visualized exceeds the limited amount of pixels of a display by several orders of magnitude. In order to cope with such a problem not only the absolute data growth and hardware performance have to be compared, but also the software and the algorithms to bring this data in an appropriate way onto the screen. As the amount of data is continuously growing and the amount of pixels on the display remains rather constant, the rate of compression on the display is steadily increasing. Therefore, more and more details get lost. It is an essential task of visual analytics to create a higher-level view onto the dataset, while maximizing the amount of details at the same time.

The field of *problem solving, decision science, and human information discourse* constitutes a further visual analytics challenge since it not only requires

understanding of technology, but also comprehension of typical human capabilities such as logic, reasoning, and common sense. Many psychological studies about the process of problem solving have been conducted. In a usual test setup the subjects have to solve a well-defined problem where the optimal solution is known to the researchers. However, real-world problems are manifold. In many cases these problems are intransparent, consist of conflicting goals, and are complex in terms of large numbers of items, interrelations, and decisions involved. The dynamics of information that changes over time should not be underestimated since it might have a strong impact on the right decision. Furthermore, social aspects such as decision making in groups make the process even more delicate.

While many novel visualization techniques have been proposed, their wide-spread usage has not taken place primarily due to the users' refusal to change their daily working routines. *User acceptance* is therefore a further visual analytics challenge, since the advantages of developed tools need to be properly communicated to the audience of future users to overcome usage barriers, and to tap the full potential of the visual analytics approach.

Visual analytics tools and techniques should not stand alone, but should *integrate* seamlessly into the applications of diverse domains, and allow interaction with existing systems. Although many visual analytics tools are very specific (i.e., in astronomy or nuclear science) and therefore rather unique, in many domains (e.g., business or network security applications) integration into existing systems would significantly promote their usage by a wider community.

Finally, *evaluation* as a systematic analysis of usability, worth, and significance of a system is crucial to the success of visual analytics science and technology. During the evaluation of a system, different aspects can be considered such as functional testing, performance benchmarks, measurement of the effectiveness of the display in user studies, assessment of its impact on decision-making, or economic success to name just a few. Development of abstract design guidelines for visual analytics applications would constitute a great contribution.

Key Applications

Visual analytics is essential in application areas where large information spaces have to be processed and analyzed. Major application fields are *physics* and *astronomy*. Especially the field of *astrophysics* offers many opportunities for visual analytics techniques: Massive volumes of unstructured data, originating from different directions of space and covering the whole frequency spectrum, form continuous streams of terabytes of data that can be recorded and analyzed. With common data analysis techniques, astronomers can separate relevant data from noise, analyze similarities or complex patterns, and gain useful knowledge about the universe, but the visual analytics approach can significantly support the process of identifying unexpected phenomena inside the massive and dynamic data streams that would otherwise not be found by standard algorithmic means.

Monitoring *climate* and *weather* is also a domain which involves huge amounts of data, collected by sensors throughout the world and from satellites in short time intervals. A visual approach can help to interpret these massive amounts of data and to gain insight into the dependencies of climate factors and climate change scenarios, which would otherwise not be easily identified. Besides weather forecasts, existing applications visualize the global warming, melting of the poles, the stratospheric ozone depletion, as well as hurricane and tsunami warnings.

In the domain of *emergency management*, visual analytics can help determine the on-going progress of an emergency and identify the next countermeasures (e.g., construction of physical countermeasures or evacuation of the population) that must be taken to limit the damage. Such scenarios can include natural or meteorological catastrophes like flood or waves, volcanoes, storm, fire or epidemic growth of diseases (e.g., bird flu), but also human-made technological catastrophes like industrial accidents, transport accidents or pollution.

Visual analytics for *security* and *geographics* is an important research topic. The application field in this sector is wide, ranging from terrorism informatics, border protection, path detection to network security. Visual analytics supports investigation and detection of similarities and anomalies in large data sets, like flight customer data, GPS tracking or IP traffic data.

In *biology* and *medicine*, computer tomography, and ultrasound imaging for three-dimensional digital reconstruction and visualization produce gigabytes of medical data and have been widely used for years. The application area of bio-informatics uses visual analytics techniques to analyze large amounts of biological data.

From the early beginning of sequencing, scientists in these areas face unprecedented volumes of data, like in the Human Genome Project with three billion base pairs per human. Other new areas like Proteomics (studies of the proteins in a cell), Metabolomics (systematic study of unique chemical fingerprints that specific cellular processes leave behind) or combinatorial chemistry with tens of millions of compounds even enlarge the amount of data every day. A brute-force computation of all possible combinations is often not possible, but interactive visual approaches can help to identify the main regions of interest and exclude areas that are not promising.

Another major application domain for visual analytics is *business intelligence*. The financial market, with its hundreds of thousands of assets, generates large amounts of data every day, which accumulate to extremely high data volumes throughout the years. The main challenge in this area is to analyze the data under multiple perspectives and assumptions to understand historical and current situations, and then monitoring the market to forecast trends or to identify recurring situations. Other key applications in this area are fraud detection, detection of money laundering, or the analysis of customer data, insurance data, social data, and health care services.

Cross-references

- ▶ Cluster Visualization
- ▶ Comparative Visualization
- ▶ Data Mining
- ▶ Data Visualization
- ▶ Human-Computer Interaction
- ▶ Multidimensional Visualization Methods
- ▶ Multivariate Visualization Methods
- ▶ Parallel Visualization
- ▶ Scientific Visualization
- ▶ Visual Classification
- ▶ Visual Clustering
- ▶ Visual Content Analysis
- ▶ Visual Data Mining
- ▶ Visual Metaphor
- ▶ Visual On-Line Analytical Processing (OLAP)
- ▶ Visualization for Information Retrieval

Recommended Reading

1. Card S.W., Mackinlay J.D., and Shneiderman B. (eds.) Readings in Information Visualization: Using Vision to Think. Morgan Kaufmann, San Francisco, CA, USA, 1999.
2. Chen C. Information Visualization – Beyond the Horizon. Springer, Berlin, 2nd edn., 2004.
3. Keim D.A. Visual exploration of large data sets. Commun. ACM, 44(8):38–44, 2001.
4. Keim D.A. and Thomas J. Scope and challenges of visual analytics. Tutorial at IEEE Visualization Conf., 2007.
5. Spence R. Information Visualization – Design for Interaction. Pearson Education Limited, Harlow, England, 2nd edn., 2006.
6. Thomas J. and Cook K. Illuminating the path: Research and development agenda for visual analytics. IEEE-Press, Los Alamitos, CA, USA, 2005.
7. Tukey J.W. Exploratory data analysis. Addison-Wesley, Reading, MA, USA, 1977.
8. Ware C. Information Visualization – Perception for Design. Morgan Kaufmann, San Francisco, CA, USA, 1st edn., 2000.
9. Wong P.C. and Thomas J. Visual Analytics – Guest Editors' Introduction. IEEE Trans. Comput. Graph. Appl., 24(5):20–21, 2004.

Visual Association Rules

Li Yang
Western Michigan University, Kalamazoo, MI, USA

Synonyms

Association rule visualization

Definition

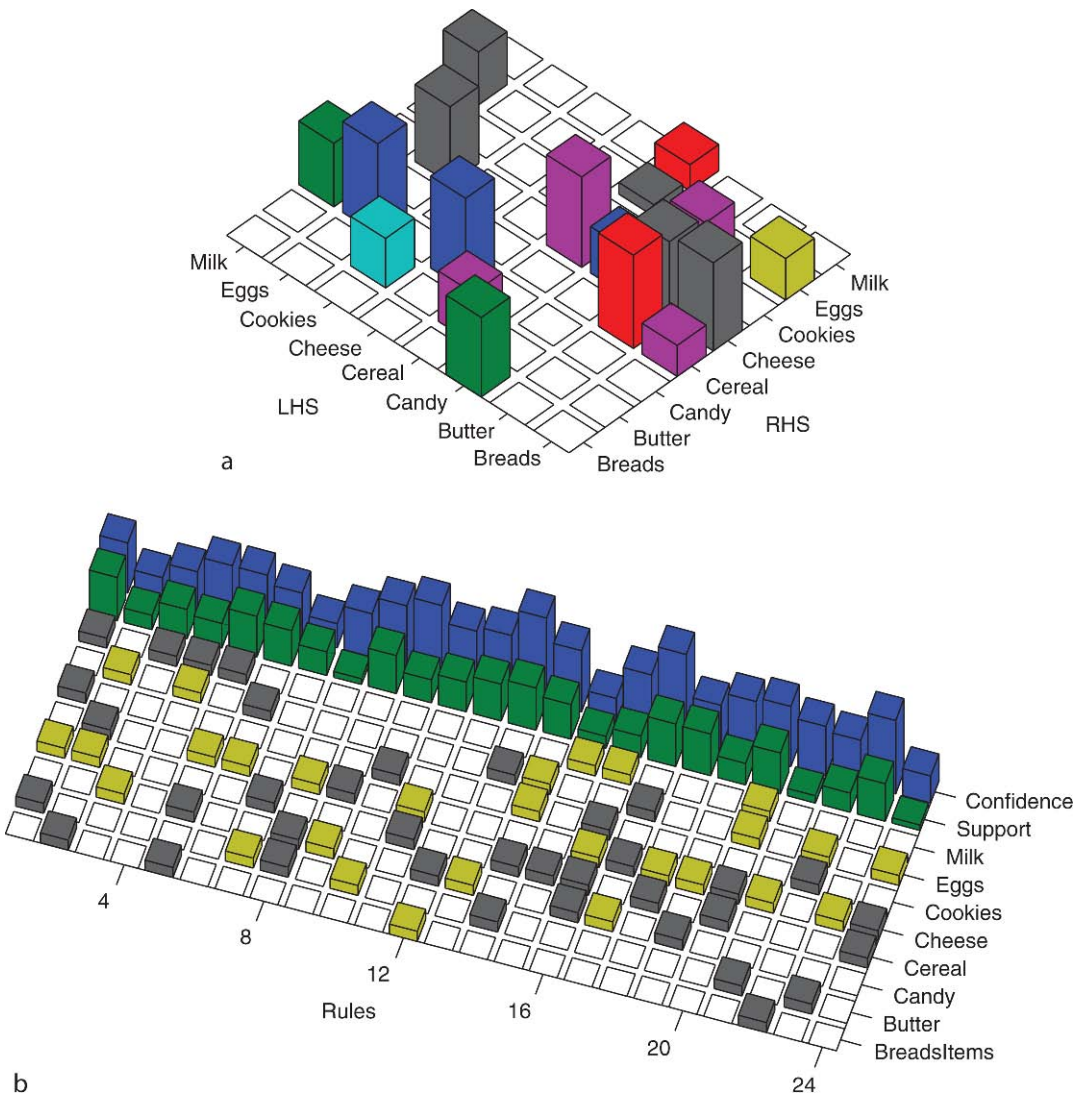
Association rule mining finds frequent associations between sets of data items from a large number of transactions. In market basket analysis, a typical association rule reads: *80% of transactions that buy diapers and milk also buy beer. The rule is supported by 10% of all transactions.* In this example, the 10% and 80% are called support and confidence, respectively. Depending on the user-specified minimum support and minimum confidence, association rule mining often produces too many rules for humans to read over. The answer to this problem is to select the most interesting rules. As interestingness is a subjective measure, selecting the most interesting rules is inherently human being's work. It is expected that information visualization may play an important role in managing a large number of association rules, and in identifying the most interesting ones.

Historical Background

An association rule reflects a many-to-many relationship. In lacking of an effective visual metaphor

to display many-to-many relationships, information visualization has received a technical challenge in order to display and interact with many association rules. Most existing approaches have been designed to visualize association rules in restricted forms. The restriction can be on the number of items on each side of the rule, the type of the rule, the total number of items, or the total number of rules. These existing approaches can be classified into two categories: matrix-based approaches and graph-based approaches.

A rectangular matrix can be used to visualize one-to-one association rules, where the left-hand side (LHS) items of all rules are listed along one axis and the right-hand side (RHS) of all rules are listed along the other axis of the rectangular matrix. A rule could be shown as an icon in the corresponding cell. Figure 1a shows an example 3D visualization, where the height and color of each bar represent support and confidence values of the corresponding rule, respectively. The major restriction of this approach is that it allows only one item on each side of the rule. Wong et al. [9] gave an alternative approach of arranging axes, where one axis is used to list items and the other is used to list rules. A rule is then visualized by a 3D bar chart against all items in the rule. This approach is able to visualize many-to-many rules, as long as the number of rules is kept reasonably small. Figure 1b illustrates the approach where the color of each bar indicates whether the



Visual Association Rules. Figure 1. Two major matrix-based approaches: (a) LHS versus RHS and (b) Rule versus Item.

item is on the LHS or RHS of the rule. Support and confidence values of the rules are visualized alongside the bar charts. Another matrix-based approach is to use mosaic plots and double deck plots [4] to visualize the contingency table of a frequent itemset that gives rise to a many-to-one association rule. However, only many-to-one rules derived from a single frequent itemset can be visualized each time. Fukuda et al. [2] gave a matrix-based approach of visualizing numeric association rules, which contain two numeric attributes on the LHS and a Boolean attribute on the RHS. The approach displays the set of rules as a bitmap image.

Directed graph is another technique to depict associations among items. There are two alternatives to assign graph nodes:

1. Each node represents an itemset. An association rule is visualized as an edge from the node representing its LHS itemset to the node representing its RHS itemset. Such an approach has been used in IBM DB2 Intelligent Miner. A major problem is that there may easily be too many nodes to display.
2. Each node represents an item. An association rule is visualized as a bunch of edges from LHS items to an intermediate node and another bunch of edges from the intermediate node to RHS items. Such an approach may work well only when a few items and rules are involved. The graph can quickly turn into a tangled mess with as few as a dozen rules. Klemettinen et al. [5] introduced a rule visualizer that uses this approach.

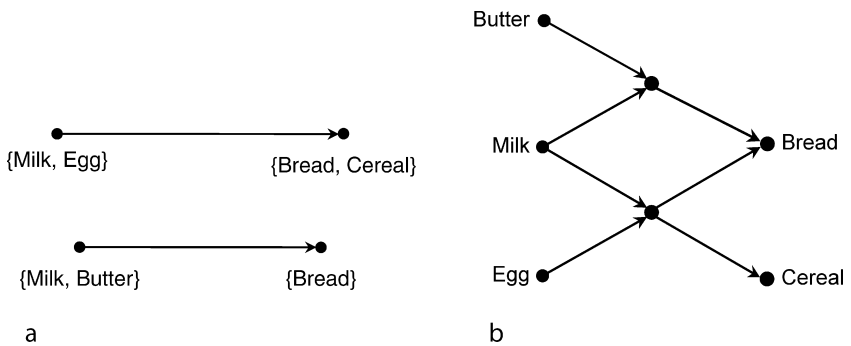
Figure 2 illustrates the two alternatives of visualizing association rules as directed graphs.

Visualization of association rules can be found in commercial data mining packages. SAS Enterprise Miner and SGI MineSet use matrix-based approaches and allow the user to visualize one-to-one association rules. IBM DB2 Intelligent Miner has a Associations Visualizer that uses the directed graph approach where each graph node displays an itemset.

Foundations

Association rule mining [1,3] is one of the mostly researched areas in data mining. Let $I = \{i_1, i_2,...,i_k\}$ be a set of items. A subset $A \subseteq I$ of items is called an itemset. Input data to association rule mining are n subsets $\{T_1, T_2,...,T_n\}$, called transactions, of I. Transaction T_i supports an itemset A if $A \subseteq T_i$. The percentage $P(A)$ of transactions that support A is called the support of A. A *frequent itemset* is an itemset whose support is no less than a minimum value specified by the user. An *association rule* is an expression $A \rightarrow B$ where A and B are itemsets and $A \cap B = \emptyset$. $P(A \cup B)$ is called the *support* of the rule. $P(A \cup B)/P(A)$ is called the *confidence* of the rule. The problem of association rule mining is to find all association rules whose supports are no less than a minimum support value and whose confidences are no less than a minimum confidence value. Frequent itemsets hold a well-known Apriori property: subsets of a frequent itemset are frequent. Efficient association rule mining algorithms [1] have been developed using the Apriori property for early pruning of search space.

The problem of too many discovered rules has also been studied. Klemettinen et al. [5] studied association rules at the border of frequent itemsets and used pattern templates to specify what the user wants to see. Testing criteria such as maximum entropy and



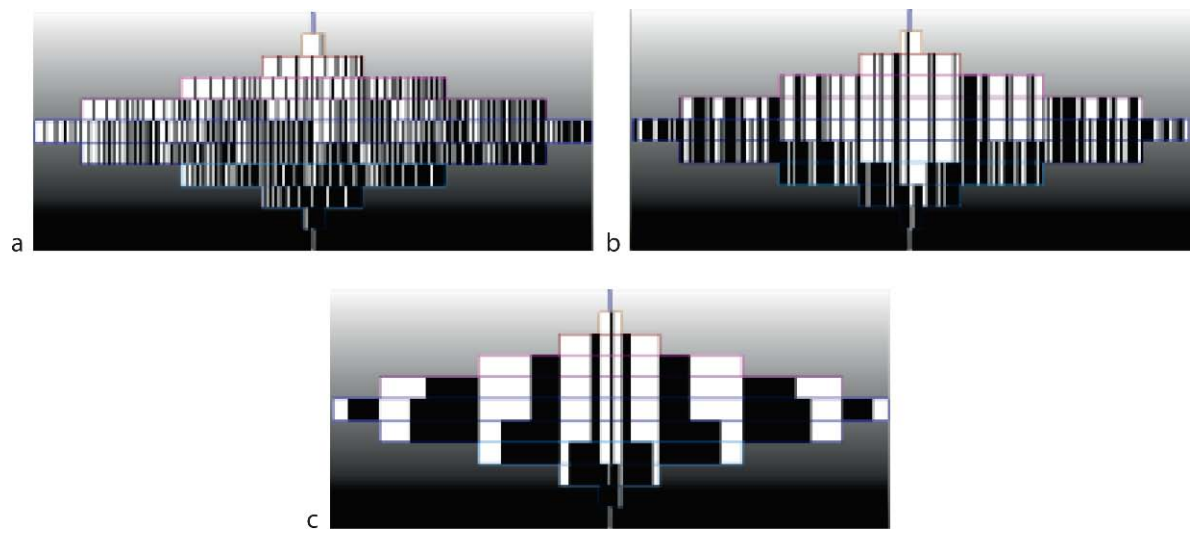
Visual Association Rules. Figure 2. Two major graph-based approaches: (a) Itemset as node and (b) Item as node.

chi-square(χ^2) have been suggested to replace the confidence test for the purpose of finding the most interesting rules. Liu et al. [7] have given a way of pruning and visualizing association rules with the presence of item taxonomy, by allowing a user to specify knowledge in terms of rules. Unexpected rules (that do not conform to the user's knowledge) are selected and visualized in a structured way.

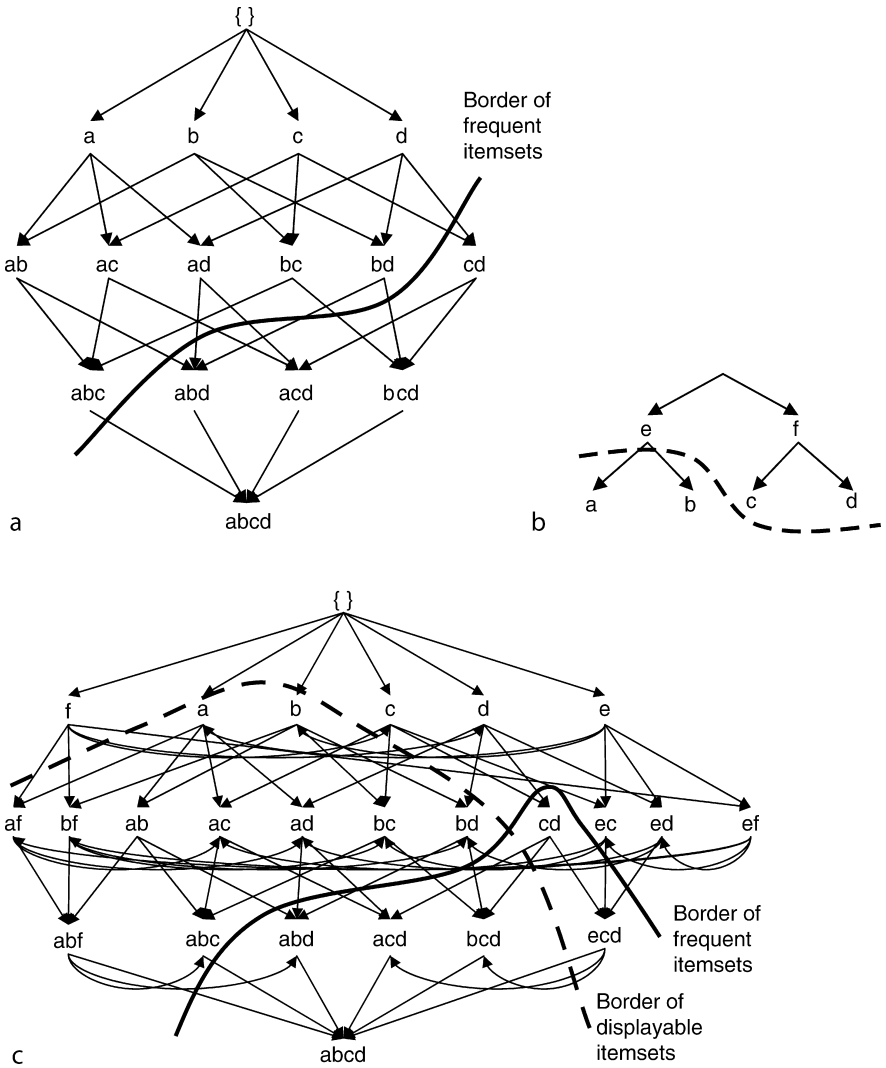
Association rules are difficult to visualize for several reasons. First, an association rule reflects a many-to-many relationship and there is no effective visual metaphor to display many many-to-one, never to say many-to-many, relationships. Second, frequent itemsets and association rules have inherent closure properties. For example, any subset of a frequent itemset is also frequent; if $a \rightarrow bc$ is a valid association rule, then $a \rightarrow b$, $a \rightarrow c$, $ab \rightarrow c$ and $ac \rightarrow b$ are all valid association rules. Third, the challenge is to visualize many such itemsets or rules. With the absence of an effective visual metaphor to present many-to-many relationships and to accommodate the closure properties, frequent itemsets and association rules pose fundamental challenges to information visualization.

Taking a function-theoretic view, the Apriori property basically says that frequent itemsets define an anti-monotone Boolean function $f(i_1,i_2,...,i_k)$ where $i_1,i_2,...,i_k$ are Boolean variables (items). As the function is anti-monotone, there is a border which separates 1's from 0's. Visualization of monotone Boolean functions has been studied by Kovalerchuk and Delizy in [6], although association rules are not their concern. Boolean vectors are displayed in multiple disk form where Boolean vectors of the same norm (equivalently, itemsets with the same number of items) are placed on the same disk. Figure 3 shows visualizations of an anti-monotone Boolean function $f(i_1,i_2,...,i_{10}) = \neg i_1$. Such a function is equivalent to frequent itemsets on $\{i_1,i_2,...,i_{10}\}$ where $\{i_2...i_{10}\}$ is a frequent itemset and $\{i_1\}$ is infrequent. In Fig. 3a, Boolean vectors are arranged on each disk in their numeric order. The visualization does not permit any real understanding of the border of frequent itemsets. Fig. 3b rearranges Boolean vectors on each disk so that all vectors on a Hansel chain are aligned vertically. A chain is a sequence of Boolean vectors such that each vector produces the next vector by changing a "0" element to "1," and Hansel chains provide a way to non-repeatedly visit all vectors. As the function is monotone on each chain, Fig. 3b does show a border, although the border is highly zigzagged. Figure 3c rearranges Hansel chains according to the position of the first "1" value within each chain and shows the border in a clearer way.

Frequent itemsets and association rules are often presented in a set-theoretic view. Given a set $I = \{i_1, i_2,...,i_k\}$ of items, its power set $\mathcal{P}(I)$ forms a lattice $<\mathcal{P}(I), \subseteq>$ where the subset relationship $\subseteq$ specifies the partial order. For an example set of items, $I = \{a, b, c, d\}$, this lattice can be illustrated in Fig. 4a. Frequent



Visual Association Rules. Figure 3. Visualizing the anti-monotone Boolean function $f(i_1,i_2,...,i_{10}) = \neg i_1$: (Courtesy of [6].)

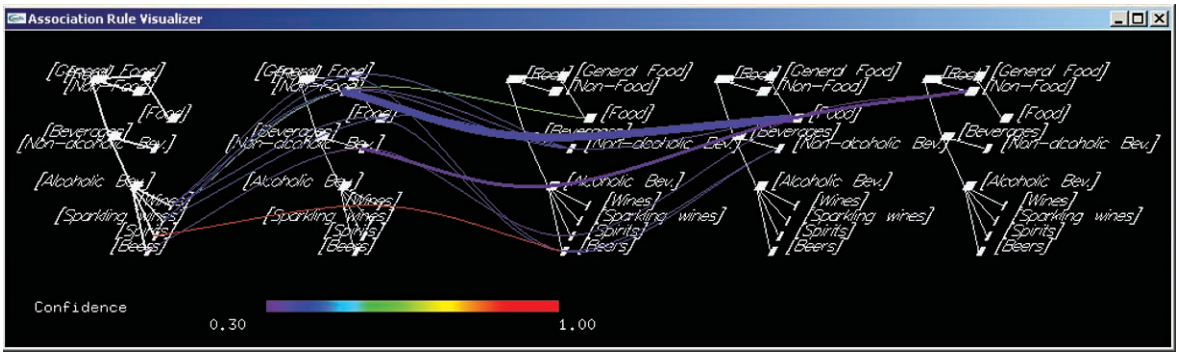


Visual Association Rules. Figure 4. Itemset lattice, item taxonomy tree, and generalized itemset lattice on $I = \{a, b, c, d\}$: (a) Itemset lattice, (b) An item taxonomy tree, and (c) Generalized itemset lattice. (Courtesy of [10].)

itemsets define a border on the lattice: if an itemset is frequent, so is every subset of it; if an itemset is infrequent, so is every superset of it. The objective of frequent itemset visualization is to visualize the border. In fact, this has been tried in the three visualizations in Fig. 3, which have visualized the whole lattice structure. Each disk in Fig. 3 displays $\binom{k}{l}$ itemsets where l is the disk level. Clearly, such an approach would fail when k is large.

In order to deal with the problem of long border of frequent itemsets, Yang has developed an approach [10] for visualizing generalized frequent itemsets and association rules by introducing user interaction with the border. Generalized association rules come from the introduction of item taxonomy. As shown in Fig. 4b, an item taxonomy is a directed tree whose leaf nodes are items and whose non-leaf nodes are item categories. Mining generalized association rules across item taxonomy was studied in [8]. Item taxonomy introduces another dimension of closure property. For example, an ancestor itemset of a frequent itemset is also frequent.

The presence of item taxonomy extends an itemset lattice to a generalized itemset lattice. The user can choose which items or item categories to display in an item taxonomy tree. The partially displayed item taxonomy tree induces a border of displayable itemsets in the generalized itemset lattice. For example, if the



Visual Association Rules. Figure 5. Visualizing association rules with item taxonomy. (Courtesy of [10].)

items *c, d, e, f* in Fig. 4b are displayed, this induces a border of displayable itemsets shown in Fig. 4c. Only non-redundant displayable frequent itemsets, for example, *ec* and *ed* in Fig. 4c, are visualized. In association rule visualization, only non-redundant rules derived from frequent displayable itemsets are visualized.

The non-redundant displayable frequent itemsets and association rules are visualized in 3D through parallel coordinates, where each parallel coordinate is replaced by a standing visualization of item taxonomy tree. An itemset or rule is visualized as a Bézier curve connecting all items in it. Parameters such as support and confidence can be mapped to graphical features such as width or color of the curve. Figure 5 gives a screen snapshot in the interactive visualization of many-to-many association rules, where association rules are aligned according to where the RHSs separate from the LHSs. In Fig. 5, the left two coordinates represent the LHSs of the rules and the right three coordinates represent the RHSs of the rules. The displayed item taxonomy tree can be expanded or shrunk by user interaction, and each change triggers a new set of itemsets or rules to be displayed.

Key Applications

As a standard feature in many data mining software packages, association rule visualization has been used widely in business intelligence and scientific research.

Future Directions

Although approaches have been developed to visualize association rules, none of them has gained predominant acceptance and can simultaneously manage a large number of rules with multiple items on both sides. Visual association rule research has many open problems. An obvious one is how to prune uninteresting itemsets and rules, a topic that has bothered researchers for years. Specific to visualization, open problems include how to visualize a large number of many-to-many rules and how to associate closure properties of itemsets or rules with intuitive visual presentations. Data visualization on the lattice structure has potential applications beyond visual association rules. For example, it can be used in the visualization of iceberg data cubes in data warehousing.

Cross-references

- ▶ Association Rules
- ▶ Data Visualization
- ▶ Visual Analytics
- ▶ Visual Data Mining

Recommended Reading

1. Agrawal R. and Srikant R. Fast algorithms for mining association rules. In Proc. 20th Int. Conf. on Very Large Data Bases, 1994, pp. 207–216.
2. Fukuda T., Morimoto Y., Morishita S., and Tokuyama T. Data mining using two-dimensional optimized association rules: Scheme, algorithms, and visualization. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1996, pp. 13–23.
3. Han J. and Kamber M. Data Mining: Concepts and Techniques. Morgan Kaufmann, San Francisco, CA, USA, 2nd edn., 2005.
4. Hofmann H., Siebes A., and Wilhelm A. Visualizing association rules with interactive mosaic plots. In Proc. 6th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, 2000, pp. 227–235.
5. Klemettinen M., Mannila H., Ronkainen P., Toivonen H., and Verkamo I. Finding interesting rules from large sets of discovered association rules. In Proc. Int. Conf. on Information and Knowledge Management, 1994, pp. 401–407.
6. Kovalerchuk B. and Delizy F. Visual data mining using monotone Boolean functions. In Visual and Spatial

Analysis: Advances in Data Mining, Reasoning, and Problem Solving, B. Kovalerchuk, J. Schwing (eds.). Springer, Berlin, 2004, pp. 387–406.
7. Liu B., Hsu W., Wang K., and Chen S. Visually aided exploration of interesting association rules. In Advances in Knowledge Discovery and Data Mining, 3rd Pacific-Asia Conf., 1999, pp. 380–389.
8. Srikant R. and Agrawal R. Mining generalized association rules. In Proc. 21th Int. Conf. on Very Large Data Bases, 1995, pp. 407–419.
9. Wong P.C., Whitney P., and Thomas J. Visualizing association rules for text mining. In Proc. IEEE Symp. on Information Visualization, 1999, pp. 120–123.
10. Yang L. Pruning and visualizing generalized association rules in parallel coordinates. IEEE Trans. Knowl. and Data Eng., 17 (1):60–70, January 2005.

Visual Classification

Mihael Ankerst
Allianz, Munich, Germany

Synonyms

Cooperative classification

Definition

Decision trees have been successfully used for the task of classification. However, state-of the-art algorithms do not incorporate the user in the tree construction process. Through the involvement of the user in the process of classification, he/she can provide domain knowledge to focus the search of the algorithm and gain a deeper understanding of the resulting decision tree. In a cooperative approach, both the user and the computer contribute what they do best: the user specifies the task, focuses the search using his/her domain knowledge and evaluates the (intermediate) results of the algorithm. The computer, on the other hand, automatically creates patterns satisfying the specified user constraints. The cooperative approach is based on a novel visualization technique for multi-dimensional data representing their impurity with respect to their class labels.

Historical Background

The idea of visual classification has been built upon recent progress in the area of information visualization and decision trees.

Many different algorithms for learning decision trees have been developed over the last 20 years. For instance, *CART* [1] was one of the earliest systems which, in particular, incorporates an effective pruning phase. Their so-called minimum cost complexity pruning cuts off branches that have a large number of leaf nodes, yielding just a small reduction of the apparent error. *SLIQ* [2] is a scalable decision tree classifier that can handle both numerical and categorical attributes. In order to make SLIQ scalable for large training sets, special data structures called attribute lists are introduced, which avoid sorting the numerical attributes for each selection of the next split. Furthermore, a greedy algorithm for efficient selection of splits of categorical attributes is presented. An experimental evaluation demonstrates that SLIQ produces decision trees with state-of-the-art accuracy, and tree size with a much better efficiency for large training sets.

Visual representation of data as a basis for the human–computer interface has evolved rapidly in recent years. The increasing amount of available digital information in all kinds of applications has led to the challenge of dealing with both high dimensionality and large amounts of data. [3] gives a comprehensive overview over existing *visualization techniques* for large amounts of multidimensional data, which have no standard mapping into the Cartesian coordinate system.

Foundations

A decision tree classifier constructs a tree in a top-down fashion, performing a greedy search through the very large space of all possible decision trees. At each current node, the attribute that is most useful for the task of classification (with respect to the subset of all training objects having passed all tests on the path to the current node) is selected. Criteria such as the information gain or the gini index have been used to measure the usefulness of an attribute. Domain knowledge about the semantics of the attributes and the attribute values is not considered by the criteria. Note that greedy algorithms for decision tree construction do not allow to backtrack to a previous choice of an attribute when it finally turns out to be suboptimal.

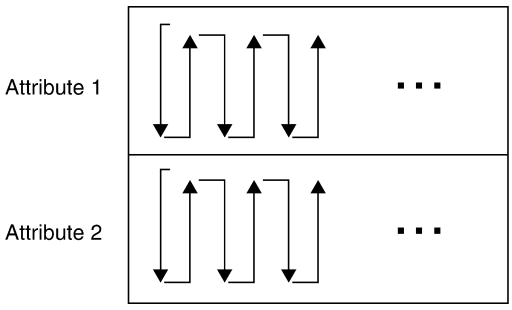
V*isual classification* [4,5] is a user-centered approach to decision tree construction where the user and the computer can both contribute their strengths. The user provides domain knowledge and evaluates intermediate results of the algorithm, the computer

automatically creates patterns satisfying user constraints and generates appropriate visualizations of these patterns. In this cooperative approach, domain knowledge of the user can direct the search of the algorithm. Additionally, by providing adequate data and knowledge visualizations, the pattern recognition capabilities of the human can be used to increase the effectivity of decision tree construction. Furthermore, the user gets a deeper understanding of the decision tree than just obtaining it as a result of an algorithm.

The three components of visual classification is visualizing the data, visualizing the decision tree and the integration of algorithms into decision tree construction.

Visualizing the data is done by the pixel-oriented bar technique. The bar visualization technique is performed as follows. Within a bar, the sorted attribute values are mapped to pixels in a line-by-line fashion according to their order. Each attribute is visualized independently from the other attributes in a separate bar. Figure 1 illustrates the method of the bar visualization for the case of two attributes. The amount of training data that can be visualized by the bar technique is determined by the product of the number of data records and the number of attributes.

Visualizing the decision tree is done by a visualization technique, such that each node is represented by the data visualization of the chosen splitting attribute of that node. For each level of the tree a bar is drawn representing all nodes of this level. The top level bar corresponds to the root node of the decision tree. On lower levels of the tree the number of records and thus the number of pixels is reduced if there are leaves in upper levels – leaves are underlined with a black line. Black vertical lines indicate the split points set in the current bar. On lower levels, partitions of the data inherited from upper levels are marked by white vertical lines at the same horizontal position as the original split point. Attribute and node information at the mouse pointer position (attribute name, attribute value, min., max. value and number of records in this node) is displayed on demand. Upon a mouse click the system switches back to the data visualization of the corresponding node.



Visual Classification. Figure 1. Illustration of bar visualization.

Compared to a standard visualization of a decision tree, a lot of additional information is provided which is very helpful in explaining and analyzing the decision tree:

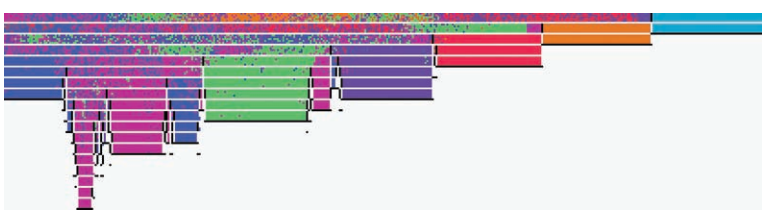
- Size of the node (number of training records corresponding to the node)
- Quality of the split (purity of the resulting partitions)
- Class distribution (frequency and location of the training instances of all classes)

Figure 2 illustrates the visualization of a decision tree for the Segment training data from the Statlog benchmark [6] having 19 numerical attributes.

The integration of algorithms into decision tree construction is solved in the following way.

Propose Split

For a set of attributes selected by the user, the attribute with the best split together with the optimum split point of this attribute is calculated and visualized. If a singleton attribute is specified as input, only the optimum split point for this attribute is determined. The function *propose split* turns out to be useful in two



Visual Classification. Figure 2. Illustration of knowledeg visualization.

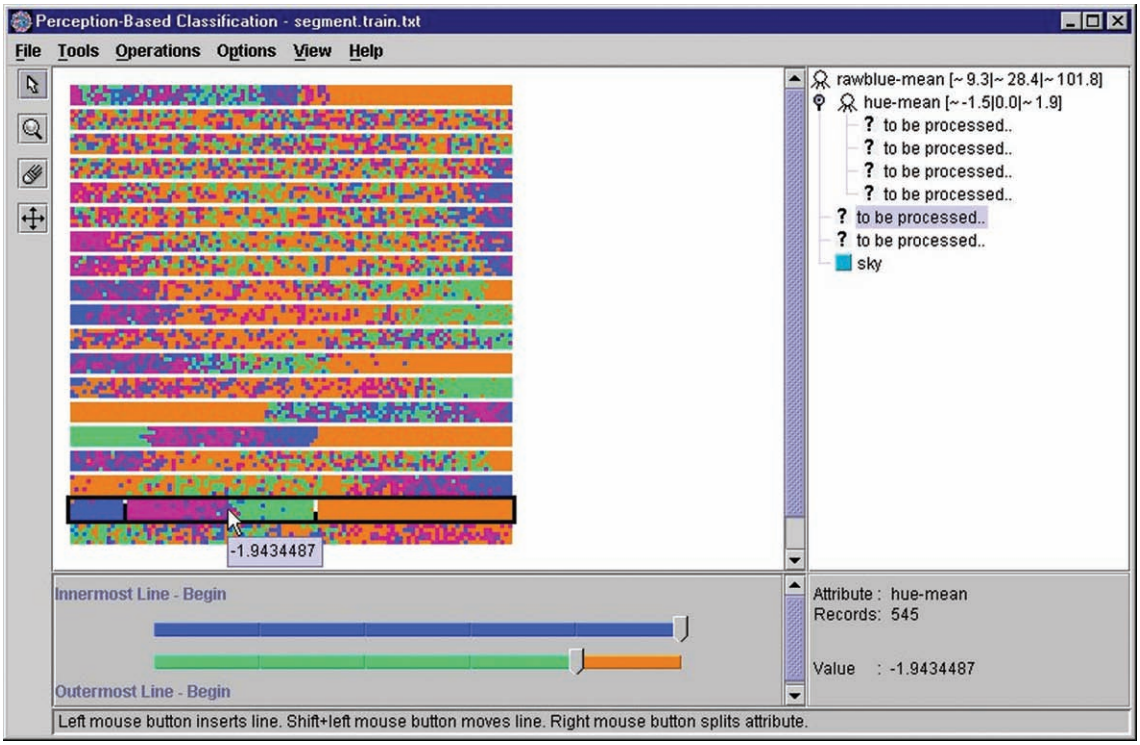
cases: first, whenever there are several candidate attributes with very similar class distributions and, second, when none of the attributes yields a good split which can be perceived from the visualization.

Look-Ahead

For some hypothetical split of the active node of the decision tree, the subtree of a given maximum depth is calculated and visualized with the new visualization technique for decision trees. This function offers a view on the hypothetical expansion up to a user specified number of levels, or until a user specified minimum number of records per node. If the lookahead function is invoked for a limited number of levels, it is very fast (some seconds of runtime) because no pruning is performed in this case. The *look-ahead* function may provide valuable insights for selecting the next split attribute. Without the look-ahead function, when there are several candidate attributes the user selects one of them, chooses one or several split points and continues the tree construction. If at a later stage the expanded branch does not yield a satisfying subtree, the user will backtrack to the root node of this branch and will remove the previously expanded subtree. He/she will select another candidate attribute, split it and proceed. Utilizing the new function, however, the user requests a look-ahead for each candidate attribute before actually performing a split. Thus, the necessity of backtracking may be avoided.

Expand Subtree

For the active node of the decision tree, the algorithm automatically expands the tree. Several parameters may be provided to restrict the algorithmic expansion such as the maximum number of levels and the minimum number of data records or the minimum purity per leaf node. The pruning of automatically created subtrees rises some questions. Should one only prune the subtree created automatically or prune the whole tree – including the subtrees created manually? According to the paradigm of the user as a supervisor, pruning is only applied to automatically created trees. It is important to distinguish two different uses of the *expand subtree* function: the maximum number of levels is either specified or it is unspecified. In the first case, the decision tree is usually post-processed by the user. No pruning is performed if a maximum number of levels is specified. Otherwise, if no maximum



Visual Classification. Figure 3. PBC system.

number of levels is specified, the user wants the system to complete this subtree for him and pruning of the automatically created subtree is performed if desired. The function *expand subtree* is useful in particular if the number of records of the active node is relatively small. Furthermore, this function can save a lot of user time because the manual creation of a subtree may take much more time than the automatic creation.

The visual classification approach is implemented in the PBC system, as depicted in Fig. 3.

Key Applications

The visual classification approach can be applied to any domain where decision tree classification is applicable. The value of the approach is the deeper understanding of the data and the decision tree.

Cross-references

- ▶ Decision Trees
- ▶ Pixel-Oriented Visualization Techniques

Recommended Reading

1. Ankerst M., Elsen C., Ester M., and Kriegel H.-P. Visual classification: an interactive approach to decision tree construction. In Proc. 5th Int. Conf. on Knowledge Discovery and Data Mining, 1999, pp. 392–396.
2. Ankerst M., Ester M., and Kriegel H.P. Towards an effective cooperation of the computer and the user for classification. ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, 2000.
3. Breiman L., Friedman J.H., Olshen R.A., and Stone P.J. Classification and Reggression Trees. Wadsworth, Belmont, CA, 1984.
4. Keim D.A. Visual database exploration techniques. Tutorial at Int. Conf. on Knowledge Discovery and Data Mining, 1997.
5. Mehta M., Agrawal R., and Rissanen J. SLIQ: A Fast Scalable Classifier for Data Mining. In Advances in Database Technology, Proc. 5th Int. Conf. on Extending Database Technology, 1996.
6. Michie D., Spiegelhalter D.J., and Taylor C.C. Machine Learning, Neural and Statistical Classification. Ellis Horwood, 1994. See also http://www.ncc.up.pt/liacc/ML/statlog/datasets.html.

Visual Clustering

Mike Sips
Stanford University, Stanford, CA, USA

Synonyms

Visual mining; Visual data mining

Definition

Synthesis of computational methods and interactive visualization techniques that represents a clustering structure, defined in higher dimensions to the human analyst in order to support the human analyst to explore and refine the clustering structure of high dimensional data spaces based on his/her domain knowledge.

Historical Background

The advancements made in computing technology over the last two decades allow both scientific and business applications to produce large data sets with increasing complexity and dimensionality. Automated clustering algorithms are indispensable for analyzing large n-dimensional data sets but often fall short to provide completely satisfactory results in terms of quality, meaningfulness, and relevance of the revealed clusters. With the increasing graphics capabilities of the available computers, researchers realized that an integration of the human into the clustering process based on visual feedbacks helps to improve the effectiveness of automated clustering algorithms. A synthesis of automated clustering algorithm and visualization usually does not only yield better clustering results, but also a higher degree of user satisfaction and confidence in the findings.

Foundations

Clustering is one of the basic data analysis tasks. It is a process of organizing data into similar groups. Unlike classification, clustering is unsupervised learning. In particular, the classes are unknown and no training set with pre-computed class labels is available. A clustering algorithm in general can be seen as an external source that labels the data.

Automated clustering algorithms are indispensable in analyzing large n-dimensional data spaces, but they often fall short in computing completely satisfactory results. In general, two interconnected reasons can be observed. First, many clustering algorithms use assumptions about specific properties of clusters either as built-in defaults or as input parameter settings. Automated clustering algorithms are very sensitive to these input parameters. Analysts often get different clustering results even for slightly different parameter settings. Useful parameter settings are in general hard to determine a priority because humans have huge difficulties in understanding the impact of different parameter settings on the revealed clusters; especially in n-dimensional data

spaces. Second, large n-D data spaces often have skewed distributions. The density, distribution and shape of the clusters are quite diverse in different subspaces. Skewed distributions are in general difficult to reveal using just one global parameter setting.

A variety of different visual clustering approaches have been proposed. Visual clustering can be classified into four common approaches, based on their mechanism to incorporate the data analyst into the clustering process. Several exploratory data analysis systems allow an interactive exploration of a given clustering structure based on projections. The aim is to make a given clustering structure in n-D interpretable for the human. A few visual clustering approaches are extensions of an optimized clustering algorithm with advanced visualization techniques. Other approaches extract the clustering structure based on the analyst's feedback. In these scenarios, clusters are characterized based on the analyst's specific domain knowledge. More recently, novel data analysis techniques from related disciplines such as machine learning or statistics supporting visual clustering have been proposed. An interesting approach uses visualization to support the selection of subspaces that contain strong clusters. Another interesting approach is Self-Organizing Maps.

Visual Exploration of a Given Clustering Structure

Many interactive systems use orthogonal standard projections to visualize the given clustering structure of an n-dimensional data space. Orthogonal standard projections are widely used in exploratory data analysis, because they are easy to understand. Unlike in unlabeled data, the data item's class labels are exploited in these approaches to find interesting projections of the clustering structure, i.e., projections that preserve properties of the clusters.

The n23Tool proposed by Yang [15] uses 3–D cluster-guided projections [5]. The idea of cluster-guided projections is to find a 3–D subspace that neatly separates four given clusters. A cluster-guided projection is based on the following observation. Any combination of four distinct and non co-linear cluster centers defines a unique 3-D subspace. Moreover, such a subspace neatly separates the four clusters, and in addition it also preservers the inter-cluster distance between any two of the cluster centers. To support an efficient exploration of clusters in n-dimensional data spaces, the n23 tool combines cluster-guided projections with Grand Tour [3]. A cluster-guided tour allows an analyst to quickly get an impression of all clusters by smoothly moving through the space of cluster-guided projections.

Koren and Carmel [11] propose an alternative measure of goodness to preserve the clustering structure in projections. The basic idea is to weight the distances between points differently depending on whether they have the same class label. Given this objective function, the approach then searches for the best linear transformation of the n-dimensional data. This method has a significant advantage in comparison to traditional PCA or MDS, because it captures the cluster structure of the data, and in addition the intra-cluster shapes. One general problem with general projections and embeddings is that users may have trouble interpreting the display. Another problem is that these approaches work well whether the clusters are neatly separated in n–D or the data contains only small amounts of noise.

The Self-Organizing Map (SOM) is a neural network algorithm based on unsupervised learning (see [10] for further readings). The basic idea is to stretch a 1-D or 2-D neuron grid through the data. The lattice of the grid can be either rectangular or hexagonal. Each neuron is represented by an n-dimensional prototype vector. During the iterative training of a SOM, the neurons become the cluster means, and points closest to the neurons are considered to belong to that cluster. Intuitively, a SOM can be seen as a constraint k-means algorithm (the net of neurons is very flexible and folds onto the data clouds).

SOM's can be used to visualize the clustering structure of an n-dimensional data space. The 1-D or 2-D grids get wrapped into a flat 2-D layout [12]. An alternative visual representation of the clustering structure can be achieved by visualizing either the distances of each grid unit to its neighboring grid units or the similarity of grid units. In most cases, gray shading (U-Matrix) and geometric shapes (Distance-Matrix) are used to represent distances to neighboring grid units, and color to show similarity of grid units (similarity coloring). A detailed discussion of SOM-based data visualization techniques is presented in [14].

Extension of an Optimized Clustering Algorithm

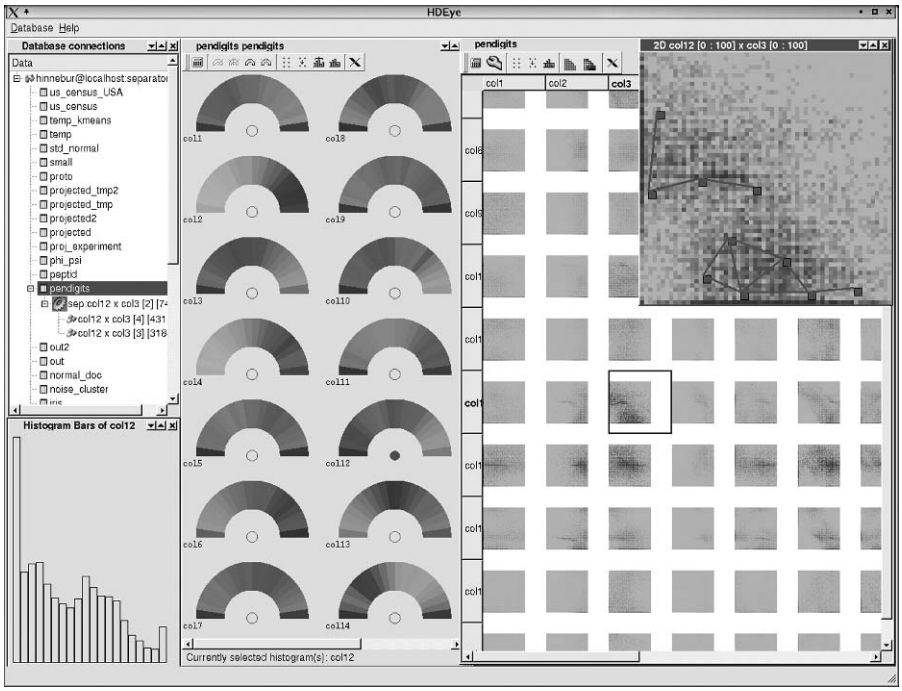
An interesting extension of OptiGrid [8] with both an iconic visualization technique and kernel- density plots is HD-Eye [9]. OptiGrid is a density-based clustering algorithm that uses contracting projections and

separators to build up an n-dimensional grid. Clusters are defined as highly populated grid cells.

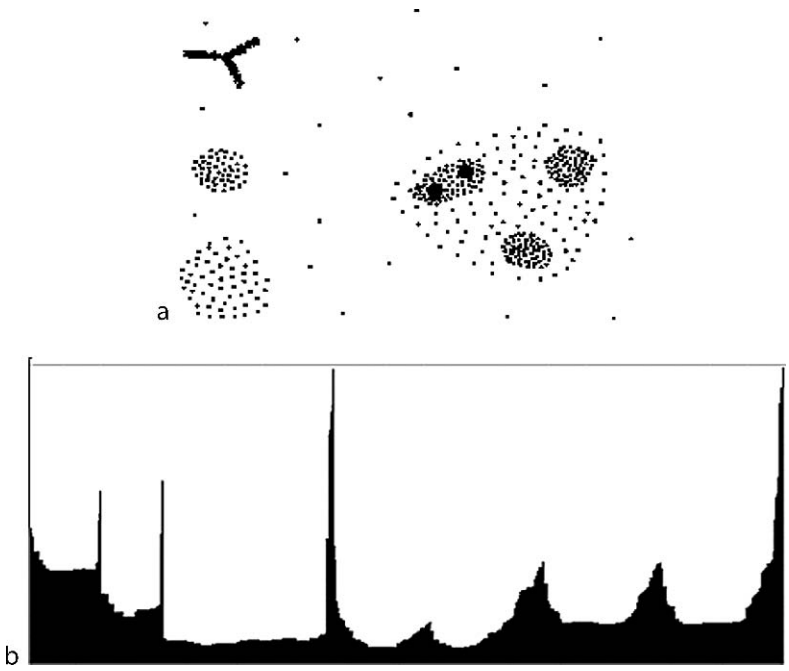
HD-Eye considers clustering as a partitioning problem. A projection is of potential interest in finding cluster separators whether it neatly separates a number of clusters in the projected space. Finding interesting projections and specifying good separators are two difficult problems; both are difficult to compute automatically. The basic idea of HD-Eye is to allow specific user feedbacks in two crucial steps of the partitioning process. To support the user in surveying the vast set of orthogonal projections in order to find potential useful projections, HD-Eye employs an abstract iconic display (Fig. 1). The shapes of the icons are determined based on the number of maxima of the data's density function, and how well the maxima are separated in the projections. The color of the icons represents the number of data items belonging to a maxima. Once a potential interesting projection has been selected, HD-Eye provides 1-D/2-D color-based density plots for a further analysis of the projection. Density plots easily allow the analyst to check whether a projection separates a number of clusters well.

The HD-Eye approach doesn't require that a projection neatly separates all clusters but at least two clusters. HD-Eye follows an iterative partitioning process. Based on his/her domain knowledge and intuition, the analyst interactively selects interesting projections from the iconic display. Then, the analyst defines useful separators within the density plots by drawing either a split line (linear partitions) or a split polygon (non-linear partitions). The analyst can easily follow the interactive partitioning process by inspecting the separator tree. After each interactive partitioning, the clusters get refined, i.e., highly populated regions based on the refined grid, and added into the separator tree.

Another interesting approach is OPTICS [2] (Fig. 2). OPTICS is an extension of the DBSCAN [6] algorithm and creates a one-dimensional ordering of the data items. The ordering of the data items represents its density-based clustering, which can be obtained from a broad range of input parameter



Visual Clustering. Figure 1. The HD-Eye interfaces has three main visual components – the iconic display showing properties of the projections, 1–D and 2–D kernel-density plots for a further analysis of selected projections, and the separator tree. Clockwise starting from the top: separator tree, iconic representation of 1–D projections, 1–D projection histogram, 1–D color-based density plots, iconic representation of multi dimensional projections and color-based 2D density plot. Figure is taken from the exploration of a large molecular biology data set. (used with permission of Alexander Hinneburg).



Visual Clustering. Figure 2. The figure shows an example data (Fig. 2a) and its reachability plot (Fig. 2b) – objects are on the x-axis together with their reachability values on the y-axis. (used with permission of Mihael Ankerst).

settings. Clusters in DBSCAN are determined based on ε and MinPt parameters, when ε is the radius of a local neighborhood around each data item, and MinPt determines the minimal number of data items in the local neighborhood. The basic observation of OPTICS is that given a constant MinPt value, density-based clusters with respect to a higher density (lower ε) are completely contained in density-based clusters with respect to a lower density (greater ε).

The ordering of the data items can be used to visualize how the data items are clustered. These visualizations are called reachability plots. The data items are visualized according to the clustering ordering on x-axis, together with their associated reachability distance on y-axis. Intuitively, data items sharing a common cluster are close to each other in the cluster ordering and have similar reachability distances. The reachability distance jumps whether a different cluster starts in the ordering. In addition, the ordering of the data items can be used to compute a cluster hierarchy (dendrogram).

Clustering Based on Visual User Feedback and Refinement

An interesting approach is presented by Aggarwal [1]. The basic idea is similar to HD-Eye, i.e., the search for projection that neatly separates clusters. In contrast to HD-Eye that uses an iconic display to show properties of the projections and then supports an interactive search for interesting projections, Aggarwal proposed an iterative cluster partitioning based an automated subspace detection algorithm.

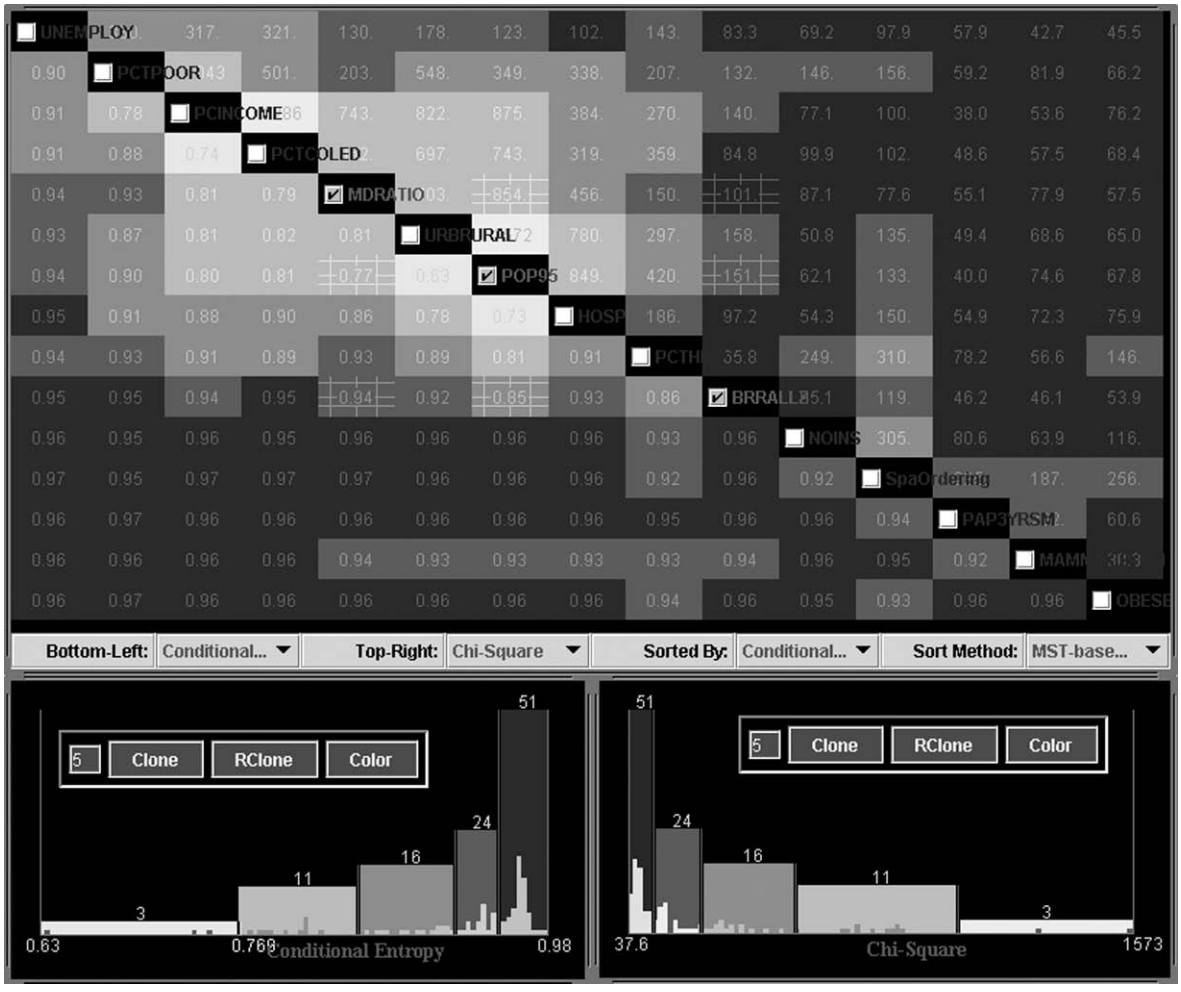
The idea of the algorithm is straightforward and follows an iteration process. In each iteration step, a number of data items called polarization points are randomly chosen from the data set. Next, an analytic method is used to find a lower-dimensional subspace that neatly groups the data items around the polarization points. Once an interesting projection has been found, HD-Eye and Aggarwal use kernel-density plots of the projected data in order to discover the clustering structure. In contrast to the HD-Eye approach that allows an interactive separation of clusters in the density plots, Aggarwal's approach separates clusters by choosing noise levels.

The iterative cluster partition process terminates whether all data items belong at least to one cluster. The final clustering structure is extracted in a frequent item analysis based on the recorded user selections in each partitioning step. To control the granularity of the cluster separation in the final clustering structure, the analyst can choose different noise levels for the same projection.

ClusterSculpture [13] is an interesting approach that allows an analyst to interactively refine an initially computed hierarchical clustering structure. The hierarchical clustering structure is described as a dendrogram. The analyst refines that dendrogram by either splitting or merging a number of nodes; if it's necessary the analyst might want to reorganize the whole dendrogram. ClusterSculpture supports the analyst in deciding whether a node should be split or merged by presenting the relationship between the attribute values and the clustering structure, as well as the distances of neighboring data items around a selected cluster.

ClusterSculpture provides three main visualizations. A dendrogram shows the cluster hierarchy, and it allows the analyst to quickly survey the clustering structure. The distribution of the attribute values in each dimension are shown in a point map. Each data item is visualized as a horizontal line with one colored pixel for each dimension. The color of each pixel represents the normalized attribute value in that dimension. Additionally, the data items are sorted along the y-axis to emphasize the clustering structure. The distances of neighboring data points around the leaf cluster closest to a selected data item is presented when the analyst selects a data item in the point map. Three different colors are used to represent the selected data item, and whether or not data items share the same neighbor with the selected data item.

A number of exploratory data analysis systems have interactive visualizations to support the analyst in finding clusters. GGobi [4] supports the analyst by providing a spin-and-brush mechanism. Starting with



Visual Clustering. Figure 3. The figure shows a visualization of the entropy matrix of a real cancer data set with 72 dimensions. The two histograms at the bottom of the image allows the analyst to adjust the classification and coloring in an interactive fashion. (used with permission of Diansheng Guo).

an initial projection of the data, the analyst tours through the space of possible projections until an interesting projection is reached. The analyst then paints a cluster in that projection using an easy to distinguish color, and continues touring until no more clusters are revealed.

Visualization Used as Interactive Feature Selection for Clustering

An interesting subspace selection method that effectively identifies subspaces determining strong clusters is proposed by Guo [7]. The method is based on the computation of the pairwise conditional entropy values of 2-D subspaces. The pairwise conditional entropy values are visualized in the so-called entropy matrix, which allows the data analyst to easily understand relationships among dimensions (Fig. 3). Interesting multi-dimensional subspaces can be either automatically extracted or visually identified in the entropy matrix. The selected dimensions can be fed into a clustering algorithm in order to improve the effectiveness of the clustering results. Although designed as a feature selection method, the entropy matrix allows the analyst to get an understanding of the overall clustering structure of an n-dimensional data space.

Key Applications

Effective mining of large traditional relational databases, complex 2-D and 3-D multimedia databases, geo-spatial databases, and bio-databases.

Cross-references

- ▶ Automated Clustering Algorithms
- ▶ Exploratory Data Analysis
- ▶ Visual Analytics
- ▶ Visual Data Mining

Recommended Reading

1. Aggarwal C.C. A human–computer interactive method for projected clustering. IEEE Trans. Knowl. Data Eng., 16(4):448–460, 2004.
2. Ankerst M., Breunig M.M., Kriegel H.P., and Sander J. OPTICS: ordering points to identify the clustering structure. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1999, pp. 49–60.
3. Asimov D. The grand tour: a tool for viewing multidimensional data. SIAM J. Sci. Stat. Comp., 6(1):128–143, 1985.
4. Cook D. and Swayne D.F. Interactive and Dynamic Graphics for Data Analysis – With R and GGobi. Springer Science and Business Media, New York, NY, USA, 2007.
5. Dhillon I.S., Modha D.S., and Spangler W.S. Visualizing Class Structure of Multidimensional Data. In Proc. 30th Symp. on the Interface: Computing Science and Statistics, 1998, pp. 488–493.
6. Ester M., Kriegel H.P., Sander J., and Xu X. A density-based algorithm for discovering clusters in large spatial databases with noise. In Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining, 1996, pp. 226–231.
7. Guo D. Coordinating computational and visual approaches for interactive feature selection and multivariate clustering. Inform. Vis., 2(4):232–246, 2003.
8. Hinneburg A. and Keim D.A. Optimal grid-clustering: towards breaking the curse of dimensionality in high-dimensional clustering. In Proc. 25th Int. Conf. on Very Large Data Bases, 1999, pp. 506–517.
9. Hinneburg A., Keim D.A., and Wawryniuk M. HD-Eye: visual mining of high-dimensional data. IEEE Computer Graphics and Applications, 19(5):22–31, 1999.
10. Kohonen T. Self-Organizing Maps. third edn., Springer Series in Information Science, 2001.
11. Koren Y. and Carmel L. Robust Linear Dimensionality Reduction. IEEE Trans. Vis. Comput. Graph., 10(4):459–470, 2004.
12. Kraaijveld M., Mao J., and Jain A. A nonlinear projection method based on kohonen's topology preserving maps. IEEE Trans. Neural Networks, 6(3):548–559, 1995.
13. Nam E.J., Han Y., Mueller K., Zelenyuk A., and Imre D. ClusterSculptor: A Visual Analytics Tool for High-Dimensional Data. In IEEE Symp. on Visual Analytics Science and Technology, 2007, pp. 75–82.
14. Vesanto J. Som-Based Data Visualization Methods. Intell. Data Analy., 2(3), 1999.
15. Yang L. Interactive exploration of very large relational datasets through 3D dynamic projections. In Proc. 6th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, 2000, pp. 236–243.

Visual Content Analysis

Marcel Worring, Cees Snoek
University of Amsterdam, Amsterdam, The Netherlands

Synonyms

Video indexing; Image indexing

Definition

Visual content analysis is the process of deriving meaningful descriptors for image and video data. These descriptors are the basis for searching large image and video collections. In practice, before the process starts, one applies image processing techniques which

take the visual data, apply an operator, and return other visual data with less noise or specific characteristics of the visual data emphasized. The analysis considered in this contribution starts from here, ultimately aiming at semantic descriptors.

Historical Background

Analyzing the content of visual data using computers has a long history, dating back to the 1960s. Some initial successes prompted researchers in the 1970s to predict that the problem of understanding visual material would soon be solved completely. However, the research in the 80s showed that these predictions were far too optimistic. Even now, understanding visual data is still a major challenge.

In the 90s a new field emerged, namely content-based image retrieval (CBIR), where the aim is to develop methods for searching in large image archives. Where the computer vision community was publishing papers on the analysis of 10–50 images, the CBIR researchers started with 1,000 images or more. Nowadays, research papers consider data sets which are orders of magnitudes larger in size. The paper by Smeulders et al. [10] reviewed the state-of-the-art in 2000, almost all of the findings are still very relevant for current research.

The CBIR field has not only developed itself since then, it also had a major impact on the computer vision field. Large visual collections are also becoming the standard in computer vision. It also led to new methods for analyzing visual data. In early methods for understanding the content of the data pre-defined models were used. In current research, models are learned automatically from large sets of annotated examples, based on the appearance of objects in the visual data. This makes the computer vision algorithms more flexible and easier to adapt to new data sets.

Video content analysis started later than image analysis due to the high processing power required. Computer capacity, however, has increased substantially over the last 10 years making it feasible to process also large collections of video. Research in video content analysis started with camera shot segmentation and simple analysis. Soon the research community realized that the result of automatic speech recognition is an important, sometimes decisive, factor in video content analysis. Best results are obtained if the visual and audio modalities are analyzed in conjunction [11]. Relying on speech is fine for video programs with a clear speech signal, nicely separated from environmental sounds and music. More and more, however, people create video data without such clear speech, so the role of visual content has moved to the forefront in video content analysis again.

So, although visual content analysis started around 50 years ago, it is still an active research area with many research challenges ahead.

Foundations

Automatically extracting the semantics of visual data is a notoriously difficult problem which has become known as the semantic gap [10]:

Semantic gap: The semantic gap is the lack of coincidence between the information that one can extract from the sensory data, and the interpretation that the same data has for a user in a given situation.

The existence of the gap has various causes. One reason is that different users interpret the same visual data in a different way. This is especially true when the user is making subjective interpretations of the visual data. In the following part those subjective interpretations are not considered. However, also for objective interpretations like the presence of a car in a picture, developing automatic methods is still difficult. Consequently, their performance is not optimal. The difficulties are due to the large variations in appearance of visual data corresponding to one semantic concept. Cars, for example, come in different models, shapes, and color. The above causes are inherent to the problem. Visual analysis methods should address these fundamental problems.

There are also variations in appearance which are not due to the richness of the semantics. Varying the viewpoint, lighting and other circumstantial conditions in the recording of a scene will deliver different data, whereas the semantics have not changed. These variations induce the so called sensory gap, which is defined as [10]:

Sensory gap: The sensory gap is the gap between the object in the world and the information in an image recording of that scene.

One of the consequences of the semantic and sensory gap is the fact that recordings of semantically different objects might appear more similar than two recordings of the same object in different recording conditions. Methods are needed that are minimally affected by the sensory gap, while still being able to distinguish objects with different semantics. This is the

essence of visual content analysis. General methods for doing the analysis are described next.

Methods for deriving the semantics of visual data are mostly based on a supervised learning scheme. In such a scheme, the developer provides the system with a large set of annotated examples. For all of the examples, the system extracts low-level features. The system then uses a machine learner to build a model for each semantic concept. Now, if a new image or video without annotation is given, the system again computes features. It then uses the model to derive a measure of the probability of the concept being present in the visual data. An overview of the general concept based video content analysis scheme is depicted in Fig. 1.

So, the two main things to consider are the features to extract and the machine learning scheme.

Features are many, see [10,11] for an extensive overview. A good starting point for a set of practical visual features is defined in the MPEG-7 standard [8]. There are different categories of features namely color features, texture features, shape features, and motion features, which will now be described.

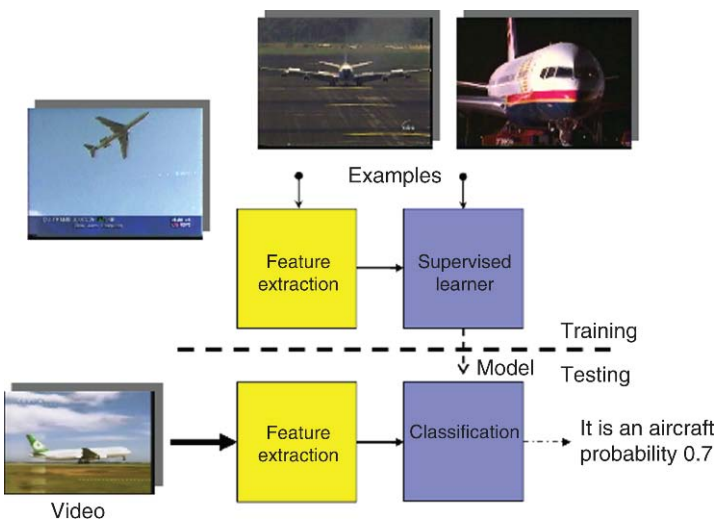
Color features in their simplest form are global descriptions of the visual content based on color histograms. There are numerous color spaces to choose from in which to compute the histogram, including standard *RGB*, the intuitive *HSV* space, the perceptually uniform *Lab* space, or the invariant set of color spaces in [4]. Methods which use more elaborate methods than histograms take the color layout into account, in addition to global characteristics.

Where color is a property which can be measured at every pixel in the visual data, texture is a measure which considers patterns in the image. Again, there is a variety of methods to choose from with varying characteristics in terms of invariance. Many can be traced back to statistics on co-occurrence of grey values in a local neighborhood, or filters emphasizing patterns in certain directions like Gabor filters. In [3], a convenient set of features for texture description has been defined based on natural statistics.

Color and texture often form the basis for segmenting the visual data in homogeneous regions. This process is known as weak segmentation as it is purely based on the visual data itself, not on the interpretation of the data. The latter is known as strong segmentation, which is delineating the contour of a semantic object in the image.

Strong segmentation is required before shape features can be measured. As a consequence, shape features are only applicable for those applications where the background can be controlled easily, like the recognition of logos or a set of objects on a turntable. A distinction is made here between descriptors that describe the object as a region, e.g., based on properties of the medial axis, and measures that take the contour as the basis, like the curvature scale space descriptor.

In most practical cases, strong segmentation is impossible, whereas global descriptors loose too much of



Visual Content Analysis. Figure 1. General scheme for concept-based learning in visual data using annotated examples.

the spatial information. Some methods therefore focus on weak segmentation, describing the resulting regions individually, or move to the detection of key points. These key points are characteristic points on the contours of objects, which in many cases capture the most important information of the object [7].

All of the above features are equally important for video data. However, for video one can also make use of the temporal dimension. One can analyze the motion pattern to find the camera motion, track regions or points of interest through the sequence and describe their tracks, or derive general measures of the activity in the scene. These additional information sources can enhance the interpretation of the data.

Feature extraction yields a feature vector for every image or video segment. As indicated, for interpretation of the data, state-of-the-art systems follow a supervised learning approach.

A good overview of learning methods is presented in [5]. Although there is a large variety of methods, the Support Vector Machine has become the standard choice in most visual indexing schemes [1,14].

The above considers the visual data in isolation, but often the data has associated text data like the automatic speech recognition results for video. This data can be used as an additional source to improve the understanding of the data. In fact, for some concepts the textual data gives almost all the clues to interpret the visual data, others perform best when only the visual data is used. Relations between the concepts detected can also provide essential information for some concepts. Therefore, a method like [13], which learns for a given concept the optimal analysis scheme through appropriate analysis of the data, is the best choice.

Key Applications

Video content analysis plays a role in various application areas, the major ones are considered here.

Broadcasting

The broadcasting field generates enormous amounts of audio-visual data, which should be stored for later reference and re-use. Manual annotation is laborious and expensive. Tools for automatic analysis are of great importance to deal with this flood of videos.

Narrow Casting

Producing video, up to now, has been limited to major film studios or specialized companies. Now, every organization, association or individual can create video and put it on the web. Even more than for broadcasting companies, narrow casters have limited resources to manually analyze the content of their videos.

Law Enforcement

Through the availability of cheap cameras, video content analysis plays a major role in law enforcement. There is a need for content analysis to detect and analyze illicit video material on confiscated hard disks containing hundreds of gigabytes of video data. In video surveillance, a single camera generates 7*24 h of video per week. Any visual content analysis which helps the operator in reducing the load of watching these video streams is of great importance.

Web Search

There is no need to explain that there is a tremendous amount of video available on the Internet. Video search on the web, however, is still mostly limited to keyword-based search for full video clips. Video content analysis is needed to delve into the actual content of all these videos out there.

Future Directions

Obviously the research in video content analysis has not reached its end yet. On the contrary, the community is only beginning to automatically understand the meaning of visual material. To bring the field a step further, researchers should tackle the large number of open questions out there.

Tangential contributions will introduce features with higher discriminatory power, while having better invariant properties, and new machine learning techniques, better suited for the specifics of visual data.

A new direction is tackling the major bottleneck in supervised learning, namely the amount of training examples needed, by employing user tagged visual data like Flickr. The latter annotations are less accurate than the current practice in supervised learning, but the amount of training samples is orders of magnitude larger.

Another promising direction is the use of capture time meta data, contextual information, and social networks to provide additional clues for interpreting the visual data.

Data Sets

The research community in video content analysis is very active. In recent years there was an important

role for benchmarks encouraging research in the various subfields. They do so by providing large test collections and uniform evaluation procedures. The benchmarks provide a forum where researchers can compare their results, bringing the research a big step further every year.

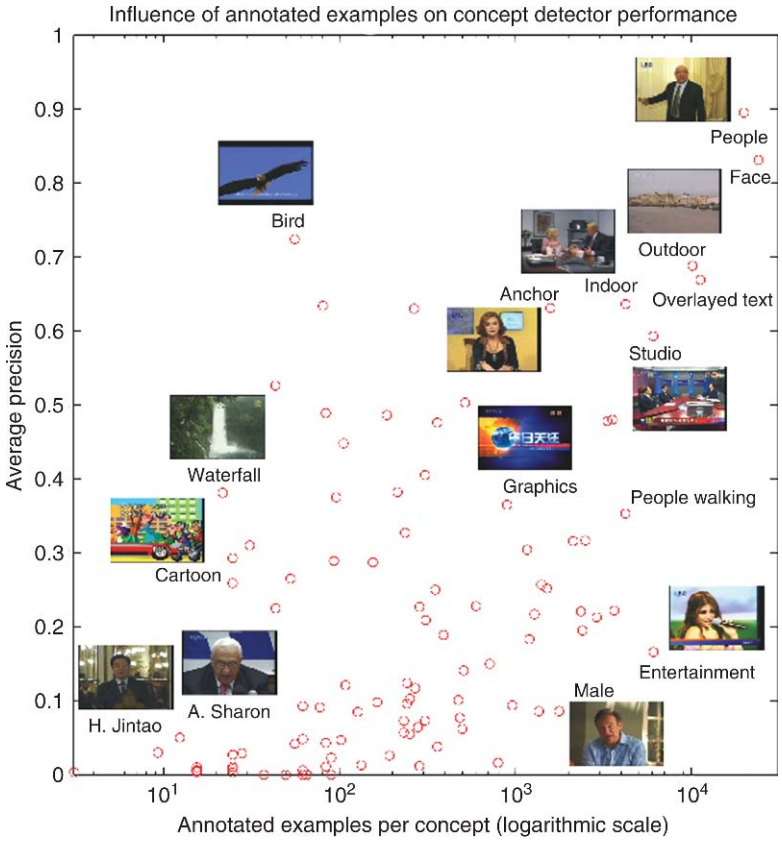
For visual content analysis of images the most elaborate benchmark is the PASCAL VOC challenge [2]. This benchmark contains image sets of several hundreds to thousands of images with a ground truth at the semantic level, a box containing the object, as well as complete outlines of the object in the scene. In this manner, it provides a challenging data set for testing.

Video analysis, especially for the purpose of search, has been studied for several years now in the TRECVID benchmark organized by NIST. In this benchmark, videos from mostly news and other broadcasts have been semantically annotated at the shot level [9].

Collections contain hundreds of hours of video material. Hence they not only pose a challenge at the analysis level, but also at the computer performance level. In addition, one can see that the analysis of such data sets requires expertise from different fields, making the threshold to work on such collections large. Therefore, the MediaMill Challenge [12] has taken the idea of making shared data sets available one step further. This resource not only contains annotations of the TRECVID data, but also a collection of five pre-cooked experiments. To allow an easy start on the topic, the results of each sub-step are provided. Later Columbia and the VIREO group made similar resources available [6,15]. The above efforts allow for visual database research without actually having to perform the visual analysis.

Experimental Results

The performance of video analysis algorithms for information retrieval is typically measured by the average precision. This is a single-valued measure proportional to the area under a recall-precision curve. This value is the average of the precision over all relevant judged



Visual Content Analysis. Figure 2. Indication of the state-of-the-art performance in video content analysis. Note the clear relation between the performance and the amount of annotated examples.

shots. Hence, it combines precision and recall into one performance value. To be precise, let $L^k = \{l_1, l_2, ..., l_k\}$ be a ranked version of the answer set A. At any given rank k let $R \cap L^k$ be the number of relevant results in the top k of L, where R is the total number of relevant results. Then average precision is defined by:

$$average\ precision = \frac{1}{R}\sum_{k=1}^{A}\frac{R \cap L^k}{k}\lambda(l_k), \qquad (1)$$

where indicator function $\lambda(l_k) = 1$ if $l_k \in R$ and 0 otherwise. As the denominator k and the value of $\lambda(l_k)$ are dominant in determining average precision, it can be understood that this metric favors highly ranked relevant results. To give an indication of the state-of-the-art, Fig. 2 gives an overview of results for TRECVID data using the features defined for the MediaMill challenge. Performance is shown as function of number of annotated examples as this has proven to be an important indicator for the quality of the result.

One can conclude that performance is not perfect, but a rapid increase in performance over the last years can be observed. Hence, one could expect that more and more reliable concept detectors will become available in the coming years.

Cross-references

- ▶ Content-Based Video Retrieval
- ▶ Decision Tree Classification
- ▶ Image Retrieval
- ▶ Multimedia Databases
- ▶ Multimedia Information Retrieval
- ▶ Video Scene and Event Detection
- ▶ Video Content Analysis

Recommended Reading

1. Chang C.C. and Lin C.J. LIBSVM: a library for support vector machines. 2001, software available at: http://www.csie.ntu.edu.tw/~cjlin/libsvm/.
2. Everingham M., van Gool L., Williams C., Winn J., and Zisserman A. The PASCAL visual object classes homepage. Available at: http://www.pascal-network.org/challenges/VOC/.
3. Gemert J., Geusebroek J., Veenman C., Snoek C., and Smeulders A. Robust scene categorization by learning image statistics in context. In Proc. Int. Workshop on Semantic Learning Applications in Multimedia, 2006.
4. Geusebroek J., Boomgaard R., Smeulders A., and Geerts H. Color invariance. IEEE Trans. Pattern Anal. Mach. Intell., 23(12):1338–1350, 2001.
5. Jain A., Duin R., and Mao J. Statistical pattern recognition: A review. IEEE Trans. Pattern Anal. Mach. Intell., 22(1):4–37, 2000.
6. Jiang Y.G., Ngo C.W., and Yang J. VIREO-374: LSCOM semantic concept detectors using local keypoint features. Available at: http://www.cs.cityu.edu.hk/~yjiang/vireo374/.
7. Lowe D.G. Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vis., 60(2):91–110, 2004.
8. Sikora T. The MPEG-7 visual standard for content description-an overview. IEEE Trans. Circ. Syst. Video Tech., 11:696–702, 2001.
9. Smeaton A. Large scale evaluations of multimedia information retrieval: The TRECVid experience. In Proc. 4th Int. Conf. Image and Video Retrieval, 2005, pp. 19–27.
10. Smeulders A., Worring M., Santini S., Gupta A., and Jain R. Content based image retrieval at the end of the early years. IEEE Trans. Pattern Anal. Mach. Intell., 22(12):1349–1380, 2000.
11. Snoek C. and Worring M. Multimodal video indexing: A review of the state-ofpascal-the-art. Multimed. Tool Appl., 25(1):5–35, 2005.
12. Snoek C., Worring M., van Gemert J.C., Geusebroek J.M., and Smeulders A. The challenge problem for automated detection of 101 semantic concepts in multimedia. In Proc. 14th ACM Int. Conf. on Multimedia, 2006.
13. Snoek C., Worring M., Geusebroek J., Koelma D., Seinstra F., and Smeulders A. The semantic pathfinder: Using an authoring metaphor for generic multimedia indexing. IEEE Trans. Pattern Analy. Mechine Intell., 28(10):1678–1689, 2006.
14. Vapnik V. The nature of statistical learning theory. Springer-Verlag, New York, NY, USA, 2nd edn., 2000.
15. Yanagawa A., Chang S.F., Kennedy L., and Hsu W. Columbia university's baseline detectors for 374 LSCOM semantic visual concepts. Columbia University, 2007, aDVENT technical report 222-2006-8.

Visual Data Analysis

- ▶ Visual Analytics
- ▶ Visual Data Mining

Visual Data Exploration

- ▶ Dense Pixel Displays
- ▶ Disclosure Risk

V

Visual Data Mining

Simeon J. Simoff
University of Western Sydney, Sydney, NSW, Australia

Synonyms

Visual data analysis; Visual analysis; Visual discovery; Immersive data mining; VDM

Definition

Visual data mining (VDM) is the process of interaction and analytical reasoning with one or more visual representations of abstract data. The process may lead to the visual discovery of robust patterns in these data or provide some guidance for the application of other data mining and analytics techniques. It facilitates analysts in obtaining deeper understanding of the underlying structures in a data set. The process relies on the tight interconnectedness of tasks, selection of visual representations, the corresponding set of interactive manipulations, and respective analytical techniques. Discovered patterns form the information and knowledge utilized in decision making.

Historical Background

Visual exploration of large data sets had been used as a complementary technique to data mining in order to obtain additional information about the data set. Since the early 1990s there has been recognition of the need for specific visualization techniques for large data sets that enable data mining tasks. The term "visual data mining" emerged in the late 1990s, labeling such techniques combined with interactive functionality and some guidelines for sense making from different visual displays of the data and/or the output of data mining algorithms. Ankerst's and Niggemann's theses around the millennium considered the establishment of more systematic scientific and methodological foundations of the field. The 3D Visual Data Mining (3DVDM) project in Aalborg University added a new twist, placing the analyst literally "within" the visual representation of the data set in various stereoscopic virtual reality systems.

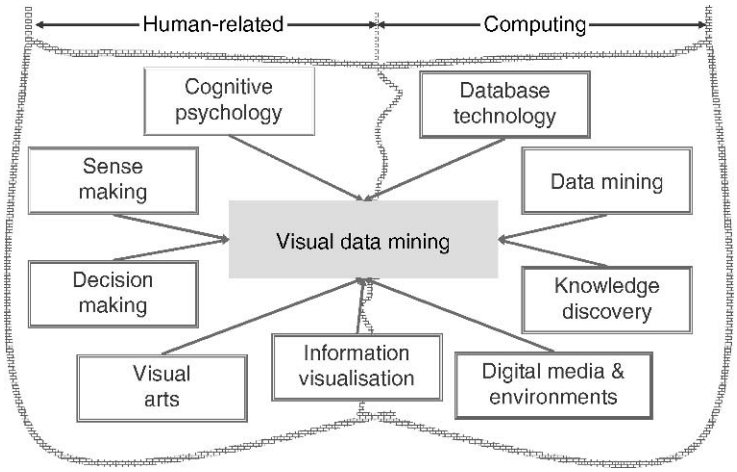
During 2001–2003 a series of workshops on visual data mining, conducted in conjunction with major KDD conferences on both sides of the Atlantic, bridged scholars in the fields of data mining and knowledge discovery in databases, information visualization, human-computer interaction as well as cognitive psychologists related to computing, industry practitioners, and developers of visual data mining tools. They established some common elements that need to be specified when developing methodologies for visual data mining, including: (i) the initial assumptions posed by the respective visual representations; (ii) the set of interactive operations over the respective visual representations and guidelines for their application and interpretation, and; (iii) the range of applicability and limitations of the visual data mining methodology. The majority of the works, published in the proceedings of these workshops after completion, have made their way into the books and special issues, referred in section "Recommended Reading". These and numerous subsequent research and development efforts led to the establishment of visual data mining as an essential component of visual analytics.

Foundations

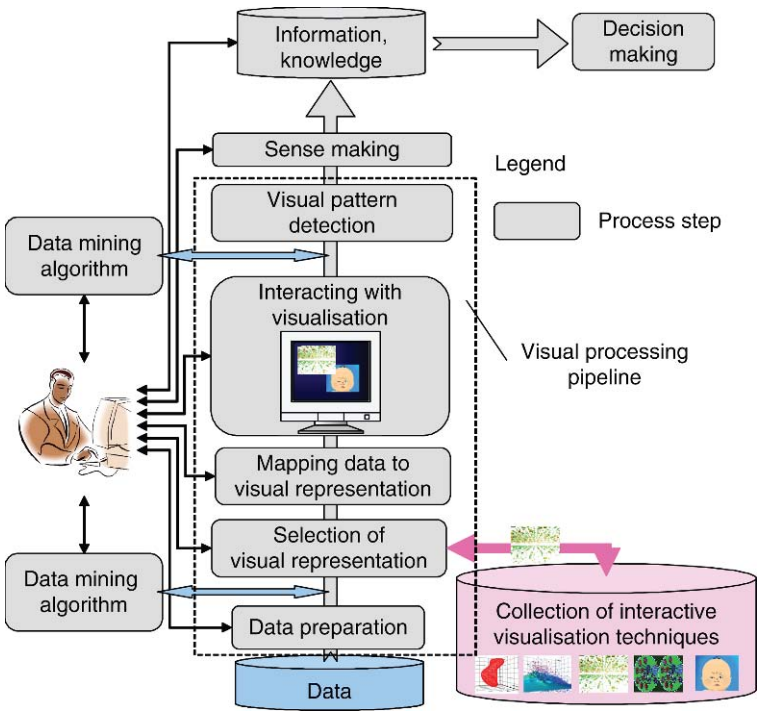
Visual data mining integrates the abilities of the human perceptual system for pattern recognition and human reasoning from images, with the data processing and display power of computers, aiming at capitalizing on the best of both worlds. From a disciplinary perspective, visual data mining emerged as the confluence of several disciplines, the main of which are grouped into two clusters in Fig. 1. On the right-hand side are those, primarily related to computing, when on the left-hand side are those primarily related to humans. From the perspective of scientific fundamentals, though not exhaustive, this list shows the diversity of sources that contribute to the methods and approaches developed in visual data mining.

The VDM process relies on the interactive visual processing pipeline, shown in Fig. 2. Each step within the pipeline involves interaction with the analyst and all the iterative loops in the process close via the analyst. Data mining algorithms can be used to assist the process before any visualization has been considered, and/or after a visual interaction with the data.

Being the core of this human-centred process, visual processing is constrained by the capacity of the human visual system to somewhat 10^6 to 10^7 observations. Large data sets easily surpass this constraint. Consequently, essential for visual data mining are low-dimensional visual representations of high dimensional data that (i) preserve relations between the data points in the high-dimensional space, and (ii) can make such relations visually detectable. Coupling of visualization algorithms with statistical techniques is one way it generating visual data representations that more accurately present the underlying properties of the data. Such visual representations are expected to fit consistently certain metaphors that are close to human mental models, as humans depict structures in the data by forming a mental model which captures only a gist of the information. These are important design considerations in visual data mining, as visualization



Visual Data Mining. Figure 1. Visual data mining as a confluence of disciplines.



Visual Data Mining. Figure 2. Visual data mining is a human-centred interactive analysis and discovery process.

algorithms display data sets that usually lack inherent 2D and 3D semantics.

As manipulation of visual representations of the data set (or subsets of it) is essential in enabling visual pattern detection, the success of the process illustrated in Fig. 2 depends on: (i) the breadth of the collection of visualization techniques, (ii) the consistency of the design of these visual representations, (iii) the ability to remap interactively the data attributes to the parameters of the visual representations, (iv) the set of functions for interacting with the visualization, and (v) the capabilities that these functions offer in support of the sense-making process. From an analyst point of view, the suite of visual representations and interactive techniques is the toolbox for constructing "virtual worlds" from the data. The visual metaphors in such worlds are the vehicles to

transfer meaning from the visual structure of the data into the semantics of the underlying problem. The transfer is enabled by the rules of behavior of the visualization elements that constitute a visual language. The consistent design of visual languages and techniques for interaction with different visual representations of a dataset take into account the specifics of human long-term and short-term memory, knowledge of interaction with virtual environments, including frames of reference, orientation, and navigation. These considerations are also central in order to support the sense-making process in Fig. 2 (the process itself has several iterative steps that are not shown in Fig. 2).

Dealing with large volumes of data means *scalability* and interactivity means *real-time response* of the visualization algorithms in the toolbox in Fig. 2, and the overall visual processing pipeline in order to analyze the details available in the large databases. Interactivity also means support of various visual data mining operations, for example, a visual drill down or visual clustering.

The presentation of the extracted information and knowledge during the decision making step in Fig. 2 usually requires a different visualization approach to the visual data mining step. Such separation of visual analysis tasks from the visual presentation of the results is reflected in contemporary visual data mining technology.

Key Applications

Visual data mining is applied in a wide range of areas of human endeavor, which generate large volumes of otherwise incomprehensible data. These include science, business, technology development, medicine, and healthcare. Chemistry and new drug discovery are examples of successful application of VDM methods. The applications in various fields provide feedback to the development of VDM technology, in terms of integration of 1D, 2D, 2.5D, and 3D visual representations, respective sets of interactive operations, and evaluations of the visual data mining capabilities and interfaces by the analysts in respective fields.

Special key application areas of visual data mining are the analysis of geospatial and spatio-temporal data sets, visual link analysis and visual text mining. Spatial data sets include spatially related attributes, which provides guiding information for the choice of visual representations and interaction methods. Visual link analysis is applied in areas where there is a need for identification of hidden links and chain of links between the attributes in a data set, for example, the fraud detection type of problems in different environments – government, corporate, insurance, retailing, and crime. These problems have the same feature: different cases of fraud may differ in terms of the patterns that reflect them in the data, and the nonvisual data mining and analytics methods may not be able to detect them. The advent of social computing has led to embedding visual link analysis technologies in the social computing systems. Visual text mining uses a variety of statistical techniques to relate groups of words with groups of documents, and then generates visual data summarizations using different metaphors.

Future Directions

Visual data mining, as a discovery mode of interaction with databases, plays a central role in visual analytics. Though interaction has been recognized, there is a strong demand on the development of interactive visualizations, which are fundamentally different from the static visualizations. Designed with the foundations of perceptual and cognitive theory in mind, and focused on supporting the processes and methods in visual data mining, these visual data representations are expected to be relevant to specific tasks and effective in terms of achieving the analytics goals. Within this direction essential are methods for: (i) design and evaluation of visualizations for visual data mining, including metrics for the evaluation of the interactivity of visualizations and their ability to facilitate discovery processes; (ii) visualization techniques of projections of high-dimensional data that preserve the statistical properties of the patterns in the original data; (iii) further integration of data visualization and sonification and development of interactive mining methods, including immersive ones, that operate with such data representations; and (iv) enabling VDM systems to support research strategies and inquiry styles of different data analysts.

Experimental Results

As a rule, research papers in visual data mining, reporting on novel visual analysis techniques, present the results from the application of respective techniques to several data sets. Papers, which explore

human-computer interaction aspects, report the outcomes of observational case studies and usability experiments.

Data Sets

Data sets are available from the KDNuggets site (http://www.kdnuggets.com/datasets/index.html), government bureaus of statistics, research centers and other sources. For visual link analysis the Enron organizational e-mail data set is available at http://www.cs.cmu.edu/~enron/.

URL to Code

This selective rather than inclusive sample of tools is divided into two categories: (i) visual data exploration and mining tools, and (ii) general visualization platforms that can be used to develop visual data mining tools.

Visual Data Exploration and Mining

Open Source Xmdv Tool: Source: http://davis.wpi.edu/~xmdv/.
Offers five methods for displaying flat form data and hierarchically clustered data and variety of interactive operations.

GGobi: http://www.ggobi.org/
Offers similar variety of visualization and interaction methods.

VizRank: http://www.ailab.si:8088/supp/bi-vizrank
Offers informative scatter-plot projections in high dimensional class-labeled data.

3DVDM: http://www.inf.unibz.it/dis/projects/3dvdm/download.html
Offers tools for exploring data in 3D virtual reality environments.

Commercial Miner3D: http://www.miner3d.com/
Highly flexible visual data mining tool with many display and interaction methods.

NetMap Analytics: http://www.netmap.com.au/
Tool for visual link analysis. Separate visual analysis and visual presentation modes (the presentation mode bears some similarity to VisualLinks (http://www.visualanalytics.com)).

NetMiner: http://www.netminer.com/
A nice collection of visual metaphors for interactive analysis of graph structures in data sets.

IN-SPIRE: http://in-spire.pnl.gov/- visual text miner.

General Visualization Engines and Development Tools

Open Source VTK: http://www.vtk.org/
The Visualization ToolKit (VTK) software system for 3D computer graphics, image processing, and visualization.

Mondrian: http://moose.unibe.ch/tools/mondrian
Information visualization engine that lets the visualization be specified via a script, hence, can be utilized for development of specialized visual data mining tools.

ParaView: http://www.paraview.org/New/index.html
Visualization platform that supports distributed computational models for processing large data sets

Gapminder: http://www.gapminder.org
Based on the original Trendalyzer tool for knowledge extraction from interactive animation of statistical time-series, the development has been overtaken by Google.

Cross-references

- ▶ Data Mining
- ▶ Human-Computer Interaction
- ▶ Information Visualization
- ▶ Knowledge Discovery in Data Bases
- ▶ Visual Analytics
- ▶ Visual Clustering
- ▶ Visual Information Processing

Recommended Reading

1. Ankerst M. Visual Data Mining. Faculty of Mathematics and Computer Science, University of Munich, Munich, 2000.
2. Chen C. Information Visualization: Beyond the Horizon. Springer, London, 2004.
3. Chittaro L., Combi C., and Trapasso G. Data mining on temporal data: a visual approach and its clinical application to hemodialysis. J. Visual Lang. Comput., 14:591–620, 2003.
4. Demšar U.K. Investigating visual exploration of geospatial data: an exploratory usability experiment for visual data mining. Comput. Environ. Urban., 31:551–571, 2007.
5. de Oliveira F., Crisina M., and Levkowitz H. From visual data exploration to visual data mining: a survey. IEEE T. Vis. Comput. Gr., 9(3):378–394, 2003.
6. Isenberg P., Tang A., and Carpendale S. An exploratory study of visual information analysis. In Proc. SIGCHI Conf. on Human Facters in Computing Systems, 2008.
7. Keim D.A., Mansmann F., Schneidewind J., and Ziegler H. Challenges in visual data analysis. In Proc. Int. Conf. on Information Visualization, 2006.
8. Keim D.A. and North S.C. Visual data mining in large geospatial point sets. IEEE Comput. Graph., 24(5):36–44, 2004.

9. Keim D.A., Sips M., and Ankerst M. Visual data-mining techniques. In Visualization Handbook, Hansen Johnson Elsevier, Amsterdam, 2005, pp. 831–843.
10. Kovalerchuk B. and Schwing J. (eds.). Visual and Spatial Analysis: Advances in Data Mining, Reasoning, and Problem Solving. Springer, Dordrecht, 2004.
11. Niggemann O. Visual Data Mining of Graph-Based Data. Department of Mathematics and Computer Science, University of Paderborn, Paderborn, Germany, 2001.
12. Shneiderman B. Inventing discovery tools: combining information visualization with data mining, In Proc. Discovery Science, 2001, pp. 17–28.
13. Simoff S.J., Böhlen M., and Mazeika A. (eds.). Visual Data Mining: Theory, Techniques and Tools for Visual Analytics. Springer, Heidelberg, 2008.
14. Soukup T. and Davidson I. Visual Data Mining: Techniques and Tools for Data Visualization and Mining. John Wiley & Sons, London, 2002.
15. Thomas J.J. and Cook K.A. Illuminating the Path: The Research and Development Agenda for Visual Analytics. IEEE CS Press, Silver Spring, MD, 2005.

Visual Discovery

► Visual Data Mining

Visual Displays of Nonnumerical Data

► Visualizing Categorical Data

Visual Displays of Numerical Data

► Visualizing Quantitative Data

Visual Formalisms

David Harel, Shahar Maoz
The Weizmann Institute of Science, Rehovot, Israel

Definition

Visual formalisms are diagrammatic and intuitive, yet mathematically rigorous languages. Thus, despite their clear visual appearance, they come complete with a syntax that determines what is allowed, and semantics that determines what the allowed things mean. The main emphasis in the visuality is typically placed on topological relationships between diagrammatic elements, such as encapsulation, connectedness, and adjacency. Geometric and metric aspects, such as size, shape, line-style, and color, may also be part of the formalism. Icons can be used too. Such languages typically involve boxes and arrows, and are often hierarchical and modular. Visual formalisms are typically used for the design of hardware and software systems. This includes structural as well as more complex behavioral specifications.

Historical Background

Two of the oldest examples of visual formalisms are graphs and Venn diagrams, which are both originally due to Euler [7,8]. A graph, in its most basic form, is simply a set of points, called nodes or vertices, connected by lines, called edges or arcs. Its role is to represent a set of elements S and some binary relation R on them. The precise meaning of the relation R is part of the application and has little to do with the mathematical properties of the graph itself. Certain restrictions on the relation R yield special classes of graphs that are of particular interest, such as ones that are connected, directed, acyclic, planar, or bipartite. Graphs are used intensively in all branches of computer science; the elements represented by the nodes range from the most concrete (e.g., physical gates in a circuit diagram) to the most abstract (e.g., complexity classes in a classification scheme), and the edges have been used to represent many kinds of relations, e.g., logical, temporal, causal, or functional. Many of the fundamental algorithms in computer science are related to graphs. Some open problems are related to graphs too, such as the problem of whether graph isomorphism can be decided in polynomial time.

A hypergraph is a generalization of a graph, where the relation R is not necessarily binary (in fact, it need not have a fixed arity). The corresponding edges are called hyperedges. Hypergraphs have applications in database theory (see, e.g., [9]). The information conveyed by a graph (or a hypergraph) is nonmetric, and is captured by the purely topological notion of connectedness; shapes, locations, distances, and sizes, for example, have no significance.

Euler circles, or Venn diagrams (see [8,25] for the original work), represent sets using the fact that simple closed curves partition the plane into disjoint

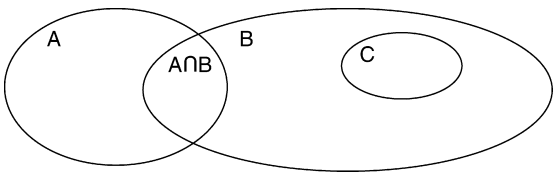
inside and outside regions. They thus give the topological notions of enclosure, exclusion, and intersection the obvious set theoretic meanings of being a subset of, being disjoint from, and having a nonempty intersection with, respectively (see Fig. 1). Typical uses of Venn diagrams include only a small number of circles. However, they can be expanded to *n*-set diagrams (see, e.g., [5]).

A higraph [17] is a formalism that further generalizes graphs, in the spirit of Euler circles. It adds not only hyperedges, but Euler/Venn-like topologically related "blobs" for the nodes, as well as a certain kind of partitioning and adjacency. It thus adds to the relation R notions of orthogonality and hierarchy, as well as multilevel and multinode edges (and hyperedges).

The term "visual formalism" appears to be first used in the work on statecharts [16,17], a language for the specification and design of reactive systems. Higraphs [17] are the underlying graphical representation for statecharts.

In general, visual formalisms provide mechanisms for higher levels of abstraction and thus help engineers in coping with the challenges posed by the ever increasing size and complexity of systems. The effectiveness of visual formalisms also relies on the power of human vision and cognitive perception in recognizing and memorizing visual properties, such as topological/spatial relationships between visual entities. They thus provide an appealing representation and communication medium. The advent of modern tools for graphical editing and display has also contributed to the increasing popularity of various visual formalisms among practitioners. Moreover, some tools, indeed go beyond plain editing and syntax correctness checks, and allow their users to take advantage of the rich semantics of some visual formalisms. Thus, some tools have automated procedures for languages that are sufficiently powerful, such as code generation and formal verification. These are directly applied to the visual artifacts and thus indeed exploit their full potential.

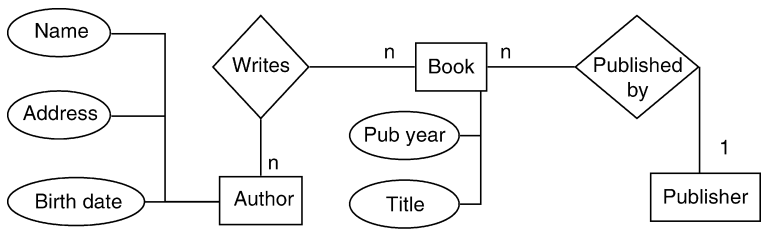
The Unified Modeling Language (UML) [23], an OMG standard formed in the late 1990s, is a collection of visual languages for software and systems specification, defined under a common meta model, and supported by many tools. The standard, however, does not come with satisfactory semantics and hence, by itself, cannot be regarded as a fully fledged visual formalism, though many researchers have provided semantics for many parts thereof.



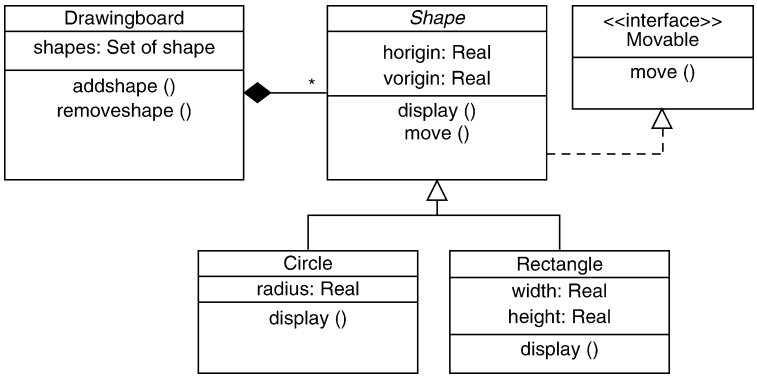
Visual Formalisms. Figure 1. An example of a Venn diagram.

Foundations

Some widely used visual formalisms include flowcharts, used for the representation of series of actions and decisions, mainly for simple imperative programs (see [14] for what seems to be the original usage thereof, and [12] for a fundamental paper on verification, which was carried out in the framework of flowcharts); data flow diagrams (DFD) (see, e.g., [13]), used to represent the flow of data through a system; entity relationship diagrams (ERD) (Fig. 2, and see [2] for the original work) and their modern application to objects, object model diagrams (OMD) and class diagrams (part of the UML [23], see Fig. 3); message sequence charts (MSC) [21] and their modal extension, live sequence charts [3,19], for inter-object



Visual Formalisms. Figure 2. An example of an entity relationship diagram.



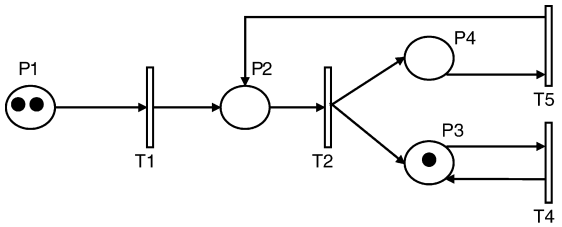
Visual Formalisms. Figure 3. An example of a class diagram.

scenario-based specification; the specification and description language (SDL) [20], for the specification and description of the behavior of reactive and distributed systems; Petri nets (see, e.g., [24], and see Fig. 4); statecharts [16,18] (see Fig. 5); and constraint diagrams [22], used to express logical constraints extended with quantification and navigation of relations.

Statecharts are a prime example of a visual formalism and are therefore shortly discussed below. However, the ideas presented are more general and have indeed been extended to other domains, such as security, databases, and more generally, UML-style behavioral and structural modeling.

In essence, statecharts are finite state machines drawn as state transition diagrams and extended with two key ideas: hierarchy and orthogonality. The hierarchy is introduced using a substating mechanism, and the orthogonality is introduced using a notion of simultaneity. Both are reflected in the graphics themselves, i.e., the hierarchy by encapsulation inspired by Euler/Venn diagrams, and the orthogonality by partitioning blobs into adjacent regions separated by dashed lines.

Topological features are a lot more fundamental than geometric ones, in that topology is a more basic branch of mathematics than geometry in terms of symmetries and mappings. One thing being inside another is more basic than it being smaller or larger than the other, or one being a rectangle and the other a circle. Being connected to something is more basic than being green or yellow or being drawn with a thick line or with a thin line. The brain understands topological features given visually much better than it grasps geometrical ones. The mind can see easily and immediately whether things are connected or not,

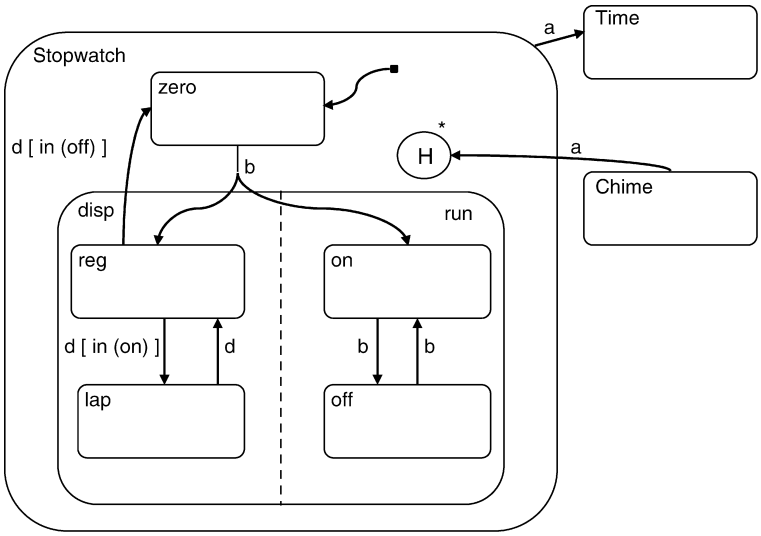


Visual Formalisms. Figure 4. An example of a Petri net.

whether one thing encompasses another, or intersects it, etc. (see, e.g., [15], for early work on this).

Statecharts are not exclusively visual/diagrammatic. Their non-visual parts include, for example, the events that cause transitions, the conditions that guard against taking transitions, and actions that are to be carried out when a transition is taken. For these, statecharts borrow from both the Moore and the Mealy variants of state machines, in allowing actions on transitions between states or on entrances to or exits from states, as well as conditions that are to hold throughout the time the system is in a state.

Unfortunately, some people tend to take diagrams too lightly, finding it difficult to consider a collection of graphics serious or "formal" enough. A common misconception about formality is that textual and symbolic languages are inherently formal, whereas visual and diagrammatic languages are not, as if one could measure formality by how many Greek letters and mathematical symbols the language contains. This myth, together with the deviously deceptive simplicity of graphics, lead to the so called doodling phenomenon – a mindset that says that diagrams are what an engineer scribbles on the back of a napkin but that the real work is done with textual languages.



Visual Formalisms. Figure 5. An example of a statechart.

An interesting topic related to the use of visual formalisms concerns the layout of diagrams, and more generally, their usability and aesthetics. This includes the development of algorithms for the automatic layout of diagrams (see, e.g., [4]), as well as experimental research towards measuring the usability and effectiveness of different visual languages and the development of aesthetic principles.

Another research direction deals with the mechanisms of the language definitions themselves and with the development of new visual languages, for example, domain specific visual formalisms targeted for specific application areas.

Key Applications

Software and Systems Structural Specification

Object model diagrams, and more generally, other structural diagrams, are very popular and are used successfully by software and systems engineers. See the various structural diagrams of the UML [23] and SDL [20].

Reactive Systems Behavioral Specification

Petri nets, statecharts and variants of message sequence charts are successfully used in the design and development of reactive systems. For example, in embedded and real-time safety-critical systems in the automotive, avionics, and telecommunication industries. They are used across the development life cycle of systems, from requirements analysis to specification to simulation to code generation to testing, formal verification, and documentation. In recent years such languages have also been used for the modeling and analysis of biological systems as reactive systems [6,11], which appears to be an application area of great potential.

Data Modeling and Querying

Entity relationship diagrams (ERD) are used for data modeling, or more broadly, for the conceptual specification of databases. Hypergraphs have applications in database theory (see, e.g., [10]). Finally, visual formalisms are also used for the specification of database queries (see [1] for a survey).

Cross-references

- ▶ Activity Diagrams
- ▶ Temporal Visual Languages
- ▶ Visual Query Language

Recommended Reading

1. Catarci T., Costabile M.F., Levialdi S., and Batini C. Visual query systems for databases: a survey. J. Vis. Lang. Comput., 8(2): 215–260, 1997.
2. Chen P.P.-S. The entity-relationship model – toward a unified view of data. ACM Trans. Database Syst., 1(1):9–36, 1976.
3. Damm W. and Harel D. LSCs: Breathing Life into Message Sequence Charts. J. Form. Methods Syst. Des., 19(1):45–80,

2001. Preliminary version in P. Ciancarini, A. Fantechi and R. Gorrieri (eds.). In Proc. 3rd IFIP Int. Conf. on Formal Methods for Open Object-Based Distributed Systems, 1999, pp. 293–312.
4. Di Battista G., Eades P., Tamassia R., and Tollis I.G. Graph Drawing: Algorithms for the Visualization of Graphs. Prentice-Hall PTR, Upper Saddle River, NJ, USA, 1998.
5. Edwards A.W.F. Cogwheels of the mind: the story of Venn diagrmas. Johns Hopkins University Press, 2004.
6. Efroni S., Harel D., and Cohen I.R. Towards Rigorous Comprehension of Biological Complexity: Modeling, Execution and Visualization of Thymic T Cell Maturation. Gen. Res., 13(11): 2485–2497, 2003.
7. Euler L. Commentarii academiae scientiarum Petropolitanae, Vol. 8. 1741.
8. Euler L. Lettres il une Princesse d'Allemagne, Vol. 2. 1772. letters 102–108.
9. Fagin R. Degrees of acyclicity for hypergraphs and relational database schemes. J. ACM, 30(3):514–550, 1983.
10. Fagin R., Mendelzon A.O., and Ullman J.D. A simplified universal relation assumption and its properties. ACM Trans. Database Syst., 7(3):343–360, 1982.
11. Fisher J., Piterman N., Hubbard E.J.A., Stern M.J., and Harel D. Computational insights into C. elegans vulval development. In Proc. Natl. Acad. Sci., 102(6):1951–1956, 2005.
12. Floyd R.W. Assigning meanings to programs. In J.T. Schwartz (ed.). In Proc. Symposia on Appl. Math., Vol. 19. American Mathematical Society, 1967, pp. 19–32.
13. Gane C.P. and Sarson T. Structured Systems Analysis: Tools and Techniques. Prentice-Hall, Englewood, Cliffs, NJ, 1979.
14. Goldstine H.H. and von Neumann J. Planning and Coding of Problems for an Electronic Computing Instrument. Institute for Advanced Study, Princeton, N.J., 1947. Reprinted in von Neumann's Collected Works, Vol. 5 , A.H. Taub (ed.). Pergamon, London, 1963, pp. 80–151.
15. Green T.R.G. Pictures of Programs and Other Processes, or How to Do Things with Lines. Behav. Inform. Tech., 1:3–36, 1982.
16. Harel D. Statecharts: a visual formalism for complex systems. Sci. Comput. Program., 8:231–274, 1987.
17. Harel D. On visual formalisms. Commun. ACM, 31(5):514–530, 1988.
18. Harel D. and Gery E. Executable object modeling with statecharts. Computer, July 1997, pp. 31–42.
19. Harel D. and Marelly R. Come, Let's Play: Scenario-Based Programming Using LSCs and the Play-Engine. Springer, Berlin Heidelberg, New York, 2003.
20. ITU. ITU-T Recommendation Z.100: Specification and Description Language. Technical report, International Telecommunication Union, 1992.
21. ITU. ITU-T Recommendation Z.120: Message Sequence Charts. Technical report, International Telecommunication Union, 1996.
22. Kent S. Constraint diagrams: visualizing invariants in object-oriented models. In Proc. 12th ACM SIGPLAN Conf. on Object-Oriented Programming Systems, Languages & Applications, 1997, pp. 327–341.
23. Object Management Group (OMG). UML: Unified Modeling Language. Available at: http://www.omg.org.
24. Reisig W. Petri Nets: An Introduction, Monographs in Theoretical Computer Science. An EATCS Series, Vol. 4. Springer, Berlin Heidelberg, New York, 1885.
25. Venn J. Symbolic Logic. Macmillan and Co., London, 1881.

Visual Interaction

Maristella Matera
Politecnico di Milano University, Milan, Italy

Synonyms

Visual interaction design; Graphic design

Definition

In the field of Human-Computer Interaction (HCI), *Visual Interaction* refers to the adoption of user interfaces for interactive systems, which make use of visual elements and visual interaction strategies with the aim of supporting perceptual inferences instead of arduous cognitive comparisons and computations. The design of Visual Interaction focuses on the definition of interaction mechanisms through which (i) the users can perform actions on the interactive system by means of visual elements, and (ii) the system can provide feedback to the user, by visually representing the results of the computations triggered by the user actions. The flow of user actions and system feedback over time then has to be coordinated.

In the field of Databases, Visual Interaction refers to the adoption of visual interfaces that provide access to the collection of data stored in databases, by means of visual formalisms and mechanisms supporting both the presentation and the interaction with data.

Historical Background

The information proliferation that characterized the end of the last century led to an increased use of databases by a continuously growing number of users. As an effect of such a trend, the traditional paradigm to access databases, based on formal, text-based query languages, became inadequate for a large portion of inexperienced users.

In 1977, Moshe Zloof proposed QBE (Query By Example) [14], a pioneer query environment where a skeleton of a database relational schema was presented

and, based on it, the users were then able to express queries by entering commands, example elements, and selection conditions in visual tables. QBE was the precursor of a new research line, fully emerged during 1990s on the wave of the consensus gained by interactive systems and visual interfaces. In fact, in that period, several research initiatives, both in the HCI and the database field, focused on enhancing the quality of the interaction with databases, by replacing the traditional, text-based query languages with visual paradigms. A new generation of visual tools, the so called Visual Query Systems (VQSs) (see [4] for a survey), was therefore proposed, which promoted visual formalisms and visual interaction mechanisms at the level of both the presentation of and the interaction with the information contained in databases. The most adopted visual representations were forms, diagrams, and icons, each one being used in accordance with the nature of the information to be represented or queried. Some systems also adopted 3D visualizations [1] or virtual reality techniques [12].

Following the first proposals for VQSs, which especially focused on the interaction with structured, record-based information, some visual query languages for XML were also proposed [2,9]. However, it was the Web that led to the major change in the interaction with databases, proposing a novel, easy-to-use visual paradigm, which suddenly allowed a huge number of users to access huge amounts of data distributed across several data sources.

The Web, today, provides the most widely adopted visual interface for accessing databases.

Foundations

In the interaction with database it is possible to recognize at least two different needs: (i) to let the users *understand* the database content, and (ii) to allow the users to *interact* with the content to extract the information they are interested in. From the visual interaction perspective, the main issue is therefore to find effective mechanisms to transmit to the users the information content of the database and to support users during the interaction with such content. This can result, for instance, in using different *visual metaphors*, defined as a mapping between a data model and a visual model [5], both for the understanding phase and for the interaction phase.

Several works focused on the effectiveness and efficiency of visual metaphors. Some of them (see for example [5,10]) dealt with the effectiveness of schema display, trying to define and formalize the notion of adequate metaphors for representing the database schema, thus supporting the understanding phase. Some other works also covered the effectiveness of the interaction phase. For example, in [3] authors pointed out the need for multi-paradigmatic systems, able to provide querying environments in which *adaptive interfaces* exploit several visual representations and interaction mechanisms depending on the user capabilities and tasks. In [11] the author instead specifically focused on the problem of automatically generating effective presentations of query results, adapting the data representation to the nature of the data themselves, and also to the tasks that such data are meant to support.

The majority of these works were characterized by stratified, multi-component architectures supporting the definition of proper visual formalisms. For example, in [10], authors propose a definition of the visual formalisms adopted for schema display characterized by (i) a *data model* that captures schemas, (ii) a *visual model* that captures visualizations, and (iii) a *visual metaphor* that provides the mapping between data and visual models. Such a stratified definition especially aimed at ensuring flexibility for the selection of the most appropriate metaphor for schema display, and also induced the formalization of many interesting properties. For instance, authors precisely defined the concept of correctness of the visual metaphors (no valid visual schema will map to an invalid data schema and vice versa) through a relationship between the constraints in the data model, and those in the visual model.

While the previous work mainly focused on schema display during the understanding phase, in [3] the authors propose a stratified multi-component structure for the architecture of a multi-paradigmatic VQS, supporting both the understanding and the interaction phase. The most notable features were:

1. The provision of *a formalism for expressing databases*, which is able to represent, in principle, a database expressed in any of the most common data models.
2. The construction and the management of *a user model*, which allows the system to propose to the users the most appropriate visual representation according to their skills and needs.

3. A *transformation layer*, defining the translations among the different available representations, in terms of both database content and visual interaction mechanisms, which allows the users to shift from one representation to another according to their preferences.

Given the always increasing relevance of XML-based data descriptions, in late 1990s some visual query languages for XML also started to be proposed, offering the advantage of letting users manipulate visual objects instead of managing the complexity of the underlying XML-base query languages (e.g., XQuery) [2]. However, even if such languages employed visual primitives, the proposed interaction paradigms still preserved a tight correspondence with the traditional (textual) query formulation. Therefore, their advantages have still to be proved.

An incisive step forward in the visual interaction with databases was, however, determined by the growth of the Internet. The World Wide Web suddenly allowed a huge number of users to easily retrieve and access a large amount of information available in different formats – from unstructured multimedia data to traditional relational data. The Web indeed introduced the *information browsing* paradigm, a very powerful (from the user experience point of view) visual interaction mechanism, based on the "one-click" selection of hyperlinks to explore the information space. This innovation immediately provided an easier alternative to the traditional, text-based way to access and query databases. Especially with the advent of *dynamic Web pages*, relational databases were fully integrated into the Web, and Web pages became a composition of "content units," each one corresponding to an application component in charge of querying at runtime the application data source, and displaying the extracted contents in the page. Additionally, search mechanisms were introduced. Form-based interfaces allowed the users to input (even complex) selection conditions that the Web application was able to transform into suitable SQL queries over the application contents; the query results were then returned into a new Web page, dynamically generated on the fly, typically in the form of an index of linked items.

The Web thus definitely became a "de facto" visual interface for databases. The so called "data-intensive" Web applications gained momentum, offering Web interfaces over huge collections of data equipped with both navigation and search mechanisms, as it happened for on-line trading and e-commerce applications, digital libraries, and several other classes of Web applications interfacing large data sources. Such applications share with the initially proposed VQSs a stratified architecture, being characterized by three orthogonal dimensions:

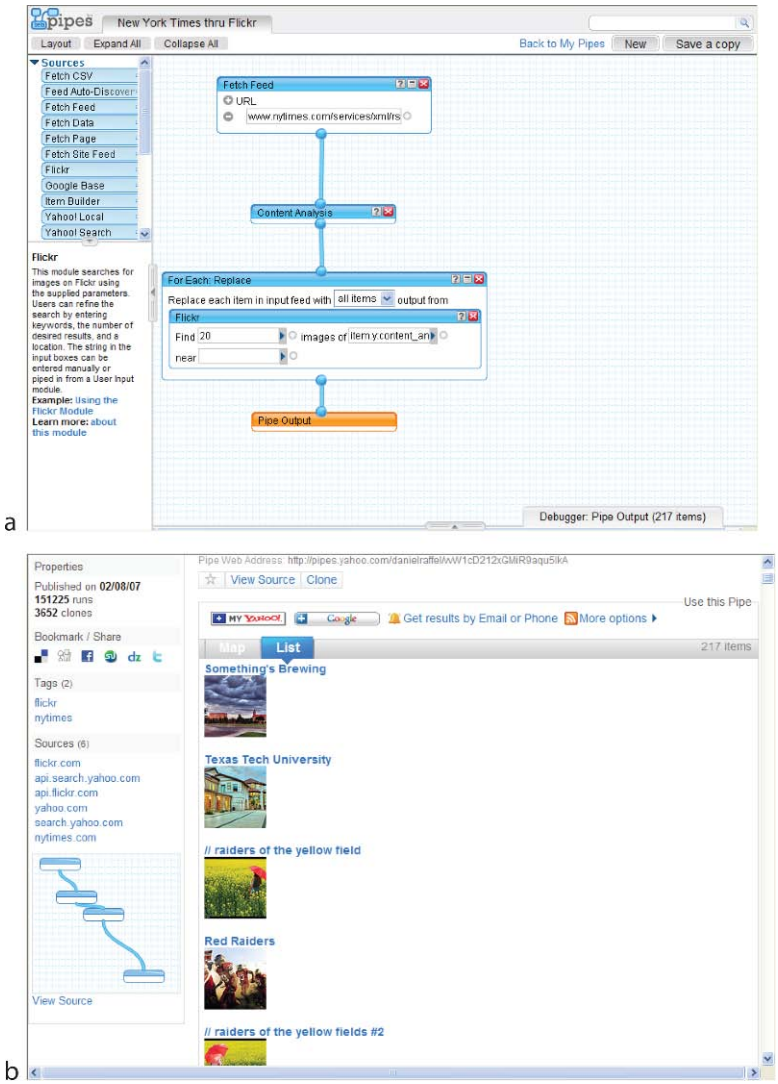
1. The *application data*, collecting the contents to be published by the application.
2. The *hypertext interface*, defining the way in which content units extracted from the application data source are clustered within pages, as well as the navigation mechanisms allowing users to move along the different pages.
3. The *presentation style*, defining layout and graphic properties for the rendering of both contents and navigation mechanisms within pages.

Similarly to VQSs, such a stratified architecture ensures flexibility, by enabling the provision of multiple interfaces over the same content. Hypertext interfaces indeed can be considered *views* over the application data, and therefore several hypertext structures can be constructed over a same database [7] to satisfy the different requirements characterizing different classes of users, different access devices, or even different situations of use. Also, once a hypertext interface is defined, different presentation styles, defining the layout and the graphic properties of the constituent Web pages, can be applied on it. Therefore, especially in the case of adaptive Web applications, several combinations of hypertexts and presentation styles can be adopted to convey contents in the most appropriate modality with respect to the characteristics of the user or of the usage environment, possibly described in a user or a context model [6].

Recently, the so-called *data-driven* methods have been proposed for the design of the hypertext interfaces. They fund their approach on data modeling, and in particular on some properties of the data to be published in the hypertext interface, which can have impact on the effectiveness of the interface itself. In particular, the method proposed in [8] aims to enhance *content accessibility*, which is the ability of a Web application to offer effective and efficient access to its contents, supporting both (i) the identification of the most relevant content entities the application deals with and their mutual relationships (understanding phase), and (ii) the retrieval of the desired content

instances (interaction phase). In Latins, "rem tene, verba sequentur" (manage content, words will follow). Rephrasing this motto, the idea behind this method is that if "Rem" is the information globally available to the application, and "Verba" is the hypertext interface enabling information identification and retrieval through navigation and search, one can say that organizing the application content in accordance with some proper modeling abstractions can induce the definition of hypertext interfaces able to effectively support the information access on the Web. Authors therefore propose some data modeling patterns, called *Web Marts*, which suggest a data organization where:

1. Some *core entities* reflect the most important concepts addressed by the application.
2. Some *detail entities* represent additional information that complete the description of the core concepts.
3. Some *access entities* then classify the core contents for facilitating their access.



Visual Interaction. Figure 1. An example of visual composition of a Yahoo Pipe (http://pipes.yahoo.com/pipes/) (a) and its corresponding rendering (b). The pipe accesses the New York Times homepage (*Fetch Feed* component), analyzes it (*Content Analysis* component), and uses the extracted keywords to find related pictures at the Flickr Web site (nesting a *For each Replace* component with a *Flickr* component).

By taking into account such data characterization, it is possible to achieve Web interfaces able to highlight the application core concepts, facilitating their retrieval by means of navigation and search mechanism defined on top of access entities, and support the browsing of the core concepts by means of navigation patterns defined on top of detail entities. In other words, the resulting hypertext interfaces are able to effectively support both the understanding of and the interaction with the Web application data.

Key Applications

Nowadays, principles of Visual Interaction design are essential for the construction of any system providing access to databases. Even more, they constitute the basis for the development of modern data-intensive Web applications, especially considering the modern Web 2.0 paradigm that, besides other advantages, also promotes more advanced interaction mechanisms, fully comparable to the mechanisms traditionally adopted for desktop interactive applications.

Future Directions

The current trend in the development of modern Web applications – and in particular of those applications commonly referred to as Web 2.0 applications – clearly points toward a high user involvement in the creation of contents. Users visually interact with Web sites not only as passive actors accessing information, but also as creators of the content instances published by the Web application. The so-called *social applications*, like MySpace, YouTube, or Wikipedia, are indeed living an indisputable success.

However, users are not only more and more being involved in content creation. The phenomenon of *Web mashups* is now emerging to provide Web environments where the users have the opportunity to create online applications starting from contents and functions that are provided by third parties. Typical components that may be mashed up, i.e., composed, are RSS/Atom feeds, Web services, content wrapped from third party Web sites, or programmable APIs (like the one provided by Google Maps). Online mashup tools, like Yahoo Pipes (http://pipes.yahoo.com/pipes/) or the Google Mashup Editor (http://editor.googlemashups.com/), and a number of other academic research efforts (see for example [13]) confirm this trend, which implies the addition of a further level of interaction in the Web: users are not only able to visually access data, but they are also able to visually compose the interfaces providing access to data. Figure 1, for example, shows the Yahoo pipes editor, where a simple application is visually modeled by composing some components that analyze the contents published in the New York Times home page, and then use the retrieved keywords to find related photos at another Web site (www flickr.com).

Since unskilled end users are supposed to "program" the necessary composition logic, special emphasis is therefore currently put on intuitive, visual composition environments, not requiring any programming effort. This seems to be one of the major challenges that will be addressed by future academic and industrial research efforts.

Cross-references

- ▶ Visual Interfaces
- ▶ Visual Metaphor
- ▶ Visual Query Language
- ▶ Web 2.0/3.0

Recommended Reading

1. Boyle J. and Gray P.M.D. Design of 3D metaphors for database visualisation. In Proc. IFIP 2.6 3rd Working Conf. on Visual Database Systems. Lausanne, Switzerland, 1995, pp. 157–175.
2. Braga D., Campi A., and Ceri S. XQBE (XQuery By Example): a visual interface to the standard XML query language. ACM Trans. Database Syst., 30(2):398–443, 2005.
3. Catarci T., Chang S.-K., Costabile M.F., Levialdi S., and Santucci G. A graph-based framework for multiparadigmatic visual access to databases. IEEE Trans. Knowl. Data Eng., 8(3): 455–475, 1996.
4. Catarci T., Costabile M.F., Levialdi S., and Batini C. Visual query systems for databases: a survey. J. Visual Lang. Computing, 8(2):215–260, 1997.
5. Catarci T., Costabile M.F., and Matera M. Which metaphor for which database? In People and Computers X, Proc. HCI'95 Conf., 1995, pp. 151–165.
6. Ceri S., Daniel F., Matera M., and Facca F.M. Model-driven development of context-aware web applications. ACM Trans. Internet Technol., 7(1), 2007.
7. Ceri S., Fraternali P., and Matera M. Conceptual modeling of data-intensive web applications. IEEE Internet Computing, 6(4):20–30, 2002.
8. Ceri S., Matera M., Rizzo F., and Demaldé V. Designing data-intensive web applications for content accessibility using web marts. Commun. ACM., 50(4):55–61, 2007.
9. Comai S., Damiani E., and Fraternali P. Computing graphical queries over XML data. ACM Trans. Inf. Syst., 19(4):371–430, 2001.

10. Haber E.M., Ioannidis Y.E., and Livny M. Foundations of visual metaphors for schema display. J. Intelligent Inf. Syst., 3(3/4):263–298, 1994.
11. Mackinlay J.D. Automating the design of graphical presentations of relational information. ACM Trans. Graphics, 5(2):110–141, 1986.
12. Massari A. and Saladini L. Virgilio: a VR-based system for database visualization. In Proc. Workshop on Advanced Visual Interfaces, 1996, pp. 263–265.
13. Yu J., Benatallah B., Saint-Paul R., Casati F., Daniel F., and Matera M. A framework for rapid integration of presentation components. In Proc. 16th Int. World Wide Web Conference, 2007, pp. 923–932.
14. Zloof M. Query By Example: a database language. IBM Syst. J., 16(4):324–343, 1977.

Visual Interaction Design

► Visual Interaction

Visual Interfaces

Tiziana Catarci
University of Rome, Rome, Italy

Synonyms

Graphical user interfaces; GUIs; Direct manipulation interfaces

Definition

Visual Interfaces are user interfaces that make extensive use of graphical objects (icons, diagrams, forms, etc.) that the user may directly manipulate on the screen through several kinds of pointing devices (including her/his fingers) and get an almost instantaneous feedback (near real-time interactivity). Information-intensive interfaces typically exploit one or more visual language, i.e., a language that systematically uses visual signs to convey a meaning in a formal way.

Historical Background

The importance of providing the user with an interactive environment where she/he could directly manipulate the computer content, and get an immediately graspable feedback out of her/his actions, was evident since the first demonstration of the Sketchpad system by Ivan Sutherland [7]. This system was the basis of his 1963 MIT Ph.D. thesis. SketchPad supported manipulation of visual objects using a light-pen, including grabbing objects, moving them, changing size, and using constraints. It contained the seeds of hundreds of important interface ideas. Later on, in the 1970s, many of the interaction techniques popular in direct manipulation interfaces, such as how objects and text are selected, opened, and manipulated, resulted from research at Xerox PARC. The concept of direct manipulation interfaces for everyone was envisioned by Alan Kay of Xerox PARC in a 1977 article about the Dynabook [4]. The first commercial systems to make extensive use of direct manipulation were the Xerox Star (1981) [6], the Apple Lisa (1982) [10], and Macintosh (1984) [9]. Ben Shneiderman at the University of Maryland coined the term "direct manipulation" in 1982, identified the components, and gave psychological foundations [5].

As the quantity of information available became larger and larger, as well as the potential user community of databases and other information sources, there was a growing need for user interfaces that require minimal technical sophistication and expertise by the users, and support a wide variety of information intensive tasks. For instance, information-access interfaces must offer great flexibility on how queries are expressed and how data are visualized.

The general idea was to exploit the well-known high bandwidth of the human-vision channel that allows both recognition and understanding of large quantities of information in no more than a few seconds. Thus, for instance, if the result of an information request can be organized as a visual display, or a sequence of visual displays, the information throughput is immensely superior to the one that can be achieved using textual support. User interaction becomes an iterative query-answer game that very rapidly leads to the desired final result. Conversely, the system can provide efficient visual support for easy query formulation. Displaying a visual representation of the information space, for instance, lets users directly point at the information they are looking for, without any need to be trained into the complex syntax of current query languages. Alternatively, users can navigate in the information space, following visible paths that will lead them to the targeted items. Again, thanks to the visual support users are able to easily understand how to

formulate queries, and they are likely to achieve the task more rapidly and less prone to errors than with traditional textual interaction modes.

Foundations

One fruitful sign system conceived by humans for the purpose of storing, communicating and perceiving essential information is based on bidimensional visual signs, like those contained in pictures, photographs, geographic maps. Visual signs are characterized by a high number of sensory variables: size, intensity, texture, shape, orientation, and color, which all concur to provide details about the information to be communicated [1]. For instance, Tufte suggests many different purposes for which color may be employed: "...to label (color as noun), to measure (color as quantity), to represent or imitate reality (color as representation), and to enliven or decorate (color as beauty). In a geographical map, color labels by distinguishing water from stone and glacier from field, measures by indicating altitude with contour and rate of change by darkening, imitates reality with river blues and shadows hatchures, and visually enlivens the topography quite behind what can be done in black and white alone" [8]. Exploiting the multidimensionality of the visual representation allows people to perform in a single instant of perception a visual selection, in the sense that all the meaningful correspondences among the visualized elements are captured. Other constructions, for example a linear text, do not permit this immediate grasping, the entire set of correspondences may only be reconstructed in the user memory. This is also due to the fact that visual representations are processed by human beings primarily in parallel, while text is read sequentially.

Visual interfaces capitalize on all the above characteristics of envisioning the information, yet one can say that the basic paradigms adopted for enhancing visual interaction are:

1. The *direct manipulation* interaction
2. The *metaphor*
3. The *visual representation*

These three concepts, although separately investigated in the literature, at the present state are so strictly correlated to deeply influence each other, and constitute the foundations for the development of visual interfaces. All three concepts are covered in specific entries of this Encyclopedia.

A *visual language* may be defined as a language that systematically uses visual representations to convey a meaning in a formal way. Frequently, people have used visual languages, iconic languages and graphical languages as synonyms. If one follows the above definition they are all visual languages, but it may be more precise to call iconic languages only those using icons and pictures extensively, while diagrammatic (or graphical) languages are those primarily using diagrams (such as graphs, flow-charts, block-diagrams, etc). When speaking about visual languages, some people allude to languages handling visual objects that are visually presented, i.e., images or pictorial objects of which a visual representation is given, others intend languages whose constructs are visual [3]. Chang calls the first ones *visual information processing languages* and the others *visual programming languages*. In the first case, one deals with conventional languages enhanced with subroutine libraries to handle visual objects. Image processing, computer vision, robotics, office automation, etc. are the typical application domains for these languages. Visual programming languages handle objects that do not necessarily have a visual representation. Within the class of visual programming languages, a subclass exists which manages a particular kind of non-visual objects, namely data in databases; this subclass of languages is that of *visual query languages (VQLs)*, i.e., query languages exploiting the use of visual representations.

While VQLs are usually adopted for query formulation, information visualization mechanisms are used for displaying the results that constitute the answer to a user request. Information visualization, an increasingly important field of Computer Science, focuses on visual mechanisms designed to communicate clearly to the user the structure of information, and allow her/him to make sense out of large quantities of data. Significant opportunities for information visualization may be found today in data mining and data warehousing applications, which typically access large data repositories. Also, the enormous quantity of information sources on the WWW also calls for visualization techniques.

Key Applications

Any kind of traditional database-related activity (e.g., database creation, querying, mining, updating) has been and could be further supported by the usage of suitable visual interfaces, equipped with effective visual

representations, easy-to-use interaction mechanisms and data visualizations. For instance, information-discovery interfaces must support a collaboration between humans and computers, for example, for data mining. Due to humans' limited memory and cognitive abilities, the growing volume of available information has increasingly forced the users to delegate the discovery process to computers, greatly underemphasizing the key role played by humans. Discovery should be viewed as an interactive process in which the system gives users the necessary support to analyze terabytes of data, and users give the system the feedback necessary to better focus its search.

Visual interfaces are obviously also extremely important for novel database applications, dealing with non-traditional data, such as multimedia, temporal, geographical, etc.

Moreover, that the user interface has become a crucial component in the success of any modern (information intensive) application is surely witnessed by the extraordinary growth of the Web: a click on the mouse is all users need to traverse links across the world. However, this much easier access to a huge quantity of various information has created new problems to the users, related to information digestion and assimilation that are difficult to achieve if one lets unstructured floods of data collect in perceptually-rich and information-overabundant displays. The information flood can turn out to be useful only if it can be converted in a somehow structured information flow that users can consume. The Information Foraging theory has been indeed developed in order to evaluate the valuable knowledge gainable from a system in relation to the cost of the necessary interaction.

Future Directions

Issues that are being further explored in modern interfaces include how best to array tasks between people and computers, create systems that adapt to different kinds of users, and support the changing context of tasks. Also, the system could suggest appropriate discovery techniques depending on data characteristics as well as data visualizations, and help integrate what are currently different tools into a homogeneous environment. Modern interfaces use (and will further exploit in the near future) multiple modalities for input and output (speech and other sounds, gestures, handwriting, animation, and video) and multiple screen sizes (from tiny to huge), and have an "intelligent" component ("wizards" or "agents" to adapt the interface to the different wishes and needs of the various users).

However, a key challenge in interacting with current information-abundant electronic environments is to shift from data-centric computer systems, which have been characteristic of over 40 years of Computer Science, to task-centric ones. Indeed, current desktop oriented systems propose a mostly disconnected set of generic tools (word processor, e-mail reader, video and image visualizer, etc.). The fact that these tools are running on one system without being connected leads to awkward situations, such as one re-entering or copying the same data among the different applications. Nowdays, users have to cope with an ever-growing amount of information, which they have to manage and use in order to perform their everyday tasks. This trend forces users to focus more on managing their information rather than using it to accomplish their objectives. Managing information basically amounts to saving it, and possibly being able to find (and re-find) it for subsequent reuse. Given the over-abundance of available information, the process of saving, finding, and re-finding itself needs to be supported by algorithms and tools more powerful than standard OS file browsers. During recent years, the creation of such tools has been the ultimate goal of the so-called Personal Information Management (PIM) field [9]. However, not much has been done in order to cope with the original problem, i.e., to support the user in executing her/his tasks so as to achieve her/his goals. Whereas, it would definitely be beneficial for the user to move from a (static and rigid) object-centric world to a (dynamic and adaptive) task-centric one. Tasks are more of an abstract notion than a computer manageable entity. Hence, such a move requires designing a system that is able to interpret the user's aims and to support the execution of the user's tasks. Tasks need to be explicitly represented in the system both in their static aspects, i.e., the kind of information that they manipulate, the kind of programs involved in these manipulations, security and authentication issues that may arise, and in their dynamic ones, i.e., the sequences of actions that they require, the alternative choices that are given to the user, pre-conditions, post-conditions, and invariants for the various activities that are involved in the task. Both static and dynamic aspects are of direct interest to the user, hence, need to be expressed using explicit semantics that the user can share.

Experimental Results

The key point of any interactive system should be to support, at best, people in achieving their goals and performing their tasks. To be more precise, the interaction should favor an increase in efficiency of people performing their duties without this having to cause extra organizational costs, inconveniences, dangers and dissatisfaction for the user, undesirable impacts on the context of use and/or the environment, long periods of learning, assistance and maintenance. In literature, most of the above listed requirements are synthetically associated to the qualitative software characteristic called usability. Precise usability evaluation needs to be conducted at all stages in the system life cycle in order to: (i) provide feedback which can be used to improve design; (ii) assess whether user and organizational objectives have been achieved; and (iii) monitor long term use of the product or system. In the early stage of design, emphasis is placed on obtaining feedback that can be used as a guide in the design process, while later, when a realistic prototype is available, it is possible to assess whether user and organizational objectives have been achieved. Since in the early stages of design and development changes are less expensive than in later stages, evaluation has to be started as soon as the first design proposals are available. Depending on the development stage of the project, evaluation may be used to select and validate the design options that best meet the functional and user-centered requirements, elicit feedback and further requirements from the users or diagnose potential usability problems and identify needs for improvement in the system. Expert evaluation can be fast and economical. It is good for identifying major problems but not sufficient to guarantee a successful interactive system. Controlled experiments with actual users are a key technique to identify usability issues that might have been missed by the expert evaluation, since they are related with the real usage of the system and the knowledge of the application domain. Generally speaking, evaluation techniques vary in their degree of formality, rigor and user involvement depending on the environment in which the evaluation is conducted. The choice is determined by financial and time constraints, the stage of the development lifecycle and the nature of the system being developed.

Cross-references

- ▶ Data Visualization
- ▶ Diagram
- ▶ Direct Manipulation
- ▶ Form
- ▶ Human-Computer Interaction
- ▶ Icon
- ▶ Information Foraging
- ▶ Multimodal Interfaces
- ▶ Natural Interaction
- ▶ Result Display
- ▶ Usability
- ▶ Visual Metaphor
- ▶ Visual Data Mining
- ▶ Visual Query Language
- ▶ Visual Interaction
- ▶ Visual Representation
- ▶ WIMP Interfaces

Recommended Reading

1. Bertin J. Graphics and Graphic Information Processing. Walter de Gruyter, Berlin, 1981.
2. Catarci T., Dong X.L., Halevy A., and Poggi A. Personal Information Management, chap. Structure Everything. University of Washington Press, Seattle, WA, 2008.
3. Chang S.K. Principles of Pictorial Information Systems Design. Prentice-Hall, Englewood Cliffs, NJ, 1989.
4. Kay A. Personal dynamic media. IEEE Comput., 10(3):31–42, 1977.
5. Shneiderman B. Direct manipulation: a step beyond programming languages. IEEE Comput., 16(8):57–69.
6. Smith D.C., Harslem E., et al. The Star user interface: an overview. In Proc. 1982 National Computer Conference, 1982, pp. 515–528.
7. Sutherland I.E. SketchPad: a man-machine graphical communication system. In Proc. AFIPS Spring Joint Computer Conference. ACM, New York, NY, 1963.
8. Tufte E.R. Envisioning Information. Graphics Press, Cheshire, Connecticut, 1990.
9. Williams G. The Apple Macintosh computer. Byte, 9(2):30–54, 1984.
10. Williams G. The Lisa computer system. Byte, 8(2):33–50, 1983.

Visual Interfaces for Geographic Data

Robert Laurini
LIRIS, INSA-Lyon, Lyon, France

Synonyms

Cartography; Visualizing spatial data; Interactive capture; Interactive layout

Definition

Geographic data are in essence visual multimedia data. To enter this data with key-boards and to print them as tables of data are not very practical actions. For entry, special devices exist (theodolites, lasers, aerial photos, satellite images, etc.), whereas visual layout is currently named cartography or mapping. Visual interfaces have two facets, allowing the user to present their output as maps, possibly with very large printers, and to present spatial queries visually.

Historical Background

In the 1970s, the expression used was "computer-aided mapping" to emphasize the idea that printing maps was the sole scope of using computers. Then, in the 1980s, the more important aspect was considered to be structuring of the geographic database, so then the expression "Geographic Information Systems" was coined. From this period, research has been done on presenting maps as well as presenting queries. For centuries cartographers have created a large corpus of rules for manual mapping. Those rules were progressively integrated and enlarged to compose maps. For instance, zone hatching was considered a boring task when manually done, whereas with a computer, this task is straightforward. In contrast, name placement was considered as a relatively easy task, although in computing, this is still a challenge.

Foundations

It is common to say that "approximately 80% of all government data has some geographic component" (Langlois G., "Federal Computer Week," Jan. 08, 2001). This affirmation states the importance of geographic information not only for administration but also for companies, and not only for environmental and urban planning but also for geo-marketing, since all those data are stored in databases, or more specifically, geographic or spatial databases. In addition to non-spatial attributes, geographic objects are characterized by geometric shapes and coordinates, usually in two dimensions, but increasingly with three dimensions and time. Those characteristics imply three consequences:

1. The necessity of special data models for storing.
2. The necessity of distinguishing storage format and layout format.
3. The necessity of specialized interfaces for both querying against databases and presenting the results, essentially by means of cartography.

The scope of this overall presentation is to study those interfaces. But before presenting these interfaces, it is necessary to emphasize that geographic data are not entered via keyboard but are acquired by different devices, such as theodolites, aerial photos, satellite images, and more recently, GPS (Global Positioning System). These devices are essentially characterized by different resolutions and error levels. Just as theodolites can measure objects within accuracy of less than 0.1 mm, some satellite systems or GPS systems have accuracies of 100 m or less.

On the other hand, according to the size of the screens used and the size of the territory to be mapped, different scales must be used. In other words, the resulting map must be simplified or generalized more exactly in order to reach readability and completeness, as it passes from storage format to some other layout format.

This discussion will be organized as follows: after a short introduction, visual interfaces for cartographic output will be examined first, and then interfaces for querying. Then, a small section on new barriers for mobile handheld devices will conclude.

Visual Interfaces for Cartographic Output

A map is the classical way to respond to a spatial query or to layout the results of some spatial analysis. However, over the centuries, cartographers and geographers have elaborated upon sets of knowledge and know-how to make maps. One of the main problems is known as generalization, i.e., the way to simplify a map; for instance, in a geographic database, to have a country that is described with 1 million points/segments, but must be presented in a thumbnail with only 30 points/segments.

Another aspect is called graphic semiology, i.e., the way to select the symbology for mapping.

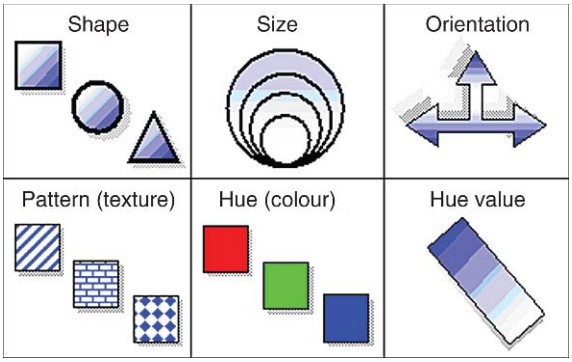
Generalities A fundamental rule in conventional cartography states that any object, once reduced after scaling to less than 0.1 mm cannot be mapped. For instance, an house or a road that is 10 m wide, at 1:1000 scale will be represented by 1 cm, whereas at 1:100,000 scale they will not be represented at all.

Another rule concerns details to be aggregated. An example would be two houses separated by a road. At some scale they will be represented separately, whereas at other scales they will be aggregated. As a consequence, at some scales a city is seen as a collection of separated houses, then as a set of city-blocks, then as a compact area, and finally as a point.

Visual Interfaces for Geographic Data. Figure 1. Example of generalization (1:25000, 1:35000, 1:50000). Source: http://recherche.ign.fr/labos/cogit/arGIGA.php [2].

Generalization By generalization, a piece-wise line can be simplified into a single piece, especially by using the Douglas-Peucker algorithm [3] or variants based on multi-agents systems [5]. Figure 1 gives some examples.

Graphic Semiology Graphic semiology, invented by J. Bertin [1], is the study of the meaning of graphics. In other words, it deals with the signification of drawings, the choice of captions, symbols and icons, together with a methodology to transmit visual messages. Six visual variables have been proposed for representing spatial objects (see Fig. 2) – shape, size, orientation, pattern, hue and value. For instance, to represent a church, a cross symbol (shape) can be used; the symbol's size can be selected according to the initial size of the church, etc.



Visual Interfaces for Geographic Data. Figure 2. Bertin's Visual variables. Source: http://atlas.nrcan.gc.ca/site/english/learningresources/carto_corner/vis_var.gif/image_view.

Animation For some applications, animated cartography can be used, characterized by movement of objects, flickering, mutation, modification of shape or of color, velocity, etc. An excellent example is the animated map for weather forecast, in which some iconized clouds are slowly moving.

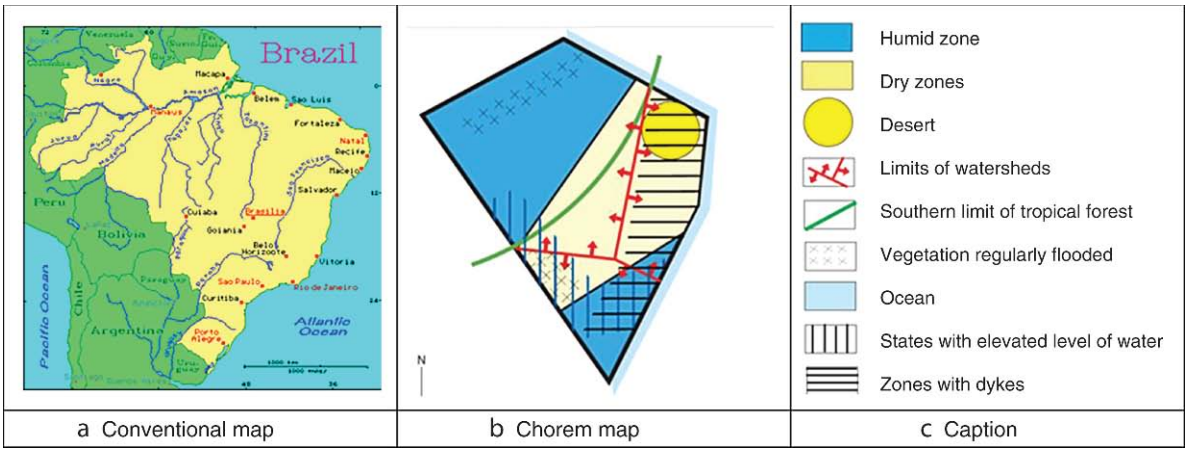
Chorems Chorems are a new way of representing schematized territories. Indeed, for several applications, it is not necessary to restitute the complete database contents, but rather to map the more important aspects. Those chorems were usually designed manually, but by means of spatial and spatio-temporal data mining, geographic patterns can be discovered and mapped. In other words, chorems are a new visual representation of geographic database summaries [4] and a way to represent geographic knowledge. The following example (Fig. 3) emphasizes the water problem in Brazil: it is easy to understand that a conventional river map (Fig. 3a) does not show the more crucial aspects as given in Fig. 3b with caption in Fig. 3c.

Query Input

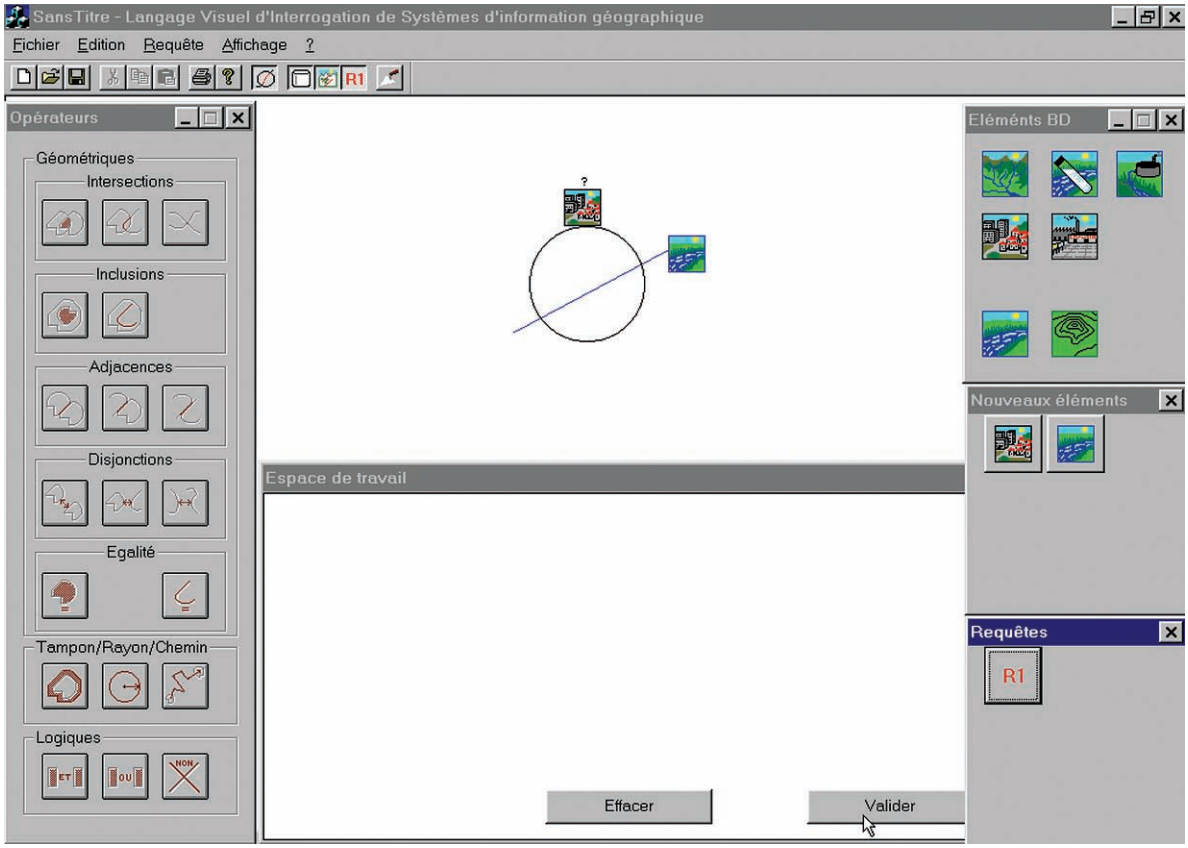
Visual interfaces for GIS are not only used for cartography, but also for query input, i.e., by means of interaction. Present systems can be classified into three categories:

1. Textual queries, such as by using spatial extensions of SQL/ORACLE
2. Tabular queries, by means of forms in which some queries are pre-programmed
3. Graphic queries based on widgets such as icons, mice, and clicks

Some basic queries such as point-in-polygon, region or buffer queries are usually given visually and interactively. However, more complex queries are also usually made, such as queries regarding



Visual Interfaces for Geographic Data. Figure 3. The water problem in Brazil using a conventional river map (a) and a chorem map (b) issued from [5].



Visual Interfaces for Geographic Data. Figure 4. Example of a visual query asking for "all cities crossed by a river" [6].

intersection or adjacency by means of Egenhofer's spatial relation. For instance, the LVIS system [2] is a visual system based upon those relations (See Fig. 4).

However, a fourth method seems more interesting, based on a tangible table in which several persons can collaborate. Figure 5 shows such a table (from the Geodan Company); see [11] for details.

Visual Interfaces for Geographic Data. Figure 5. Example of a tangible table from the Geodan Company (http://www.geodan.nl).

Visual Interfaces for Geographic Data. Figure 6. Example of a map presented on a mobile device.

Final Remarks: Challenges for Small Mobile Devices

In the early 1970s, the main problem was producing maps, and then increasingly maps were seen as results of spatial queries or as results of spatial analysis techniques. However, as it is simple to lay out maps in conventional screens or in very big screens, the screen size of the new handheld mobile devices requires discovering new modes of representing maps and interacting with them, essentially for Location-Based Services. A very common example is the need to represent the way to go from one location to another location, as shown in Fig. 6.

These new mobile handheld devices will imply new techniques for visualizing geographic data, essentially due to screen size.

Key Applications

Any domains in cartography, from urban to environmental planning, geology, archaeology, real estate mapping, location-based services, etc.

Cross-references

- ▶ Cartography
- ▶ Geographical Databases
- ▶ Geographic Information System
- ▶ Spatial Network Databases
- ▶ Spatial Indexing Techniques
- ▶ Spatial Information System

Recommended Reading

1. Bertin J. Sémiologie graphique. La Haye, Mouton, 1970.
2. Bonhomme C., Trepied C., Aufaure M.A., and Laurini R. A visual language for querying spatio-temporal databases. In Proc. 7th Int. Symp. on Advances in Geographic Inf. Syst., 1999, pp. 34–39.
3. Cécile D. Généralisation Cartographique par Agents Communicants: Le modèle CartACom. PhD Dissertation, University Paris VI, 11/06/2004, 2004.
4. Del Fatto V., Laurini R., Lopez K., Loreto R., Milleret-Raffort F., Sebillo M., Sol-Martinez D., and Vitiello G. Potentialities of chorems as visual summaries of spatial databases contents.

In Proc. 9th Int. Conf. Visual Information Systems, 2007, pp. 537–548.
5. Douglas D. and Peucker T. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. The Can. Cartographer, 10(2):112–122, 1973.
6. Kraak M.-J. and Brown A. Web Cartography. CRC, Boca Raton, FL, 2000, 208pp.
7. Kraak M.-J. and Omerling F. Cartography: Visualization of Geospatial Data (2nd edn.). Pearson Education, NJ, 2003, 205pp.
8. Lafon B., Codemard C., and Lafon F. Essai de chorème sur la thématique de l'eau au Brésil. http://histoire-geographie.ac-bordeaux.fr/espaceeleve/bresil/eau/eau.htm, 2005.
9. Laurini R. Information Systems for Urban Planning: A Hypermedia Co-operative Approach. Taylor and Francis, London, 2001, 308pp.
10. Laurini R. and Thompson D. Fundamentals of Spatial Information Systems. Academic Press, San Diego, CA, 1992.
11. van Borkulo E., Barbosa V., Dilo A., Zlatanova S., and Scholten H. Services for an emergency response system in The Netherlands. Second Symposium on Gi4DM, Goa, 2006.

Visual Metaphor

Maria Francesca Costabile[1], Alan F. Blackwell[2]
[1]University of Bari, Bari, Italy
[2]University of Cambridge, Cambridge, UK

Synonyms

Metaphor; Analogy

Definition

Metaphor is a figure of speech, whose essence is to make a comparison between things that are not literally the same. For example, saying "that man is a lion" asks the reader to imagine how a *topic* (the man) could be reinterpreted in terms of some *vehicle* (a lion).

In Graphical User Interfaces (GUIs), *visual metaphor* refers to a kind of analogy, by which designers present the user with an explanation of system behavior in terms of some image. In the early days of the GUI, attempts were made to present all computer behavior in terms of real world analogies, as in the case of the *desktop metaphor*. However, these large scale metaphors soon broke down, as the challenge of developing and maintaining whole systems of correspondence became apparent. Attempts to replicate the success of the desktop metaphor have failed [1]. In practice, UI designers now use the term to refer to visual conventions and genres that, although familiar, need not resemble any real-world objects (e.g., a dialog box or flowchart). The essential benefits of the original desktop metaphor are those provided by *direct manipulation*. Occasionally, real world analogies are successfully used to introduce real world models for new system behaviors (e.g., the shopping basket metaphor in e-commerce). In all cases, user acceptance is encouraged by the use of familiar visual forms, whether pictorial (an *icon* such as a picture of a shopping basket), or an abstract *visual formalism* (conventional layout of dialog boxes, conventional nodes and links for a flowchart).

Key Points

A common view in cognitive science is that all abstract thought is based on physical metaphors [3] (e.g., adding "up"). Computers also use expressions such as "cut and paste" to convey intended usage by analogy to existing technology. Once such figures of speech become widespread, they are unlikely to change. The best principle for visual metaphor design, therefore, is to present users with concepts and terminology that are familiar, whether from the physical world or from previous experience with computers. In creative product design, metaphor can also help designers to imagine novel forms. This more literary strategy is open to interpretation, and may provide less direct guidance to users of the final product.

In the database area, visual formalisms present abstract database concepts to users. GUIs and the growth of database users in the late 80s pushed towards the development of visual query systems, i.e., "systems for querying databases that use a visual representation to depict the domain of interest and express related queries" [2]. A visual representation combines visual formalisms (diagrams, icons, forms) that can be interpreted from experience of existing representation genres – the more familiar and appropriate, the easier it is for the user to understand the usage intended by the designer.

Visual metaphors create expectations about system functionality that, if not fulfilled, can disorientate users. Poor use of metaphor has caused some significant failures of software products. Since there are several ways of creating and interpreting metaphor, the effectiveness of any interface metaphor must be analyzed and empirically evaluated, firstly to ensure

that it is consistent with principles of direct manipulation, and secondly to determine whether users recognize and are familiar with the intended visual formalism.

Cross-references

- ▶ Direct Manipulation
- ▶ Visual Formalisms
- ▶ Visual Interaction
- ▶ Visual Query Language
- ▶ Visual Representation

Recommended Reading

1. Blackwell A.F. The reification of metaphor as a design tool. ACM Trans. Comput. Hum. Interact., 13(4):490–530, 2006.
2. Catarci T, Costabile M.F., Levialdi S., and Batini C. Visual query systems for databases: a survey. J. Vis. Lang. Comput., 8:215–260, 1997.
3. Lakoff G. and Johnson M. Metaphors We Live By. University of Chicago Press, Chicago, 1980.

Visual Mining

- ▶ Visual Clustering

Visual Multidimensional Analysis

- ▶ Visual On-line Analytical Processing (OLAP)

Visual On-Line Analytical Processing (OLAP)

Marc H. Scholl, Svetlana Mansmann
University of Konstanz, Konstanz, Germany

Synonyms

Visual multidimensional analysis; Interactive visual exploration of multidimensional data

Definition

An umbrella term encompassing a new generation of OLAP (On-Line Analytical Processing) end-user tools for interactive ad-hoc exploration of large multidimensional data volumes. Visual OLAP provides a comprehensive framework of advanced visualization techniques for representing the retrieved data set along with a powerful navigation and interaction scheme for specifying, refining, and manipulating the subset of interest. The concept emerged from the convergence of BI (Business Intelligence) techniques and the achievements in the areas of Information Visualization and Visual Analytics. Traditional OLAP frontends, designed primarily to support routine reporting and analysis, use visualization merely for expressive presentation of the data. In the visual OLAP approach, however, visualization plays the key role as the method of interactive query-driven analysis. Comprehensive analysis includes a variety of tasks such as examining the data from multiple perspectives, extracting useful information, verifying hypotheses, recognizing trends, revealing patterns, gaining insight, and discovering new knowledge from arbitrarily large and/or complex data volumes. In addition to conventional operations of analytical processing, such as drill-down, roll-up, slice-and-dice, pivoting, and ranking, visual OLAP supports further interactive data manipulation techniques, such as zooming and panning, filtering, brushing, collapsing etc.

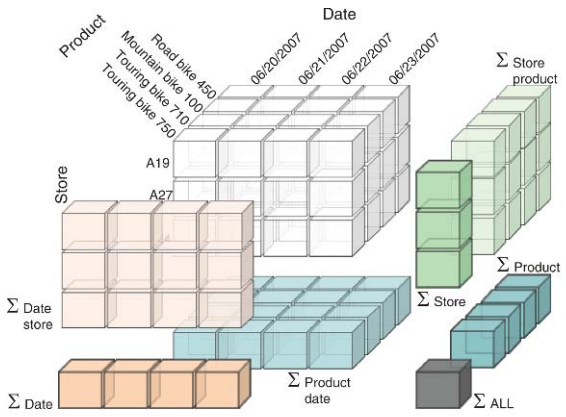
Historical Background

OLAP emerged in the mid-90s as a technology for interactive analysis of large volumes of accumulated and consolidated business data. The underlying multidimensional data model allows users to view data from different perspectives by shaping it into *multidimensional cubes* of measurable facts, or *measures*, as the cube's cells that are indexed by a set of descriptive categories, called *dimensions*. Member values within a dimension may be further arranged into a *classification hierarchy* (e.g., `city` → `state` → `country`) to enable additional aggregation levels. A data cube in a real-world application may hold millions of fact entries characterized by up to 20 dimensions, each supplied with a single or multiple classification hierarchies of various depth. Obviously, to gain insight into such huge and complex data, analysts need mechanisms for projecting the original set onto a two-dimensional (or at most three-dimensional) space and reducing it to a perceivable number of items.

First proposals on using visualization for exploring large data sets were not tailored towards OLAP applications, but addressed the generic problem of visual

querying of large data sets stored in a database. Keim and Kriegel [3] proposed VisDB, a visualization system based on a new query paradigm. In VisDB, users are prompted to specify an initial query. Thereafter, guided by a visual feedback, they dynamically adjust the query, e.g., by using sliders for specifying range predicates on single attributes. Retrieved records are mapped to the pixels of the rectangular display area, colored according to the degree of their conforming to the specified set of selection predicates, and positioned according to a grouping or ordering directive.

A traditional interface for analyzing OLAP data is a pivot table, or cross-tab, which is a multidimensional spreadsheet produced by specifying one or more measures of interest and selecting dimensions to serve as vertical (and, optionally, horizontal) axes for summarizing the measures. The power of this presentation technique comes from its ability to summarize detailed data along various dimensions, and arrange the aggregates computed at different granularity levels into a single view, preserving the "part-of" relationships between the aggregates. Figure 1 exemplifies the idea of "unfolding" a three-dimensional data cube Fig. 1a into a pivot table Fig. 1b, with cells of the same granularity marked with matching background color in both representations. However, pivot tables are inefficient for solving non-trivial analytical tasks, such as recognizing patterns, discovering trends, identifying outliers, etc. [1,4,11]. Visualization has the power to save time and reduce errors in analytical reasoning by utilizing the phenomenal abilities of the human vision system to recognize patterns [2].



a

		Measures							
		Quantity				Amount			
Dimensions		Store							
Product	Date	A19	A27	A34	Total Store	A19	A27	A34	Total Store
– Road Bike 450	06/20/2007	2	7	4	13	498	1743	996	3237
	06/21/2007	9	12	10	31	2241	2988	2490	7719
	06/22/2007	3		7	10	747		1743	2490
	06/23/2007	5	1	9	15	1245	249	2241	3735
Total Road Bike 450		19	20	30	69	4731	4980	7470	17181
– Mountain Bike 100	06/20/2007	8	10	3	21	6392	7990	2397	16779
	06/21/2007	5	11	4	20	3995	8789	3196	15980
	06/22/2007	9	7		16	7191	5593		12784
	06/23/2007	6		4	10	4794		3196	7990
Total Mountain Bike 100		28	28	11	67	22372	22372	8789	53533
– Touring Bike 710	06/20/2007	5		9	14	2995		5391	8386
	06/21/2007	7	2	12	21	4193	1198	7188	12579
	06/22/2007	4		13	17	2396		7787	10183
	06/23/2007		2	8	10		1198	4792	5990
Total Touring Bike 710		16	4	42	62	9584	2396	25158	37138
+ Total Touring Bike 750		19	12	15	46	10621	6708	8385	25714
Total Product		82	64	98	244	47308	83765	49802	133566

b

Visual On-Line Analytical Processing (OLAP). Figure 1. Projecting a multidimensional data cube onto a pivot table. (a) A sample three-dimensional data cube for storing sales transactions as measures `Quantity` and `Amount` characterized by dimensions `Product`, `Date`, and `Store`. (b) A pivot table view of sales data (measures `Quantity` and `Amount`) broken down vertically by `Product` and `Date` and horizontally by `Store`.

OLAP interfaces of the current state of the art enhance the pivot table view by providing a set of popular business visualization techniques, such as bar-charts, pie-charts, and time series, as well as more sophisticated layouts, such as scatterplots, maps, tree-maps, cartograms, matrices, grids, etc. and vendors' proprietary visualizations (e.g., decomposition trees or fractal maps). Some tools go beyond mere visual presentation of data and propose sophisticated approaches inspired by the findings in information visualization research. Prominent examples of advanced visual systems are Advizor by Visual Insights [1] and Tableau by Tableau Software [2].

Advizor implements a technique that organizes the data into three perspectives. A perspective is a set of linked visual components displayed together on the same screen. Each perspective focuses on a particular type of analytical task, such as (i) single measure view using a 3D multiscape layout, (ii) multiple measures arranged into a scatterplot, and (iii) anchored measures presented using techniques from multidimensional visualization (box plots, parallel coordinates, etc.).

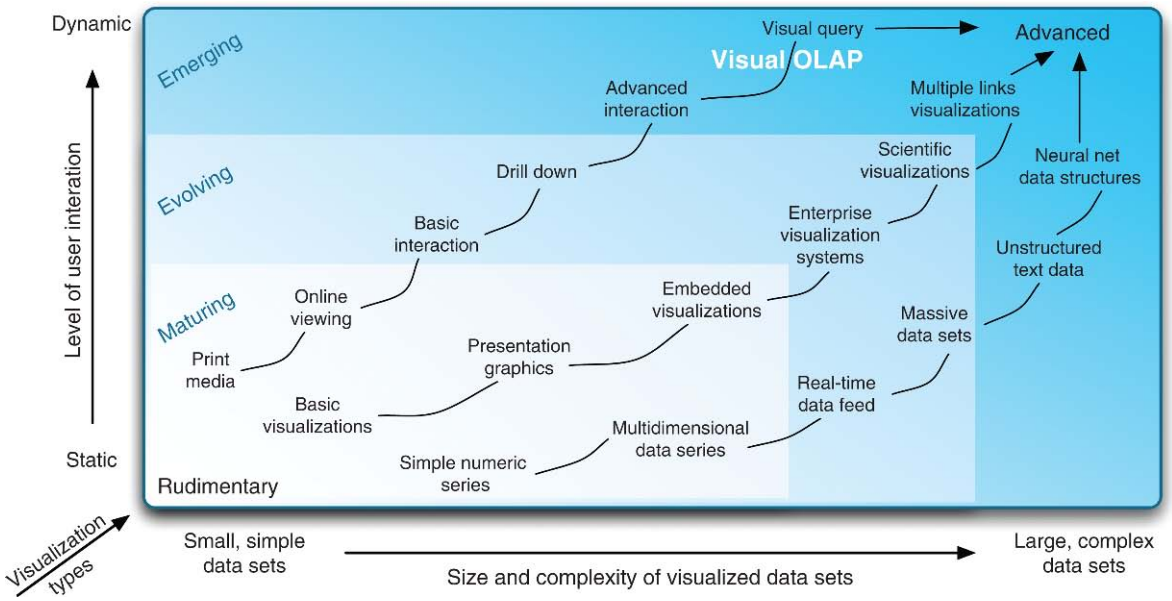
Tableau is a commercialized successor of Polaris, a visual tool for multidimensional analysis developed by a research team of Pat Hanrahan at Stanford University [12]. Polaris inherits the basic idea of the classical pivot table interface that maps aggregates onto a grid defined by dimension categories assigned to the grid's rows and columns. However, Polaris uses embedded graphical marks rather than textual numbers in the table cells. The types of supported graphics are arranged into a taxonomy, comprising rectangle, circle, glyph, text, Gantt bar, line, polygon, and image layouts.

Russom [10] summarizes the trends in business visualization software as a progression from rudimentary data visualization to advanced forms and proposes to distinguish three life-cycle stages of visualization techniques, such as maturing, evolving, and emerging, as depicted in Fig. 2. Within this classification, visual OLAP clearly fits into the emerging techniques for advanced interaction and visual querying.

Ineffective data presentation is not the only deficiency of conventional OLAP tools. Further problems are cumbersome usability and poor exploratory functionality. Visual OLAP addresses those problems by developing fundamentally new ways of interacting with multidimensional aggregates. A new quality of visual analysis is achieved by unlocking the synergy between the OLAP technology, information visualization, and visual analytics.

Foundations

A successful OLAP tool is capable of supporting a wide variety of analytical tasks. As mentioned in the previous



Visual On-Line Analytical Processing (OLAP). Figure 2. Trends in data visualization for business analysis (adopted with modifications from [7]).

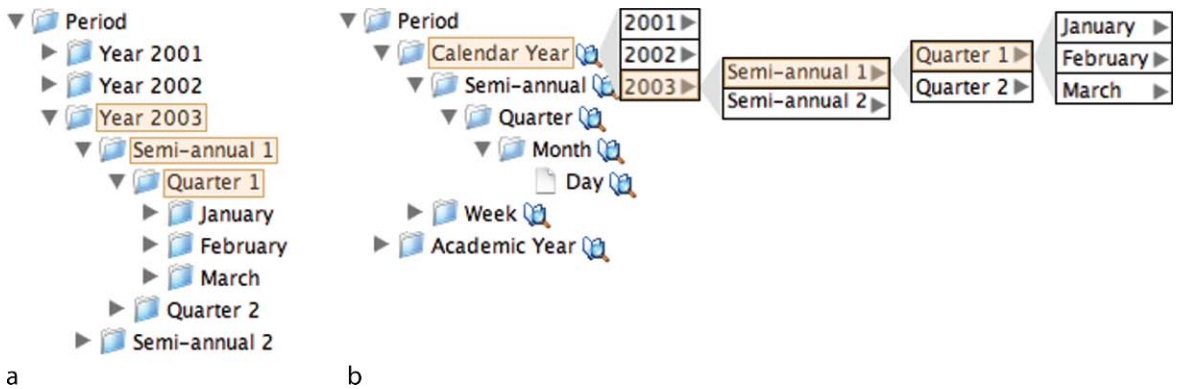
section, different tasks are best solved by applying different visual presentations. OLAP tools account for this diversity by providing a comprehensive framework, which enables users to interactively generate satisfactory visual presentations. The overall query specification cycle evolves by (i) selecting a data source of interest, (ii) choose a visualization technique (e.g., a scatterplot), and then (iii) mapping various data attributes to that technique's structural elements (e.g., horizontal and vertical axes) as well as to other visual attributes, such as color, shape, and size. The main elements of the framework are the navigation structure for visual querying of data sources, a taxonomy of available visual layouts attributes, and a toolkit of interaction techniques for dynamic query refinement and visual representation of the data. A unified framework is obtained by designing an abstraction layer for each element and providing mapping routines (e.g., navigation events to database queries, query results to a visual layout, etc.) implementing the interaction between different layers.

Visual Query Specification

Visual OLAP disburdens the end-user from formulating queries in the "raw" database syntax by allowing purely visual (i.e., by means of using a computer mouse) query specification. Data cubes are represented as a browsable structure whose elements can be queried by "pointing-and-clicking" and "dragging-and-dropping." Visual interface does not trade off advanced functionality for simplicity, it rather facilitates the process of specifying ad hoc queries of arbitrary complexity.

A common navigation paradigm is that of a file browser that represents the contents as a recursive containment of elements. The nodes in the navigation hierarchy may be of types *database*, *schema*, *table* (*cube*), *dimension*, *classification level*, and *measure*. In simplified configurations, the navigation may be limited to a single data cube and, thus, consist solely of dimensions and measures. Dimension hierarchies are presented as recursive nesting of their classification levels, thereby allowing users to browse either directly in the dimension's data or explore its hierarchy schema. In the former approach, denoted *extension-based*, the navigation tree of a dimension is a straightforward mapping of the dimension's data hierarchy: each hierarchical value is a node that can be expanded to access the contained next-level values. Alternatively, the navigation can explicitly display the dimension schema, with each level as a child node of its parent level. This so called *intension-based* approach is especially appreciated for power analysis and employing advanced visualization techniques. The latter navigation strategy becomes the only option when supporting multiple and heterogeneous dimension hierarchies [7] and a multi-cube join or drill-across [6]. Figure 3 shows the difference between instance-based and schema-based browsing for a hierarchical dimension Period.

To navigate in multidimensional aggregates and perform dimensional reduction to extract data for analysis, OLAP defines a number of standard multidimensional analysis operations incorporated into a visual framework in the form of navigation events and interaction options.



Visual On-Line Analytical Processing (OLAP). Figure 3. Browsing options for hierarchical dimensions: extension versus intension navigation. (a) dimension instances arranged in a navigation hierarchy. (b) hierarchy of dimension categories with on-demand data display and an option to switch to extension navigation.

User interactions are translated into valid database queries. From the user's perspective, querying is done implicitly by populating the visualization with data and incrementally refining the view. The output of any OLAP query is a data cube. Visual presentation is generated by assigning the cube's elements – measure values and their dimensional characteristics – to the visual variables of the display. A visualization technique is defined by the visual variables, or attributes, it employs and the way those constructs are combined. Examples of visual attributes are *position*, *shape*, *size*, *color*, and *orientation*. Typically, each of the above visual attributes is used to represent a single data field returned by a query. Various attributes behave differently with respect to the range, data type, and the number of values they can meaningfully represent.

The mapping of the cube's structure to the navigation scheme as well as the translation of navigation events into database queries rely on the metadata of the underlying data warehouse system. Metadata describes the structure of the cubes and their dimensions, measures, and applicable aggregation functions. In an advanced user interface, the analysts are able to define new measures in addition to the pre-configured ones. New measures are obtained by applying a different aggregate function (e.g., average, variance, count) or by specifying a more complex formula over a single or multiple measure attributes (e.g., computing a ratio between two aggregates).

Visualization Techniques

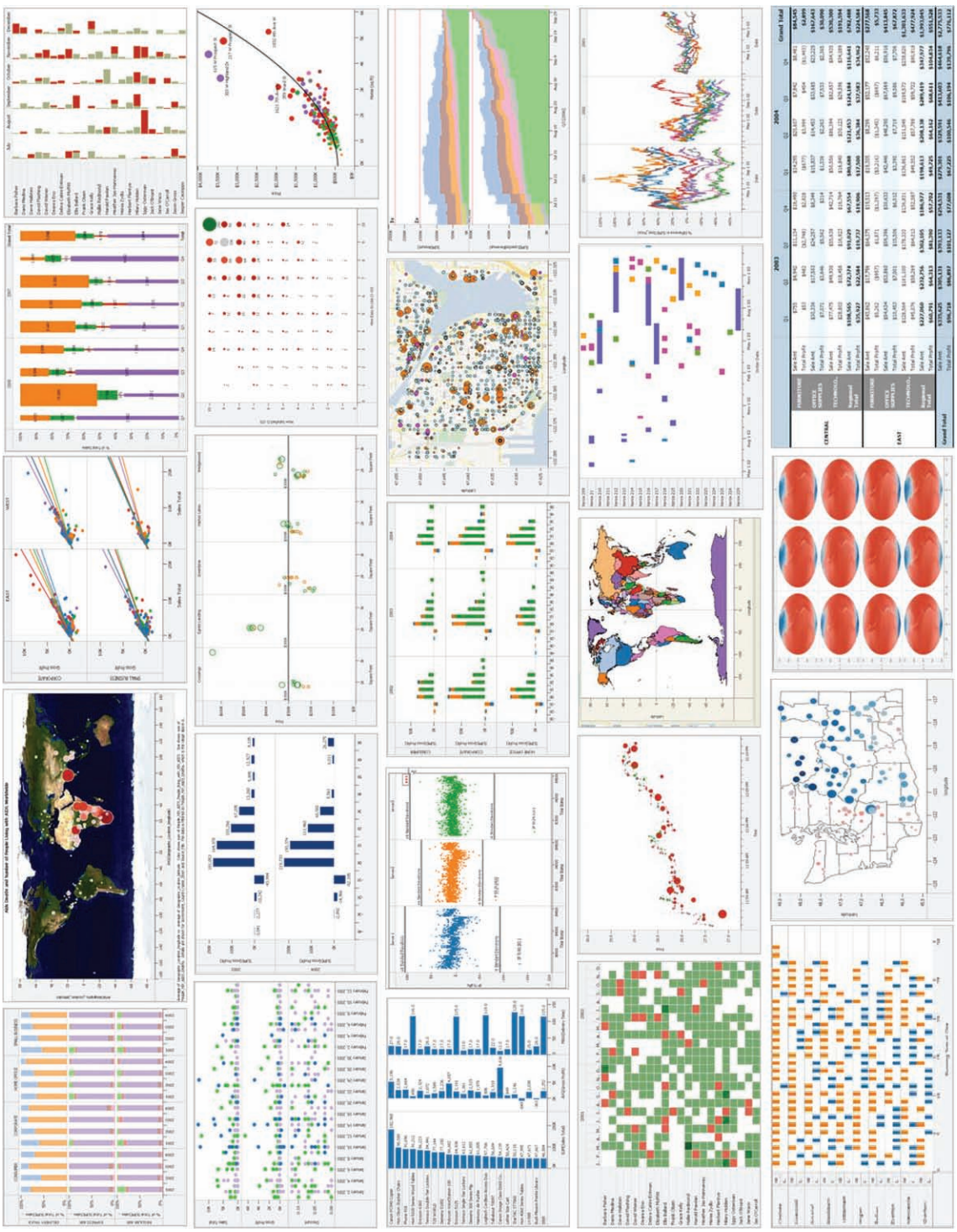
The task of selecting a proper visualization technique for solving a particular problem is by far not trivial as various visual representations (metaphors) may not only be task dependent, but also domain dependent. Successful visual OLAP frameworks need to be based on a comprehensive taxonomy of domains, tasks, and visualizations. The problem of assisting the analyst in identifying an appropriate visualization technique for a specific task is an unsolved issue in state-of-the-art OLAP tools. Typically, a user has to find an appropriate solution manually by experimenting with different layout options.

To support a large set of diverse visualization techniques and to enable dynamic switching from one technique to another, an abstraction layer has to be defined for specifying the relationships between the data and its visual presentation. Maniatis et al. propose an abstraction layer solution, called a *Cube Presentation Model* (CPM) [5], that distinguishes between two layers: a logical layer deals with data modeling and retrieval whereas a presentation layer provides a generic model for representing the data (normally, on a 2D screen). The entities of the presentation layer include points, axes, multicubes, slices, tapes, cross-joins, and content functions. The authors demonstrate how CPM constructs can be mapped to advanced visual layouts at the example of the Table Lens – a technique based on a cross-tabular paradigm with support for multiple zoomable windows of focus.

A common approach to visualization in OLAP application relies on a set of templates, wizards, widgets, and a selection of visual formats. Hanrahan et al. [2] argue, however, that an open set of questions cannot be addressed by a limited set of techniques, and choose a fundamentally different approach for their visual analysis tool Tableau: a declarative visual query language VizQL™ offers high expressiveness and composability by allowing users to create their own visual presentations by means of combining various visual components. Figure 4 illustrates the visualization approach of Tableau by showing just a small subset of sophisticated visual presentations, created using simple VizQL statements, not relying on any pre-defined template layout.

The designers of Tableau deliberately restrict the set of supported visualizations to the popular and proven ones, such as tables, charts, maps, and time series, doubting general utility of exotic visual metaphors [2]. Thereby, Tableau's approach is constrained to generating grids of visual presentations of uniform granularity and limited dimensionality. Other researchers suggest that visual OLAP should be enriched by extending basic charting techniques or by employing novel and less known visualization techniques to take full advantage of multidimensional and hierarchical properties of the data [4,11,14,15]. Tegarden [15] formulates the general requirements of business information visualization and gives an overview of advanced visual metaphors for multivariate data, such as *Kiviat diagrams* and *Parallel Coordinates* for visualizing data sets of high dimensionality, as well as 3D techniques, such as *3D Scattergrams*, *3D line graphs*, *floors and walls*, and *3D map-based bar-charts*.

Another branch of visualization research for OLAP concentrates on developing *multiscale* visualization techniques capable of presenting the data at different levels of aggregation. Stolte et al. describe their



Visual On-Line Analytical Processing (OLAP). Figure 4. Examples of sophisticated multidimensional visualizations generated by simple VizQL statements (used by permission of Tableau Software, Inc.).

implementation of multiscale visualizations within the framework of the Polaris system [13]. The underlying visual abstraction is that of a *zoom graph* that supports multiple zooming paths, where zooming actions may be tied to dimension axes or triggered by a different type of interaction.

Lee and Ong propose a multidimensional visualization technique that adopts and modifies the Parallel Coordinates method for knowledge discovery in OLAP [4]. The main advantage of this technique is its scalability to virtually any number of dimensions. Each dimension is represented by a vertical axis and the aggregates are aligned along each axis in form of a bar-chart. The other side of the axis may be used for generating a bar-chart at a higher level of detail. Polygon lines adopted from the original Parallel Coordinates technique are used for indicating relationships among the aggregates computed along various dimensions (a relationship exists if the underlying sets of fact entries in both aggregates overlap).

Mansmann and Scholl concentrate on the problem of losing the aggregates computed at preceding query steps while changing the level of detail and propose to use hierarchical layouts for capturing the results of multiple decompositions within the same display [9]. The authors introduce a class of multiscale visual metaphors called *Enhanced Decomposition Tree*: the levels of the visual hierarchy are created by decomposing the aggregates along a specified dimension and the nodes contain the resulting sub-aggregates arranged into an embedded visualization (e.g., a bar-chart). Various hierarchical layouts and embedded chart techniques are considered to account for different analysis tasks.

Sifer presents a multiscale visualization technique for OLAP based on *coordinated views of dimension hierarchies* [11]. Each dimension hierarchy with qualifying fact entries attached as the bottom-level nodes is presented using a space-filling nested tree layout. Drilling-down and rolling-up is performed implicitly by zooming within each dimension view. Filtering is realized by (de-)selecting the values of interest at any level of dimension hierarchies, resulting either in highlighting the qualifying fact entries in all dimension views (*global context coordination*) or in eliminating the disqualified entries from the display (*result only coordination*).

A similar interactive visualization technique, called the *Hierarchical Dynamic Dimensional Visualization* (HDDV), is proposed in [14]. Dimension hierarchies are shown as hierarchically aligned barsticks. A barstick is partitioned into rectangles that represent portions of the aggregated measure value associated with the respective member of the dimension. Color intensity is used to mark the density of the number of records satisfying a specified range condition. Unlike in [11], dimension level bars are not explicitly linked to each other, allowing to split the same aggregate along multiple dimensions and, thus, to preserve the execution order of the dis-aggregation steps.

Key Applications

Visual OLAP should be considered an integral part of a BI architecture. The latter appears rather universal with respect to prospective application domains and comprises virtually all business and non-business scenarios that require quantitative analysis. Prominent business application areas are Financial Risk Management, Industrial Process Control, Operations Planning, Capital Markets Management, Network Monitoring, Marketing Analysis, Fraud/Surveillance Analysis, Portfolio Management, Customer/Product Analysis, Budget Planning, Operations Management, and Economic Analysis. Non-business applications are found primarily in government, healthcare, research, and academia.

Future Directions

At present, visual OLAP is lacking a unified formal model and query specification standard. Existing visual analysis frameworks are based on proprietary formalisms and models. Furthermore, advanced OLAP tools claim to turn visualization from the presentation layout into the method of data exploration. This claim implies the need for re-defining visualization as an instrument in terms of its structural components and interaction functions.

A pioneering initiative on addressing the above formalization issues by integrating visualization into a query language is a visual declarative data query language VizQL™ developed at Tableau Software Inc. and released in 2006 [2]. However, VizQL is a proprietary solution and is limited to the visual table paradigm of the Tableau system. To be universally adopted, a new standard for visual exploration of OLAP data has to be open, flexible, and extendible to account for a wide range of visualization approaches.

A promising research direction is to evaluate a wealth of existing visualization techniques with respect

to their applicability to multidimensional analysis and to identify classes of techniques efficient for solving particular analysis tasks. One of the major visualization challenges for OLAP is the ability to present a large number of dimensions on a display. An additional visual attribute for mapping a dimension could be *animation*, as found in Gapminder software for interactive data exploration using animated scatterplots where animation is used to show the evolution of values along the timeline [9].

Another emerging research direction is concerned with spatio-temporal analysis that employs specialized techniques for spatial and/or temporal exploration, such as maps, cartograms, times series, and calendar views. These techniques are aware of the rich semantics behind temporal and geographic dimensions of the data. Rivest et al. [8] propose SOLAP (spatial OLAP) as a visual platform for spatio-temporal analysis using cartographic and general displays. The authors define different types of spatial dimensions and measures as well as a set of specialized geometry-aware operators.

Cross-references

- ▶ Business Intelligence
- ▶ Cube
- ▶ Data Visualization
- ▶ Dimension
- ▶ Hierarchy
- ▶ Measure
- ▶ Multidimensional Modeling
- ▶ On-Line Analytical Processing
- ▶ Visual Interfaces

Recommended Reading

1. Eick S.G. Visualizing multi-dimensional data. ACM SIGGRAPH Comput. Graph., 34(1):61–67, 2000.
2. Hanrahan P., Stolte C., and Mackinlay J. Visual analysis for everyone: understanding data exploration and visualization. Tableau Software Inc., 2007. White Paper, http://www.tableausoftware.com/docs/Tableau_Whitepaper.pdf
3. Keim D.A. and Krigel H.-P. VisDB: database exploration using multidimensional visualization. IEEE Comput. Graph. Appl., 14(5):40–49, 1994.
4. Lee H.-Y. and Ong H.-L. A new visualisation technique for knowledge discovery in OLAP. In Proc. First Pacific-Asia Conference on Knowledge Discovery and Data Mining, 1997, pp. 279–286.
5. Maniatis A.S., Vassiliadis P., Skiadopoulos S., and Vassiliou Y. Advanced visualization for OLAP. In Proc. ACM 6th Int. Workshop on Data Warehousing and OLAP, 2003, pp. 9–16.
6. Mansmann S. and Scholl M.H. Exploring OLAP aggregates with hierarchical visualization techniques. In Proc. 2007 ACM Symp. on Applied Computing, 2007, pp. 1067–1073.
7. Mansmann S. and Scholl M.H. Extending visual OLAP for handling irregular dimensional hierarchies. In Proc. 8th Int. Conf. Data Warehousing and Knowledge Discovery, 2006, pp. 95–105.
8. Rivest S., Bédard Y., and Marchand P. Toward better support for spatial decision making: defining the characteristics of spatial on-line analytical processing (SOLAP). Geomatica, 55(4): 539–555, 2001.
9. Rosling H., Rönnlund A.R., and Rosling O. New software brings statistics beyond the eye. In Proc. Organisation for Economic Co-operation and Development, 2006, pp. 522–530.
10. Russom P. Trends in Data Visualization Software for Business Users. DM Review, May 2000.
11. Sifer M. A visual interface technique for exploring OLAP data with coordinated dimension hierarchies. In Proc. Int. Conf. on Information and Knowledge Management, 2003, pp. 532–535.
12. Stolte C., Tang D., and Hanrahan P. Polaris: a system for query, analysis, and visualization of multidimensional relational databases. IEEE Trans. Visual. Comput. Graph., 8(1):52–65, 2002.
13. Stoltec C., Tang D., and Hanrahan P. Multiscale visualization using data cubes. IEEE Trans. Visual. Comput. Graph., 9 (2):176–187, 2003.
14. Techapichetvanich K. and Datta A. Interactive visualization for OLAP, Part III. In Proc. Int. Conf. on Computational Science and its Applications, 2005, pp. 206–214.
15. Tegarden D.P. Business information visualization. Comm. AIS, 1(1), 1999, Article 4.

Visual Perception

Silvia Gabrielli
Bruno Kessler Foundation, Trento, Italy

Synonyms

Sight; Vision

Definition

In psychology, visual perception is the ability to transform visible light stimulus reaching the eyes into information supporting recognition processes and action. The various physical and processing components which enable a human to being to assimilate information from the environment are known as the visual system.

The act of seeing starts when the cornea and lens at the front of the eye focus an image of the outside world onto a light-sensitive membrane in the back of the eye,

called the retina. The retina is actually the part of the brain which works as a transducer for the conversion of patterns of light into neuronal signals. Precisely, the photoreceptive cells of the retina detect the photons of light and respond by producing neural impulses. These signals are processed in a hierarchical fashion by different parts of the brain, such as the lateral geniculate nucleus, and the primary and secondary visual cortex of the brain.

The major problem in visual perception is that what is seen is not simply and always a translation of retinal stimuli (i.e., the image on the retina). Thus, different theories and experimental studies have been devised to explain what visual processing does to create what is actually perceived. It is worth stressing that visual perception involves the acquisition of knowledge, so it is not merely an optical process but it also entails cognitive activity.

Historical Background

The foundations of modern theories of vision were laid in the seventeenth century, when Descartes and others established the principles of optics, making it possible to distinguish between the physical properties of light, images and the psychological properties of the visual experience. It was proposed that visual perception involves adding information to that present in the retinal image in order to reach properties such as solidity of an object and meaning.

According to the empiricist position, developed primarily by von Helmholtz [8], vision originates from a form of unconscious inference: it is a matter of deriving a probable interpretation from incomplete data (a set of elementary sensations that are integrated and synthesized through a process of learning by association). Inference requires prior assumptions about the world, such as that light comes from above, or that objects are viewed from above and not below. The study of visual illusions (cases when the inference process goes wrong) has yielded a lot of insight into what sort of assumptions the visual system makes.

The unconscious inference hypothesis has recently been investigated in Bayesian studies of visual perception. Proponents of this approach consider that the visual system performs some form of Bayesian inference to derive a perception from sensory data. Models based on this idea have been used to describe various visual subsystems, such as the perception of motion or the perception of depth.

Gestalt psychologists (in the 1930s and 1940s), questioned the assumption that vision is derived from inference mechanisms and previous knowledge [10]. They considered visual perception to be direct and based on the detection of patterns or configurations, as organized wholes available in the perceptual field. According to the Gestalt *Laws of Organization* there are six main factors that determine how things are grouped in visual perception: proximity, similarity, closure, symmetry, common fate and continuity. The major problem with the Gestalt laws and theory is that they are *descriptive* of vision more than *explanatory*.

In Gibson's ecological theory of perception [4], the emphasis is placed on the role played by relations in the visual environment, which are embedded within the spatial and temporal distribution of stimuli an active observer can perceive. According to Gibson, no inference mechanism is needed to detect these *affordances,* that are directly available in the surrounding environment.

Computational approaches to the study of vision, such as Marr's work [5], have provided more detailed explanations of visual phenomena by building artificial intelligence models of the processes involved. Vision is considered as the process of forming a description or representation of what is in the scene from the retinal images. The system at work is modular and serial, consisting of a number of subprocesses, each one taking one representation and transforming it into another. It goes from the creation of the primal sketch (representing changes in light intensity occurring over space in the image and organizing these local descriptions onto a 2D representation of regions and boundaries) to the 2 1/2 D sketch (where the layout of objects surfaces, their distances and orientations relative to the observer are represented) to the creation of 3D model representations (specifying the solid shapes of objects matched against their corresponding representations in memory).

Foundations

According to a multidisciplinary study of visual perception based on perceptual psychology, neuroscience and computational analysis, the purpose of vision is to produce information about objects, locations and events in the world from imaged patterns of light reaching the viewer. Psychology uses the term 'distal stimulus' to refer to the physical world under observation and 'proximal stimulus' to refer to the retinal image. The function of vision is to create a description of aspects of the distal

stimulus given the proximal stimulus. Although visual perception is said to be veridical when it produces accurate descriptions of the real world, in practice vision is better understood in the context of the cognitive and motor functions that it serves.

Vision systems create descriptions of the visual environment based on properties of the incident illumination. The human vision system is primarily sensitive to patterns of light rather than to the absolute magnitude of light energy. The eye does not operate as a photometer. Instead, it detects spatial, temporal and spectral patterns in the light imaged on the retina, and information about these patterns of light form the basis of visual perception. A system which measures changes in the light energy rather than the magnitude of energy has an ecological utility, since it makes it easier to detect patterns of light over large ranges in light intensity. This is also an advantage for computer graphics, which can make graphic displays work effectively by only producing similar patterns of spatial and temporal change to the real world.

Depth perception: Human beings can perceive the world as 3D, although images projected onto the retina are 2D (all points placed at different distances in space, but along the same directional axis, project onto a same point of the retina). Depth perception, also called stereopsis, relies mainly on binocular cues (cues that require input from both eyes) and on monocular cues (cues available from the input of just one eye), as well as on the synthetic integration performed by a person's brain based on the full field of view perceived with both the eyes.

Among the most important binocular cues there are: (i) retinal disparity, due to the distance between the two eyes which makes the projection of objects or scenes onto each retina slightly different. By using two images of the same scene obtained from slightly different angles, it is possible to triangulate the distance to an object with a high degree of accuracy. If an object is far away, the disparity of that image falling on both retinas will be small, if it is close the disparity will be large. (ii) accommodation, focusing on far away objects, the ciliary muscles stretch the eye lens, making it thinner. (iii) convergence, to focus on a same object the two eyes need to converge. The angle of convergence is larger when the eye is fixating objects that are far away.

Examples of monocular cues are: (i) focus, the lens of the eye can change its shape to bring objects at different distances into focus. Knowing at what distance the lens is focused when viewing an object means knowing the approximate distance to that object. (ii) perspective, the property of parallel lines converging at infinity allows people to reconstruct the relative distance of two parts of an object, or of landscape features. (iii) occlusion, the blocking of the sight of objects by others is also a clue which provides information about relative distance (the occluded object is perceived as more far away). (iv) peripheral vision, at the outer extremes of the visual field, parallel lines become curved, as in a photo taken through a fish-eye lens. This effect greatly enhances the viewer's sense of being positioned *within* a real, three dimensional space.

Perceptual constancies: To view the world by the exact image projected on the retina would mean to perceive it as very unstable. Objects within a person's field of vision would constantly be changing shape, size, position and color as s/he moved toward and away from them, because the distance and therefore the amount of reflected light detected by the retina would also be changing. Indeed, this is not the case, our surroundings are perceived as solid and stable since the world is not just "seen" but actively constructed from fragmentary perceptual data [4]. It is our brain that by means of perceptual constancies interprets the changing proximal stimulation to reach stability. For instance, in the case of *size* constancy our brain perceives an object in relation not only to its visual angle, but also to the perceived distance. Also, size constancy tells us that objects moving away from us are the same size even though their retinal image is getting smaller and this is because size constancy is a property of the perceptual field (it refers to the relationship between the object and its surrounding context, which remains stable). The same principle works in *form* constancy, where an object is perceives as the same notwithstanding the different shapes that are projected onto the retina as its angle changes. That is to say that perception of the partial view becomes equivalent to perception of the whole object. Also, in the case of *color* and *light* constancies, our retina is able to distinguish the different wavelengths of light entering the eye, however, our perception remains relatively stable since it is affected by our previous experiences or knowledge, as well as by the context one is dealing with (e.g., color and lightness of an object do not depend only on the absolute intensity of light, but also on their relationships with the stimulation arriving from the surrounding areas).

There are primarily two different ways in which visual perception and the perceptual process have been scientifically investigated. Psychophysical analysis has studied how a person's perception is related to the stimulus. As an example, *Stevens' power law* defines the relationship between the magnitude of a physical stimulus and its perceived intensity or strength [7]. The general form of the law is

$$\psi(I) = kI^a,$$

where I is the magnitude of the physical stimulus, ψ is the psychophysical function capturing sensation (the subjective size of the stimulus), a is an exponent that depends on the type of stimulation and k is a proportionality constant that depends on the type of stimulation and the units used.

A second way of studying perception considers its relation to the physiological processes that are occurring within the person's sensors and/or brain. This is called *physiological analysis* of the perceptual process, which measures, for example, some aspects of a person's brain activity when s/he looks at a visual stimulus. Magnetic resonance imaging (MRI) is a typical instrument used for mapping activation patterns in the human brain as a person watches a scene.

Due to the complexity of visual perception, and the intimate relation among its psychophysical, physiological and cognitive aspects, the best way of getting a complete view of this phenomenon is by cross referencing the different disciplines that have addressed its study.

Key Applications

Computer Graphics

Visual perception has become a key component of computer graphics, since it helps to achieve the development of high fidelity virtual environments in reasonable time by exploiting knowledge of how the human visual system works (e.g., rendering time can be saved by avoiding to compute those parts of a scene that the human will fail to notice).

Artificial Intelligence

The results of studies on human visual perception inspire the development of computational models of vision that, in turn, can be used to design and implement visual abilities within artificial intelligence systems (e.g., robots).

Visual Interfaces

Visual perception provides the scientific basis for the design of visual interfaces that can best match the requirements of their users, by enabling an easier and more transparent access and interaction with information.

Information Visualization

In the field of Information visualization, visual perception principles help to produce effective visualizations of non-geometric data retrieved from large document collections (e.g., digital libraries), the World Wide Web, and databases, with the aim of supporting users to make sense of information there contained and of enhancing their creative thinking.

Future Directions

Current theories and methods for the study of visual perception in psychophysics and neuroscience need to be integrated with information processing approaches in the attempt to model vision at a more abstract computational level. An interesting cross-fertilization is expected between neuroscience experimentation and computer vision theories. Particularly, for providing a detailed account of how the brain makes effective use of top-down resources (e.g., knowledge, memories etc.), creates predictions about forthcoming stimuli and constantly matches expectations against signals from the environment [6]. It is likely that as researchers in visual neuroscience and computer vision develop a better understanding of the role played by bottom-up and top-down information processing mechanisms and their interaction, better theories of visual perception can be developed.

Experimental Results

A considerable amount of experimental evidence exists for all the theories and approaches to the study of visual perception cited above. References to results on the physiology, psychology and ecology of vision are provided in [1,3]. Application of visual perception principles to the design of Information visualization and visual interfaces is discussed in [9].

Data Sets

An online laboratory providing a collection of demonstrations and experiments on visual perception can be found at: http://www2.psych.purdue.edu/~coglab/VisLab/welcome.html.

URL to Code

The portal (above) also intends to provide software that can be used in the study of visual science.

Cross-references

► Visual Interfaces

Recommended Reading

1. Bloomer, C.M. Principles of Visual Perception. Herbert, London, 1990.
2. Bruce V. and Green P.R. Visual Perception. Erlbaum, Hove, England, 1990.
3. Bruce V., Green P.R., and Georgeson M.A. Visual Perception: Physiology, Psychology, and Ecology. Psychology, New York, 2003.
4. Gibson J.J. The Ecological Approach to Visual Perception. Houghton Mifflin, Boston, 1979.
5. Marr D. and Vision W.H. Freeman, San Francisco, CA, USA, 1982.
6. Perception on Purpose. FP6-IST project N. 027268. Available at: http://perception.inrialpes.fr/POP/.
7. Stevens S.S. On the psychophysical law. Psychol. Rev., 64 (3):153–181, 1957.
8. von Helmholtz H. (Obituary). In Proc. Royal Soc. Lond. Royal Society, Great Britain. Taylor and Francis, London, 1854.
9. Ware C. Information Visualization: Perception for Design. Morgan Kaufmann, San Francisco, CA, USA, 2004.
10. Wertheimer M. Gestalt theory. In a Source Book of Gestalt Psychology, W.D. Ellis (ed. & trans.). Routledge and Kegan Paul, London (Original work published 1925), 1938, pp. 1–11.

Visual Query Language

Tiziana Catarci
University of Rome, Rome, Italy

Synonyms

Visual query system

Definition

Visual Query Languages (VQLs) are languages for querying databases that use a visual representation to depict the domain of interest and express related requests. VQLs provide a language to express the queries in a visual format, and they are oriented towards a wide spectrum of users, especially novices who have limited computer expertise and generally ignore the inner structure of the accessed database. Systems implementing VQLs are usually called Visual Query Systems (VQSs) [8].

Historical Background

The birth of VQLs was due to several needs, including: providing a friendly human-computer interaction, allowing database search by non-technical users, introducing a mechanism for comfortable navigation even in case of incomplete and ambiguous queries. It is worth noting that the real precursor of VQLs was QBE, already proposed by Moshe Zloof in 1977 [19].

QBE was really ahead of its time. Indeed, the Zloof's paper states: "the formulation of a transaction should capture the user's thought process. . . ." This is a quite common idea today, but at that time (1977) the research on user interfaces and human-computer interaction was still in its infancy. Approximately in the same period, Smith coined the term "icon" in his Ph.D. thesis [16] and Alan Kay introduced the idea of direct manipulation interfaces that are, in principle, usable by everyone [14]. It was necessary to wait until the beginning of 1980s for the first commercial systems making extensive use of direct manipulation, namely Xerox Star, Apple Lisa, and Macintosh.

It is worth noting that QBE is based not only on the usage of examples for expressing queries, but it also relies on the direct manipulation of relational tables inside a basic graphical user interface: an environment and an action modality which were quite unknown at that time, considering the almost simultaneous publication of Alan Kay's paper.

Another anticipatory idea is the incremental query formulation, i.e., ". . .the user can build up a query by augmenting it in a piecemeal fashion." Many papers still recommend to allow the user expressing the query in several steps, by composing and refining the initial formulations [8].

Moreover, QBE was the first proposal of query language in which the attention to the user interface is coupled with a rigorous definition of the language syntax and semantics and a careful study of the language expressive power. Unfortunately, many of the later proposals of visual query languages have emphasized the aspects related to user interaction and ease of learning and of use only, without also focusing on formal aspects, such as syntax, semantics and expressive power [8]. Such query languages are usually presented through examples of query formulation, making both the study of language expressiveness and the comparison with other languages difficult. The opposite is true for QBE, whose usability was also tested comparing it with SQL in several experiments,

such as those reported in [15] and [17]. Finally, many of the QBE ideas are still up-to-date and it is amazing to note that QBE-like interfaces are nowadays adopted in commercial database systems, despite the current explosion of sophisticated visualizations and interaction mechanisms.

However, only later on, during the 1980s and early 1990s, VQLs received the greatest attention by the database community with the presentation of several diverse proposals for an effective visual and interactive query language.

Foundations

In [5], VQLs were reviewed and classified based on three criteria, namely:

- *What* can be done using the system, i.e., the *expressive power* of the environment.
- *How* the system may be used in order to build the query, i.e., the concept of *usability* as determined by the available strategies, the interaction and representation models.
- *Whom*, in terms of *classes of users*, the systems are addressed to.

The expressive power of a query system can be defined as the ability of the system to extract meaningful information from the database. More formally, according to [10], the expressive power can be based on the concept of *computable query*. Let U be a fixed countable set, called the universal domain, and let D(B), a subset of U, be a finite set which includes all the elements appearing in the database B. A *query* is a partial function giving an output (if any) which is a relation over D(B). A query Q is said to be *computable* if Q is partial recursive and satisfies a consistency criterion: if two databases are isomorphic, then the corresponding outputs of Q are also isomorphic (under the same isomorphism). In other words, the result of a query should be independent from the organization of the data in a database and should treat the elements of the database as uninterpreted objects. Excluding queries expressing undecidable problems, computable queries can be seen as the more general class of "reasonable" queries. Other meaningful classes of queries have been investigated in [11], giving rise to the so called Chandra's hierarchy. A significant class of queries inside the hierarchy is the one of first order queries, and the term *completeness* was used to indicate that a query language could express all the first order queries. However, it was apparent early that the amount of expressive power provided by such a language is not adequate for expressing useful queries, such as transitive closure and, more generally, fixpoint queries (i.e., queries obtained by augmenting the standard first order operators with the construct of least fixpoint). On the other hand, languages that friendly express queries simpler than first order queries are significant, since average database users make elementary requests.

As a consequence of the above remarks, the expressive power of the majority of visual query languages is lower or equal to the classes of first order or fixpoint queries. More precisely, most of the *iconic* languages fall in lowest level classes, since they are directed to a casual user, who is mainly interested in a friendly expression of simple queries (e.g., select-project queries). On the other hand, most of the *graphical* languages are equally or less expressive than relational algebra and very few are placed in the upper levels of the hierarchy (see [8]).

The notion of usability used in [5] is quite "database-oriented," and different from the ones used in the human-computer interaction (hci) community that is now commonly accepted. Indeed, in [5] usability was specified in terms of the models used in VQLs for denoting both data and queries, their corresponding visual representations and the strategies provided by the system in order to formulate the query. This definition does not reflect the hci view of usability as a software quality related to user perception and fruition of the software system. In particular, the data model has no significant impact on the user perception of the system. Whereas, visual representations and interaction strategies influence the user-system interaction, since they are important parts of the system interface (the only thing the user sees of a system).

As for the visual representation, the query representation is generally dependent on the database representation, since the way in which the query operands (i.e., data in the database) are presented constrains the query representation. For example, given a query on a relational database, the query may be formulated in terms of several representations, e.g., filling some fields in tables visualizing the relations, or following paths in a hypergraph that visualizes the relational schema. In this case the table and the hypergraph are two possible

representations associated with the relational database. On the other hand, the visual representation used to display the query result can be different from the database representation. This is mainly due to the fact that what is visualized for the query purpose most often is the schema of the database, while the actual database instances constitute the query result to be displayed to the user.

Visual representations used in VQLs typically use forms, diagrams, icons or a combination of them. *Form*-based representations are the simplest way to provide the users with friendly interfaces for data manipulation. They are very common as application or system interfaces to relational databases, where the forms are actually a visualization of the tables. In query formulation prototypical forms are visualized for users to state the query by filling appropriate fields. In systems such as QBE [19], only the intensional part of relations is shown: the user fills the extensional part to provide an example of the requested result. The system retrieves whatever matches the example. In more recent form-based representations the user can manipulate both the intensional and the extensional part of the database.

Diagrammatic representations are also widely used in existing systems. Typically, diagrams represent data structures displayed using visual elements that correspond to the various types of concepts available in the underlying data model. Diagrammatic representations adopt as typical query operators the selection of elements, the traversal on adjacent elements and the creation of a bridge among disconnected elements.

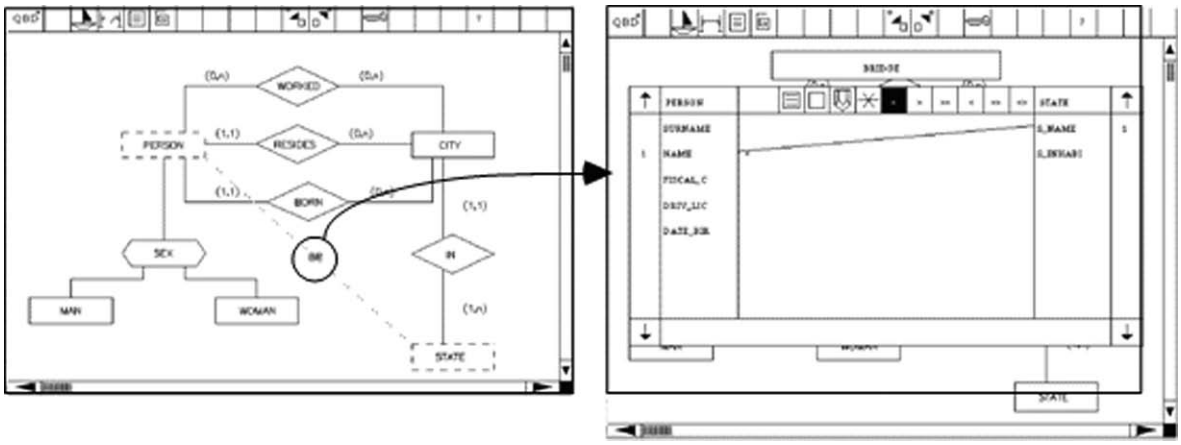
Iconic representations use icons to denote both the objects of the database and the operations to be performed on them. A query is expressed primarily by combining operand and operator icons. For example, icons may be vertically combined to denote conjunction (logical AND) and horizontally combined to denote disjunction (logical OR). In order to be effective, the proposed set of icons should be easily and intuitively understandable by most people. The need for users to memorize the semantics of the icons makes the approach manageable only for somehow limited sets of icons.

The *hybrid* representation is a combination of the above representations. Often, diagrams are used to describe the database schema, while icons are used either to represent specific prototypical objects or to indicate actions to be performed. Forms are mainly used for displaying the query result.

Similarly to most graphical user interfaces, the VQLs that have been developed so far have mainly stressed the user input aspects of the interaction and have given little thought to the visualization of output data. Conversely, an appropriate visualization of the query result allows the user to better capture the relationships amongst the output data, and some systems are progressing in this direction. For instance, AMAZE employs 3D graphs [7]. The data are shown as a 3D snapshot of the n-dimensional results. Different methods of result visualization are also planned to be available to the user. The Film Finder system [1] visualizes information about movies by means of *starfield displays*, which show database objects as small selectable spots (either points or 2D figures). The displayed data can be filtered by changing the range of values on both the Cartesian axes. The query result fits on a single screen and the system quickly, i.e., within a second, computes the new data display in response to the user's requests. This property, called near real-time interactivity, ensures high usability. A different approach is to use virtual reality techniques to present the query result with a simulation of a real environment (i.e., a virtual one) that depicts a situation familiar to the user. VQRH [12] is one of the systems that provide the user with several visual representations for both query formulation and result visualization. One possibility is to use 3D features to present the results in the simulated reality setting. For example, if the database refers to the books in a library, a virtual library can be represented in which the physical locations of the books are indicated by icons in a 3D presentation of the book stacks of the library.

Apart from the visual representation, any VQL is characterized by the way in which it allows the user to express his/her requests. Very often, the actual query specification is the second step of the user interaction, while there is a first step devoted to the understanding of the overall database content. This first phase is in general supported providing the user with browsing and/or filtering and zooming mechanisms.

Query formulation is the fundamental activity in the process of data retrieval. The query strategy *by schema navigation* has the characteristic of concentrating on a concept (or a group of concepts) and moving from it in order to reach other concepts of interest, on



Visual Query Language. Figure 1. Unconnected path in QBD* [19].

which further conditions may be specified. Such a strategy differs according to the type of path followed during the navigation (see Figure 1 for an example of unconnected path).

A second strategy for query formulation is *by subqueries*. In this case the query is formulated by composing partial results. The third strategy for query formulation is *by matching*. It is based on the idea of presenting the structure of a possible answer that is matched against the stored data.

The last strategy for query formulation is *by range selection*, allowing a search conditioned by a given range on multi-key data sets to be performed. The query is formulated through direct manipulation of graphical widgets, such as buttons, sliders, and scrollable lists, with one widget being used for every key. An interesting implementation of such a technique has been proposed in [1], and is called d*ynamic query*. The user can either indicate a range of numerical values (with a range slider), or a sequence of names alphabetically ordered (with an alpha slider). Given a query, a new query is easily formulated by moving the position of a slider with a mouse: this is supposed to give a sense of power but also of fun to the user, who is challenged to try other queries and see how the result is modified. Usually, input and output data are of the same type and may even coincide.

Key Applications

VQLs have been basically proposed to allow non-programmers to express database queries. In [5] it was also conjectured that, depending on the user class and the kind of task (i.e., query), certain VQLs are more appropriate than others. For instance, VQLs based on extensive use of icons and icon composition mechanisms [18], are typically more suited to express simple (select-project-join) queries and be used by naive users. Analysis underlying such statements were usually correct. Nevertheless, they needed to be substantiated by running user trials, and such experiments were (and still are) not so common in the VQL community. However, some of them have been carried on and basically support the hypothesis that different kinds of queries are better supported by different visual representations and interaction mechanisms. Whereas, results on the adequacy of the different visual representations with respect to the various classes of users are not so strong as it was conjectured.

Future Directions

VQLs mainly deal with traditional databases, i.e., databases containing alphanumeric data. However, in recent years the application realms of databases have raised a lot in terms of both number and variety of data types. As a consequence, specialized systems have been proposed for accessing such new kinds of databases, containing non-conventional data, such as images, videos, temporal series, maps, etc. Furthermore, the idea of information repository has been deeply influenced by the beginning of the Web age. Different visual systems have been proposed to cope with the need for extracting information residing on the Web.

Temporal Databases

There are a growing number of applications dealing with data characterized by the temporal dimension

(e.g., medical records, biographical data, financial data, etc.). Still, visual interfaces for querying temporal databases have been less investigated than their counterpart in traditional databases. Typically, end-users of these data are competent in the field of the application but are not computer experts. They need easy-to-use systems able to support them in the task of accessing and manipulating the data contained in the databases. In this case, typical interactions with the data involve the visualization of some of their characteristics over some timeframe or the formulation of queries related with temporal events, such as the change of status of an employee or the inversion of the tendency of stock exchanges.

Geographical Databases

Geographical information is most naturally conveyed in visual format. Maps and diagrams (i.e., schematic maps such as a bus network map) are the core means in user interactions, both for querying the database and displaying the result of a query. A typical query would be "show me on a city map where is the post office that is closest to this location." The query itself would most likely be expressed using preformatted forms and menus to select the city and the reference location. The result would be a blinking or otherwise highlighted point in the displayed map. Once a map is displayed, as a result of a previous query or as an initial background screen in a query formulation interaction, the map can be used to specify a new query. This typically supports queries such as "give me more information on *this*," where the value of the *this* parameter is specified by pointing in some way to a location in the map (i.e., a point on the screen). Thus in some sense visual interaction is common practice in GIS systems, due to the intrinsically spatial reference that is associated to the data.

Web Visual Access

Nowdays, the Web is the widest information repository. However, to find the information of interest among the mass of uninteresting one is a very hard task. In order to help the user in retrieving information scattered everywhere in the Web, several proposals have been made by different research communities, such as those of database, artificial intelligence, and human-computer interaction (see, e.g., [13]), a limited amount of them relate to visual querying and information visualization.

Visually Querying Digital Libraries

The main purpose of a digital library (DL) is to facilitate the users in easily accessing the enormous amount of globally networked information, which includes preexisting public libraries, catalog data, digitized document collections, etc. Thus, it is fundamental to develop both the infrastructure and the user interface to effectively access the information via the Internet. The key technological issues are how to search and how to display desired selections from and across large collections. A DL interface must support a range of functions including query formulation, presentation of retrieved information, relevance feedback and browsing.

Experimental Results

As it was mentioned in the previous sections, user trials have been conducted to compare the usability of query languages. First a well-known definition of usability is recalled: "the extent to which a product can be used with efficiency, effectiveness and satisfaction by specific users to achieve specific goals in specific environments" [3]. More precisely, effectiveness refers to the extent to which the intended goals of the system can be achieved; efficiency is the time, the money, and the mental effort spent to achieve these goals; and satisfaction depends on how comfortable the users feel using the system.

In the case of VQLs the main goal is to extract information from the database by performing queries, and the accuracy in achieving such a goal is generally measured in terms of the accuracy of query completion (i.e., user's correctness rate when writing the queries). Measures of efficiency relate the level of effectiveness achieved at the expense of various resources, such as mental and physical effort, time, financial cost, etc. In principle, both the user's and the organization's point of view should be considered. However, the user's efficiency is most frequently measured in terms of the time spent to complete a query.

The above two measures (i.e., query accuracy and response time) can be evaluated quite precisely. Frequently this is done either by recording real users performing predefined tasks with the system and then analyzing the recorded data or by directly observing the user. The most common tasks are *query writing* and *query reading*, both of which are performed by investigating the relationships between database queries expressed in natural language and the same

queries expressed in the system under study. In query writing, the question is: "Given a query in natural language how easily can a user express it through the query language statements?" The question for query reading is: "Given a query expressed through the query language statements, can the user express the query easily in natural language?" Moreover, other kinds of measures can be defined and evaluated, although with less precision.

Measures of satisfaction describe the comfort and acceptability of the overall system used. The learnability of a product may be measured by comparing the usability of a product handled by one user along a time scale. Measuring usability in different contexts can assess the flexibility of a product.

Usability of query languages has first been studied through the comparison between QBE and SQL [15,17]. The former study [15] showed better user performances when using QBE with respect to SQL, both in query reading and query writing tests. However, a later study [17] also comparing QBE and SQL took into account several factors, such as the use of the same database management system, a similar environment, etc. It is interesting to note that the query language type affected user performance only in "paper and pencil" tests, in which case QBE users had higher scores than SQL users. In on-line tests, the user's accuracy was not affected by the type of the language adopted, but the user's satisfaction was much greater with QBE, and his or her efficiency much better.

In [2], a language based on the previously-mentioned *dynamic queries* was tested against two other query languages, both providing form fill-in as the input method. One of these languages (called FG) has a graphical visualization output, and the other one (called FT) has a fully textual output. The alternative interfaces were chosen to find out which aspect of dynamic queries makes the major difference, either the input by sliders, or the output visualization. The tasks to be performed by the user concerned basically the selection of elements that satisfy certain conditions. However, the subjects were also asked to find a trend for a data property and to find an exception for a trend. The hypothesis that the dynamic query language would perform better than both the FG and the FT interface was confirmed. Similarly, the FG interface produced faster completion times than the FT interface. In particular, for the task of finding a trend, the possibility of getting an overview of the database (in the dynamic and FG interfaces) made the major difference. In searching for an exception, the dynamic interface performed significantly better than the FG and FT ones. This was due to the advantages offered by both the visualization and the sliders. The visualization allowed subjects to see exceptions easily when they showed up on the screen, and the sliders allowed them to quickly change the values to find the correct answer.

Other experiments have been conducted [17,18] to compare a diagrammatic query language, namely QBD*, against both SQL and QBI, an iconic query language. The overall objective of the studies was measuring and understanding the comparative effectiveness and efficiency with which subjects can construct queries in SQL or in the diagrammatic or iconic languages. The experiments were designed to determine if there is a significant correlation between 1) the query class and the query language type, and 2) the type of query language and the experience of the user. The subjects were undergraduate students, secretaries and professionals having different levels of expertise. The results of the comparison between QBD* and SQL confirmed the intuitive feeling that a visual language is easier to understand and use than a traditional textual language not only for novice users, but also for expert ones. The experts' errors when using SQL were mainly due to the need of remembering table names and using a precise syntax. Working with QBD*, users can gain from looking at the E-R diagrams. On the basis of the figures which have been obtained when comparing QBD* and QBI one can say that expert users perform better using the QBD* system, while a small difference exists concerning the performance of non-expert users (slightly better using QBI).

Cross-references

- ▶ Data Model
- ▶ Data Visualization
- ▶ Digital Libraries
- ▶ Expressive Power of Query Languages
- ▶ Geographic Information System
- ▶ Multimedia Databases
- ▶ Temporal Database
- ▶ Usability
- ▶ Query Language

Recommended Reading

1. Ahlberg C. and Shneiderman B. Visual information seeking: tight coupling of dynamic query filters with starfield displays. In Proc. SIGCHI Conf. on Human Factors in Computing Systems, 1994, pp. 313–317.
2. Ahlberg C., Williamson C., and Shneidermann B. Dynamic queries for information exploration: an implementation and evaluation. In Proc. SIGCHI Conf. on Human Factors in Computing Systems, 1992, pp. 619–626.
3. Angelaccio M., Catarci T., and Santucci G. QBD*: a graphical query language with recursion. IEEE Trans. Softw. Eng., 16:1150–1163, 1990.
4. Badre A.N., Catarci T., Massari A., and Santucci G. Comparative ease of use of a diagrammatic Vs. an iconic query language. In Interfaces to Databases J. Kennedy P.J. Barclay (eds.). Electronic Series Workshop in Computing, Springer, 1996.
5. Batini C., Catarci T., Costabile M.F., and Levialdi S. Visual Query Systems: A Taxonomy – nei. In Proc. 2nd IFIP W.G. 2.6 Working Conference on Visual Databases, 1991.
6. Bevan N. and Macleod M. Usability assessment and measurement. In The Management of Software Quality, M. Kelly (ed.). Ashgate Technical/Gower Press, Hampshire, UK, 1993.
7. Boyle J., Leishman S., and Gray P.M.D. From WIMP to 3D: the development of AMAZE. J. Vis. Lang. Comput. (Special issue on visual query systems), 7:291–319, 1996.
8. Catarci T., Costabile M.F., Levialdi S., and Batini C. Visual query systems for databases: a survey. J. Vis. Lang. Comput., 8(2):215–260, 1997.
9. Catarci T. and Santucci G. Diagrammatic vs Textual query languages: a comparative experiment. In Proc. IFIP W.G. 2.6 Working Conference on Visual Databases, 1995, pp. 57–85.
10. Chandra A.K. Programming primitives for database languages. In Proc. 8th ACM SIGACT-SIGPLAN Symp. on Principles of Programming Languages, 1981, pp. 50–62.
11. Chandra A.K. Theory of database queries. In Proc. 7th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, 1988, pp. 1–9.
12. Chang S.K., Costabile M.F., and Levialdi S. Reality bites – progressive querying and result visualization in logical and VR spaces. In Proc. IEEE Symp. Visual Languages, 1994, pp. 100–109.
13. Geronimenko V. and Chen C. (eds.). Visualising the Semantic Web. Springer, Berlin, 2002.
14. Kay A. Personal dynamic media. IEEE Comput., 10(3): 31–42, 1977.
15. Reisner P. Query languages. In Handbook of Human-Computer Interaction, M. M. Helander (ed.). North-Holland, Amsterdam, 1988, pp. 257–280.
16. Smith D.C. Pygmalion: A Computer Program to Model and Stimulate Creative Thought. Birkhauser Verlag, Basel, 1977.
17. Yen M.Y. and Scamell R.W. A human factors experimental comparison of SQL and QBE. IEEE Trans. Softw. Eng., 19(4):390–402, 1993.
18. Tsuda K., Yoshitaka A., Hirakawa M., Tanaka M., and Ichikawa T. Iconic browser: an iconic retrieval system for object-oriented databases. J. Vis. Lang. Comput., 1(1):59–76, 1990.
19. Zloof M.M. Query-by-example: a database language. IBM Syst. J., 16(4):324–343, 1977.

Visual Query System

► Visual Query Language

Visual Representation

Yannis Ioannidis
University of Athens, Athens, Greece

Synonyms

Graphical representation

Definition

The concept of "representation" captures the signs that stand in for and take the place of something else [5]. Visual representation, in particular, refers to the special case when these signs are visual (as opposed to textual, mathematical, etc.). On the other hand, there is no limit on what may be (visually) represented, which may range from abstract concepts to concrete objects in the real world or data items.

In addition to the above, however, the term "representation" is often overloaded and used to imply the actual process of connecting the two worlds of the original items and of their representatives. Typically, the context determines quite clearly which of the two meanings is intended in each case, hence, the term is used for both without further explanation.

Underneath any visual representation lies a mapping between the set of items that are being represented and the set of visual elements that are used to represent them, i.e., to display them in some medium. In order for a visual representation to be useful, the mapping must satisfy certain properties: it must be *expressive* as well as *effective* [1]. Expressiveness is related to the accuracy with which the visual representation captures the underlying items; an expressive mapping should neither lose any information nor lead to additional, irrelevant implications. Effectiveness is related to the quality of the interpretation of the visual representation by a human; an effective mapping should allow for a fast and unique interpretation of the underlying items, leaving no room for ambiguities or false impressions. Expressiveness and effectiveness are often enhanced significantly when

the set of visual elements used to represent a set of (interrelated) items is familiar to the user (possibly from other domains), and the corresponding mapping is intuitive so that it invokes the correct interpretation of these items effortlessly.

The mapping that underlies a visual representation is often called a *Visual Metaphor* [4], as it realizes a "transfer" (metaphor is from the Greek word $\mu\varepsilon\tau\alpha\phi o\rho\ \acute{\alpha}$ – metaphora, which means "transfer") of implicit and explicit characteristics of the visual elements to the items these represent.

The quality of visual representations depends on the visual elements used in it, the choice of which visual element is mapped to which item, and which visual element characteristic is mapped to which item characteristic, and any constraints that must be imposed on the mapping. A brief introduction to the above is the main topic of this entry.

Historical Background

Visual representation of the external world has been exercised by humans for thousands of years and, in recent history, this has extended to abstract worlds as well. Visual metaphors have been used so widely that human cognition is considered tightly interweaved, and sometimes even identified, with human vision. Work on visual representation of data and information dates as far back as the work of the Scottish political economist William Playfair, who was the first to use bar charts, pie charts, and other extremely intuitive and useful visual constructs that are used today for this purpose [6]. In recent times, many theories on the semantics of visual representations and how they may enhance human perception have been published, a great example being the work of Tufte [6,7].

Visual representation of information is extensively used in several types of computer systems, which are all using, either explicitly or implicitly, a set of visual elements and a mapping from them to data items being visually represented. Given that the variety of possibilities regarding both the elements and the mappings is endless, there have been a plethora of papers that have been written on visual representation, including some aggregate bodies of work that offer a comprehensive view of the entire field [2].

In the following section, a formal model for visual representation is presented. Although visual representations can be employed for any kind of abstract or concrete items, for ease of exposition and without loss of generality, the discussion focuses on representing data or information items.

Foundations

This section presents a formalism that captures the essence of visual representation and can be used as a means to classify and compare several approaches to the topic, and to explain some of the features of the relevant systems. For the sake of presentation clarity, it makes various simplifying assumptions and, therefore, its expressive power is not as high as some existing visual representations manifest. Nevertheless, it is expressive enough to capture a broad class of visual representations and characteristic enough to convey the main aspects of the concept overall.

Without loss of generality, any dataset is assumed to conform to the structure and constraints of some *data model* (in database terminology, that would be a *data schema*). Visualizations of such a dataset require the corresponding notion of a *visual model*, which is similar to a data model, except that its primitives are visual. In analogy to a dataset being an instance of a data model, a visual representation that is an instance of a visual model will be called a *visualset*.

In order for a visualset to obtain meaning and, therefore, become a visual representation of a dataset, it needs a mapping of its elements to the corresponding items of the dataset, so that users may see the former and understand the latter. Such a mapping is typically defined at the (data and visual) model level and propagated down to all instances of the models. Given this, such a mapping is called a (*visual*) *metaphor*.

Separation of the visual representation process into three distinct components (visual model, data model, and visual metaphor) has many benefits. It permits metaphors to be tested for correctness, evaluated, compared, and combined. Also, it permits systems, especially visual-representation and user-interface tools, to be evaluated with respect to (i) the quality of their visual models and metaphors, (ii) their flexibility with respect to defining or changing their models and metaphors, and (iii) the particular ways in which this is done.

Using the above three notions, the problem of visual representation of datasets may be stated more formally as follows. Given a data model $\mathcal{D}$, let $S(\mathcal{D})$ denote the set of valid datasets that can be constructed based on that model. Similarly, let $S(\mathcal{G})$ denote the set of visualsets that can be constructed based on a visual

model $\mathcal{G}$. The goal is to establish a binary relation between $S(\mathcal{G})$ and $S(\mathcal{D})$, the metaphor, whose specific properties depend on the precise operations that users want to perform on visualsets. Specifically,

1. If users want to be able to use a visualset of $\mathcal{G}$ to view *any* dataset of $\mathcal{D}$ in its *entirety*, then an onto function must exist of the form $f : S(\mathcal{G}) \rightarrow \mathcal{S}(\mathcal{D})$, so that every dataset can be represented visually.
2. If, in addition, users want to be able to use a visualset of $\mathcal{G}$ to update a dataset of $\mathcal{D}$, then a total onto function must exist of the form $f : S(\mathcal{G}) \rightarrow \mathcal{S}(\mathcal{D})$, so that every visualset can be uniquely interpreted as a dataset.

Clearly, not all such functions f that satisfy the above properties are useful. Many are arbitrary mappings, with no obvious correspondence between the dataset and visualset. The goal is to establish a relationship between the members of $S(\mathcal{D})$ and $S(\mathcal{G})$ so that when users view a visualset, they can infer the dataset to which it maps. This implies that f should be derived from a correspondence between the individual components and features of the data and visual models, which would result in a structural similarity between datasets and visualsets, making it easier for users to understand the former through the latter.

Data and Visual Models

The formalism below (essentially a meta-model) identifies the key components and features of a large and interesting class of data and visual models, which serves as an important example of the concepts surrounding visual representation. In that formalism, every data or visual model $\mathcal{M}$ can be seen as a quadruple $\mathcal{M} = < \mathcal{P}, \mathcal{A}, \mathcal{V}, \mathcal{C} >$ defined as follows:

- $\mathcal{P}$ is a finite set of classes (containers) of data items (resp., visual elements). Each such class P is associated with a (possibly infinite) set of unique ids $\mathcal{I}(P)$ that can be used to identify the members of class P.
- $\mathcal{A}$ is a finite set of identifiers of attributes that data items (resp., visual elements) are expected to have, depending on the class they belong in. If $\mathcal{A}(P)$ represents the attributes of the members of class P, then $\mathcal{A} = \bigcup_{P \in \mathcal{P}} \mathcal{A}(P)$. For all $P \in \mathcal{P}$, each member of $\mathcal{A}(P)$ is of the form $P.A$, where A is the name of the corresponding attribute of the members of class P.
- $\mathcal{V}$ is a (possibly infinite) set of identifiers of the possible values that the above attributes may have. If $\mathcal{V}(P.A)$ represents the possible values of attribute $P.A$, then $\mathcal{V} = \bigcup_{P \in \mathcal{P}} \bigcup_{P.A \in \mathcal{A}(P)} \mathcal{V}(P.A)$. For all $P.A \in \mathcal{A}(P)$, each element of $\mathcal{V}(P.A)$ is of the form $P.A == v$, where v is the corresponding potential value of attribute $P.A$. For an attribute $P.A$, $\mathcal{V}(P.A)$ may be equal to a set $\mathcal{I}()$ of ids of the members of some class.
- $\mathcal{C}$ is a finite set of *constraints*, i.e., rules that must be satisfied by any dataset (resp., visualset) that conforms to $\mathcal{M}$. These constraints are formulas in some prespecified language $\mathcal{L}$ and refer to members of $\mathcal{P}$, $\mathcal{A}$, $\mathcal{V}$.

To draw an analogy, in relational database terms, the above describes relational tables ($\mathcal{P}$) with attributes ($\mathcal{A}$), the attributes' domains ($\mathcal{V}$), and integrity constraints ($\mathcal{C}$). It is conceivable that some data or visual model may not be representable in (any enhancement of) the above formalism. Most of the common models fall naturally in this formalism, however, so it is the one adopted in this entry.

Data models capture abstract organization of information and their creation is a classical database problem that is well understood. As an example, consider a very simple data model on *employees*. Each employee has a *name*, a *salary*, an *age*, and a *department*. The three possible departments are "toy", "candy", and "shoe". This data model is the quadruple $\mathcal{D} = < \mathcal{P}_\mathcal{D}, \mathcal{A}_\mathcal{D}, \mathcal{V}_\mathcal{D}, \mathcal{C}_\mathcal{D} >$, where $\mathcal{C}_\mathcal{D} = \emptyset$, and the data-item classes in $\mathcal{P}_\mathcal{D}$, their attributes in $\mathcal{A}_\mathcal{D}$, and their corresponding value sets in $\mathcal{V}_\mathcal{D}$ are given in the following table.

Data item class (P)	Attribute ($P.A$)	Attribute values ($\mathcal{V}(P.A)$)
Employee	Name	text
	Salary	[0, 10000]
	Age	[0, 100]
	Dept	{toy, candy, shoe}

Contrary to the situation with data models, creating suitable visual models is less straightforward, since they must reflect not only the (visual) information to be organized, but also the medium in which the

models are expressed. To create visual item classes for a visual model, visual building blocks are needed, which may actually be combinations of standard visual elements, such as points, lines, regions, text, etc. (A more formal discussion of visual constructs may be found in [3].) Constraints in the visual model are used to hold compositions together. For example, if a visual element is a box with a piece of text in the center, then it is a composition of a region and a text item satistfying the constraint that the location of the text time (its center) is the same as the location of the region (its center).

As an example, consider a very simple visual model that supports the placement of objects in a two-dimensional data field. The only class of visual items is *point* (not in the mathematical sense), which is simply a region. This visual model is the quadruple $\mathcal{G} = < \mathcal{P}_\mathcal{G}, \mathcal{A}_\mathcal{G}, \mathcal{V}_\mathcal{G}, \mathcal{C}_\mathcal{G} >$, where again $\mathcal{C}_\mathcal{G} = \emptyset$, and the visual-item classes in $\mathcal{P}_\mathcal{G}$, their attributes in $\mathcal{A}_\mathcal{G}$, and their corresponding value sets in $\mathcal{V}_\mathcal{G}$ are given in the following table. For simplicity, only a subset of the attributes is shown.

Visual item class (P)	Attribute ($P.A$)	Attribute values ($\mathcal{V}(P.A)$)
Point	LocationX	plane-pixel-x
	LocationY	plane-pixel-y
	Size	{100 pixels}
	Shape	{square, oval, triangle}
	Color	{blue, red, green}

Visual Metaphors

A visual metaphor is defined as a correspondence between features of a visual and a data model. This correspondence induces a mapping between visualsets (instances of the visual model) and datasets (instances of the data model). Having the mapping be decomposed into mappings of the model features helps produce visualizations that, when viewed, allow the user to deduce the underlying dataset. Consider a data model $\mathcal{D} = < \mathcal{P}_\mathcal{D}, \mathcal{A}_\mathcal{D}, \mathcal{V}_\mathcal{D}, \mathcal{C}_\mathcal{D} >$ and a visual model $\mathcal{G} = < \mathcal{P}_\mathcal{G}, \mathcal{A}_\mathcal{G}, \mathcal{V}_\mathcal{G}, \mathcal{C}_\mathcal{G} >$. A metaphor includes correspondences between item classes ($\mathcal{P}_\mathcal{D}$ and $\mathcal{P}_\mathcal{G}$), their attributes ($\mathcal{A}_\mathcal{D}$ and $\mathcal{A}_\mathcal{G}$), and their attribute values ($\mathcal{V}_\mathcal{D}$ and $\mathcal{V}_\mathcal{G}$). These correspondences describe the meaning of visual model features relative to the underlying data model. To allow presentation flexibility, correspondences are permited between multiple features in the visual model and a single feature in the data model.

More formally, a *metaphor* T is an onto function from $\mathcal{G}$ to $\mathcal{D}$ (denoted by $T : \mathcal{G} \rightarrow \mathcal{D}$), which is the union of the following three onto functions:

1. Function $T_p : \mathcal{P}_\mathcal{G} \rightarrow \mathcal{P}_\mathcal{D}$
2. Function $T_a : \mathcal{A}_\mathcal{G} \rightarrow \mathcal{A}_\mathcal{D}$, where $T_a(P_G.A_G) = P_D.A_D$ only if $T_p(P_G) = P_D$
3. Function $T_v : \mathcal{V}_\mathcal{G} \rightarrow \mathcal{V}_\mathcal{D}$, where $T_v(P_G.A_G == v_G) = (P_D.A_D == v_D)$ only if $T_a(P_G.A_G) = P_D.A_D$

As an illustration of the above definition, the following metaphor T maps from the visual model to the data model discussed above.

A metaphor provides meaning to the features of a visual model by establishing a correspondence between them and features of a data model. The precise meaning is captured by T. For example, displaying a red square *point* implies the existence of an *employee* in the toy department. The metaphor must be such that users can correctly and unambiguously interpret a

x	$T_p(x)$	x	$T_a(x)$	x	$T_v(x)$
Point	Employee	point.locationX	employee.salary	point.locationX==x	employee.salary==(x*10)
		point.locationY	employee.age	point.locationY==x	employee.age==(x/5)
		point.color	employee.dept	point.color==“red”	employee.dept==“toy”
				point.color==“green”	employee.dept==“candy”
				point.color==“blue”	employee.dept==“shoe”
		point.shape	employee.dept	point.shape==“square”	employee.dept==“toy”
				point.shape==“oval”	employee.dept==“candy”
				point.shape==“triangle”	employee.dept==“shoe”

displayed visualset. This can be made more precise based on various mandatory and optional properties of T, i.e., being a function, onto, total, and 1-1.

As a minimum requirement, a metaphor T has been defined as an onto function: if it weren't onto, then some characteristics of a data model would not be captured visually; if it weren't a function, then a single visual construct would have multiple meanings and, therefore, would not be interpreted appropriately.

Beyond the above, a visual model may have features that do not carry any meaning and/or features that carry redundant meaning. Based on knowledge of T, users should be able to ignore the former and not be confused by the latter. In particular, if a metaphor is not total then some visual elements do not mean anything in the data model. This creates no problem if the metaphor will be used simply for visual display of datasets: unmapped features of the visual model will be used just for presentation. On the other hand, if the metaphor will be used for updating datasets displayed, then if a visual attribute $P_G.A_G$ is mapped by T_a, then all the values in $\mathcal{V}_{\mathcal{G}}(P_G.A_G)$ should be mapped as well, otherwise, one would be able to draw visualsets that are not translatable to datasets.

Also, for both retrieval and update, if a metaphor is not 1-1, then multiple visual elements have the same meaning in the data model. If T_p or T_v are not 1-1, then there is a choice of visual constructs that can be used, which should be left to the user or resolved via some default mechanism. If T_a is not 1-1, then there is redundancy: multiple visual attributes capture the same data attribute.

Functions that are not 1:1 establish equivalence classes among the features of the visual model, i.e., several features may have the same meaning. For example,

$$T_a(point.color) = T_a(point.shape) = employee.dept$$

specifies redundancy: visual attributes *point.color* and *point.shape* redundantly capture data attribute *employee.dept*. Value mappings are similar:

$$\begin{aligned} &T_v(\text{point.color} == ''\text{green}'') \\ &= T_v(\text{point.shape} == ''\text{oval}'') \\ &= (\text{employee.dept} == ''\text{candy}'') \end{aligned}$$

indicates that specific values of redundant visual attributes (e.g., *point.color* and *point.shape*) capture the same value of a data attribute.

Datasets, Visualsets, and Visual Representation

As defined above, a dataset or visualset may be considered as an instantiation of a data or visual model, respectively. In particular, a dataset S of a model $\mathcal{M}$ is defined as follows (similarly for a visualset):

1. For every $P \in \mathcal{P}$, there is a finite set $[P] \subseteq \mathcal{I}(\mathcal{P})$ of data items in class P that appear in dataset S.
2. For every $P.A \in \mathcal{A}$, there is a total function $[P.A] : [P] \rightarrow \mathcal{V}(P.A)$, which determines the value of the $P.A$ attribute for every data item in $[P]$.
3. For every $c \in \mathcal{C}$, there is a constraint $[c]$, constructed from c by replacing every $P \in \mathcal{P}$ by $[P]$ and every $P.A \in \mathcal{A}$ by $[P.A]$. All these constraints are satisfied by the above.

Given a metaphor $T : \mathcal{G} \rightarrow \mathcal{D}$, any dataset that conforms to data model $\mathcal{D}$ can be represented by a visualset that conforms to visual model $\mathcal{G}$ via a mapping of their contents that remains faithful to the metaphor. In particular, visual items in the classes of such a visualset represent data items in the corresponding classes (based on T) of the dataset. Similarly, the visual attributes and the values of these visual elements represent the data item attributes and their values as specified by T. Essentially, there is a function t induced by T that determines a mapping from any visualset that conforms to $\mathcal{G}$ to some dataset that conforms to $\mathcal{D}$. As with metaphor T, the induced function t can be extended so that its domain includes instantiations of constraints as well, i.e., $t([c])$ is valid for $c \in \mathcal{C}_{\mathcal{G}}$.

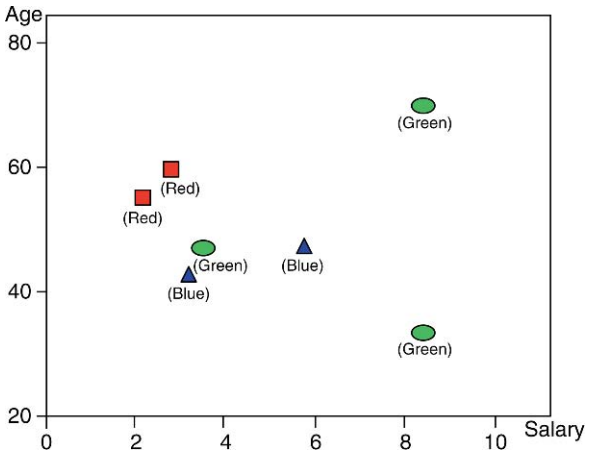
As a simple example of all the above, consider the following dataset that conforms to the data model described above:

employee(Jason, 2.1, 55, "toy")
employee(Iris, 2.6, 60, "toy")
employee(Nick, 3.2, 43, "shoe"
employee(Magda, 5.8, 47, "shoe")
employee(Chloe, 3.4, 46, "candy")
employee(Phoebe, 8.3, 31, "candy")
employee(Mike, 8.3, 70, "candy")

Figure below gives an example visualset that would be produced by applying the example metaphor above to this dataset. (The color of a point is indicated textually.) Clearly, real systems often deal with more complicated data and visual models and visual metaphors, but the underlying principles remain those highlighted in this exposition.

Key Applications

Visual representation is manifested in all applications with a (visual) user interface. Furthermore, having explicit declaration, storage, and manipulation of data and visual models as well as visual metaphors is essential to systems that offer to users the ability to modify the visual representations used, for reasons of personalization, quality control, or even plain variety.



Cross-references

- ▶ Data Visualization
- ▶ Visual Formalisms
- ▶ Visual Metaphor
- ▶ Visual Perception

Recommended Reading

1. Card S.K., Mackinlay J.D., and Shneiderman B. Information visualization. In Readings in Information Visualization: Using Vision to Think, 1999, pp. 1–34.
2. Card S.K., Mackinlay J.D., and Shneiderman B. Readings in Information Visualization: Using Vision to Think. Morgan Kaufman, Los Altos, CA, 1999.
3. Foley J.D., van Dam A., Feiner S.K., and Hughes J.F. Computer Graphics: Principles and Practice. Addison-Wesley, Reading, MA, 1990.
4. Haber E.M., Ioannidis Y., and Livny M. Foundations of visual metaphors for schema display. J. Intell. Inf. Syst., 3(3/4):263–298, 1994.
5. Mitchell W. Representation. In Critical Terms for Literary Study, Lentricchia F and McLaughlin T. (eds.), 2nd edn., Chicago, IL. University of Chicago Press, 1995.
6. Tufte E.R. The Visual Display of Quantitative Information. Graphics Press, Cheshire, CO, 1983.
7. Tufte E.R. Envisioning Information. Graphics Press, Cheshire, CO, 1990.

Visual Similarity

▶ Image Retrieval

Visual Web Data Extraction

▶ GUIs for Web Data Extraction

Visual Web Information Extraction

▶ GUIs for Web Data Extraction

Visualization for Information Retrieval

Jin Zhang
University of Wisconsin Milwaukee, Milwaukee, WI, USA

Definition

Visualization for information retrieval refers to a process that transforms the invisible abstract data and their semantic relationships in a data collection into a visible display and visualizes the internal retrieval processes for users [14]. Basically, visualization for information retrieval consists of two components: visually presenting objects in a meaningful way in a defined visual environment or space, and visualizing information seeking process within the environment or space. Transformation of the invisible and abstract information and sophisticated semantic connections into a visual environment requires clearly defining a visual environment or space, pinpointing the objects in a data collection which are displayed in the visual environment, identifying significant connections among the objects, and projecting both the objects and the relationships onto the environment. The form and way of visualizing the internal retrieval process vary in different visualization models. Visualization of the internal retrieval process depends not only on a defined visual environment, but also on the information seeking paradigms such as query searching and browsing.

Many traditional information retrieval evaluation models such as the distance evaluation model, cosine evaluation model, ellipse evaluation model, etc. can be visualized in a visual space. Furthermore, new information retrieval evaluation models can be developed in the visual space.

The primary advantage of visualization for information retrieval is to make objects, object relationships, and information seeking transparent to end users. As a result, information retrieval becomes more intuitive and effective.

Visualization methods can be applied to various information retrieval models such as the Boolean based information retrieval model, the vector-based information model, etc.

Historical Background

A traditional information retrieval system like an *OPAC* (Online Public Access Catalog) system and search engine usually matches a user's query with document surrogates in a database, and returns the user with a linear retrieval results list. The results list includes relevant items retrieved from the database. The results list may be ranked against titles, authors, publishing time, or similarities. In nature an information retrieval process is an iterative process. Users of the information retrieval system are supposed to use the results list to make the relevance judgment decisions, subject relevance analysis among the retrieved items, and readjust the search strategies. Towards this aim, an information retrieval system should not only provide users with relevance information between a query and the retrieved items, but also relevance information among the retrieved items, in the retrieval results list. The former information is apparently important for information retrieval. The latter information, which is as valuable as the former information, would facilitate the users to further expand, revise, and modify their search strategies. Unfortunately, if an information retrieval system is equipped with a similarity ranking mechanism for its linear results list structure, it only provides users with the relevance information between a query and the retrieved items. The inherent weakness of the linear result list structure cannot offer relevance information among the retrieved items, let alone the degree to which they are relevant.

In a traditional information retrieval system, the user's information seeking process is discontinuous. After transferring a user's information need into a query and submitting it to the system, the user loses control over the internal information processing. The user regains control after the retrieval results are returned to the user. The user has no clues about how the query is matched with document surrogates, and how the final retrieval decision is made. In other words, the internal retrieval processing is a "black box" for the end user. The users cannot observe the internal processing and engage in it. However, if the user could participate in the internal retrieval processing, it would make information seeking more user-friendly, relevance judgment decision-making more accurate, and the retrieval process more natural.

Information retrieval basically consists of two important fronts: browsing and querying. They are equally important for information seeking. Each has its own advantages and disadvantages. Neither can replace the other. In a traditional information retrieval context where querying dominates an information retrieval process, browsing cannot be fully utilized to explore information due to the lack of a meaningful information browsing environment. Visualization for information retrieval opens a new avenue for users to seek for information in the two fronts.

Korfhage [6], as a pioneer and leading researcher in the field of visualization for information retrieval, made a significant contribution to the early research in the field. He introduced the important concept of the reference point, which can be used as a projection reference point when objects in a high dimensional space are projected onto a low dimensional visual space. He visually interpreted an information evaluation retrieval model in a visual space, and came up with new visualization models for information retrieval.

Foundations

Models for Multiple Reference Points

A reference point (or point of interest) is introduced to represent users' information needs. In a broad sense, a query is a kind of a reference point. A reference point can reflect a particular perspective of a user's information need. It can be a subject topic, user's search preference, a previous query, or any information related to the user's information need. Since a document usually involves multiple facets, multiple reference points can reveal more information about the document from multiple perspectives.

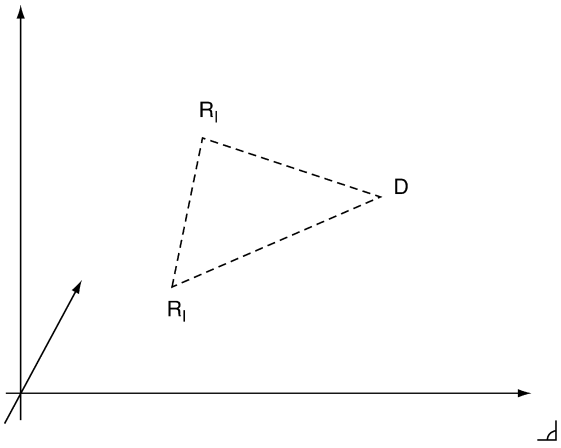
The visualization models for multiple reference points can be classified into two categories: models for fixed multiple reference points and models for movable reference points. In a model for fixed reference points [11] which is developed to visualize a sophisticated Boolean query, the vertices of a polygon represent the fixed reference points and the edges of the polygon have the same length. Within the polygon there are several concentric layers. The number of the concentric layers is equal to the number of the reference points or the number of vertices in the polygon. Each layer represents a different logic combination zone of the involved reference points. For instance, the first outer layer represents single reference point zone, the second outer layer represents the logic AND combination zone for two reference points, . . ., and the last central layer represents the logic AND combination zone for all reference points (or the final results of the query). Various icons which symbolize search results for the logic combination zones are located within the corresponding layers. The shape and the number of the icons vary in different layers so that they can be easily distinguished in the visual space. This multiple layer structure can display not only the final results of a query but also the itemized results at different combination levels. In addition, each vertex of the polygon can be defined as a sub-query. If that is the case, the vertex can extend to a new sub-polygon to visualize the sub-query results by connecting the vertex to the new sub-polygon. Thanks to this feature, the model can visualize a sophisticated Boolean query.

A model for movable reference points [9] works differently from the model for fixed reference points. It is apparent that in this model one of the most distinctive characteristics is that the involved reference points can be manipulated and placed in any place in the visual space by users. In other words, the reference points can be moved to a meaningful position. Because the model is developed based on the vector model instead of the Boolean model, the similarity between a document and any of the involved reference points can be accurately calculated by using a selected similarity measure. Basically, the location of a document in the visual space is affected directly by the strengths (similarities) between the document and all involved reference points. Consequently, after all reference points and documents are projected onto the visual space, the documents which are closer to a reference point are more relevant to the reference point. The uniqueness of the model relies on the fact that the strength between a document and a reference point is relative because the strength is measured by the ratio of the similarity between the document and the reference point to the similarities between the document and all reference points in the visual space. It is this relativity in conjunction with the fact that the similarities are not assigned to any coordinates of the visual space directly that makes the movability of the reference points possible in the visual space. That is, the positions of projected documents are determined by the locations of the reference points. The implication of the movability of the reference points on information retrieval is that users can select a reference point of interest, move it, and observe the object configuration change caused by the reference point movement in the visual space. If documents in the visual space are relevant to the moving reference point, they also move accordingly. Otherwise they stay still. Using this characteristic, users can effectively identify a group of related terms to a reference point in the visual space by selecting and moving it. This model can be applied to visualizing a returned results list from an information retrieval system, a full-text, and hyperlinks structures.

Euclidean Space Characteristics Based Models

Two of the most prominent characteristics of the Euclidean space are distance and direction. Both distance and direction are used to define and describe object locations in a space, and to demonstrate relationships among the objects in the space. When documents are organized in a vector space, they can be virtually described in a hyperspace where both the distance and direction characteristics are preserved. Furthermore, both the distance and direction have significance for information retrieval. It is natural and intuitive to utilize both distance and direction to develop information retrieval visualization models. By reducing the dimensionality of the high dimensional document vector space, spatial relationships among the documents in the vector space can be projected onto a low dimensional space to observe and manipulate the documents.

Euclidean space characteristics based models require two reference points to construct their low dimensional visual spaces. In the vector space if two reference points (R_1 and R_2) are clearly defined, a group of important parameters for a document (D) in the vector space can be calculated. The two visual distances (R_1D and R_2D) between the document and



Visualization for Information Retrieval. Figure 1. Display of the visual distances and visual sangles of a document.

the two reference points respectively, and the two visual angles ($\angle R_2R_1D$ and $\angle R_1R_2D$) formed by the three points (D, R_1 and R_2) can be calculated. For simplicity, D, R_1 and R_2 are displayed in a three dimensional space (see Fig. 1). If one of the two visual distances and one of the two visual angles are assigned to the Y-axis and X-axis, respectively, it constructs the visual space of the distance-angle based visualization model [15]. If the two angles are assigned to the Y-axis and X-axis, respectively, it constructs the visual space of the angle-angle based visualization model [13]. If the two distances are assigned to the Y-axis and X-axis, respectively, it constructs the visual space of the distance-distance based visualization model [8]. Observe that as long as these visual spaces are defined, any documents in the vector space can be effectively projected onto the visual spaces. The valid display areas vary in the models. They range from a close area to a semi-open area.

The uniqueness of these visualization models is reflected in the visualization of information retrieval evaluation models such as the distance evaluation model, cosine evaluation model, conjunction evaluation model, disjunction evaluation model, ellipse evaluation model, and oval evaluation model. In other words, they visualize the internal retrieval process in the visual spaces. For instance, in the vector space, the distance evaluation model corresponds to a hypersphere whose center is a query and radius is a retrieval threshold. Documents within the invisible sphere are regarded as the retrieved documents. This hyper sphere can be converted to a horizontal line in the distance-angle based low visual space if the query and the origin of the vector space are defined as the two reference points. That is because the distance from any point on the sphere to the center, which is one of the projection parameters, is a constant regardless of another projection parameter angle. Users can interact with the horizontal line in the visual space to control the size of the sphere in the vector space.

Visualization of Hierarchy Structures

As one important browsing mechanism, a hierarchy structure is widely used to organize and present a variety of information. A hierarchy structure can provide users with a structural categorical framework directing users to relevant information. Visualization techniques can enable users to navigate smoothly in a hierarchy structure by visualizing both global and local overviews of a hierarchy structure. Visualization technique enhances controllability and flexibility of information exploration and therefore alleviates the "disorientation" during navigation. ConeTree [10] is a visualization system that shows root, branch, and leaf nodes of a tree structure in a 3-D environment. Starting from the apex of a cone (the root of the tree), it fans out from left to right in a form of the cone. The child nodes are displayed along the cone base circumference which is a circle. The circle's diameter depends on the number of the child nodes on it. Any of these child nodes can extend to a new cone structure similar to its parent if the node has child nodes. In this way, an entire hierarchy structure can be presented visually. TreeMap [1] displays a hierarchy structure in a different way. Its visual space is a grid. A cell of the grid can include several structurally similar sub-grids. The nested sub-grids show the categories at different levels, each sub-grid corresponds to a new sub-category.

Visualization of Internet Information

The Web provides both challenge and opportunity for visualization for information retrieval. Information on the Web is huge, dynamic, diverse, and hyperlinked. It has become an indispensable source for people to search for information. One of the problems for information seeking on the Web is the notorious "lost in cyberspace" state. Visualization for information retrieval can alleviate, if not eliminate, the problem. Using information visualization techniques such as the hyperbolic method, people can effectively

visualize the hidden hyperlink relationships among the connected web pages. In such a visualization environment, web pages (nodes) are connected by edges (hyperlinks). Users can easily recognize the visited paths, revisit the browsed web pages, and explore new paths in the visual space. A hyperbolic method is employed to visualize a subject directory where nodes are linked by hyperlinks [4].

A search engine can respond to a user query with an overwhelming number of related web pages. It is difficult for users to browse and identify the relevant web pages from the returned list. Information visualization techniques provide a unique way to solve the problem. In a cartographic visual space the returned web pages from multiple search engines are metaphorically presented as cities and the semantic link between two cities as a road [5]. The interactive map facilitates the decision-making on the relevance judgment. Grokker [3] categorizes all retrieved web pages and presents them in a group of circles in its visual space. A circle which represents a category includes multiple smaller circles which are its sub-categories. Users can drill down from the top of a category to the bottom (the web pages) by selecting corresponding circles in the visual space.

It is worth pointing out that there are many other information visualization approaches such as multidimensional scaling analysis [12], pathfinder associative networks [2], self-organizing maps [7], etc. which can be applied to information retrieval. A detailed discussion of visualization for information retrieval, comparisons among various approaches and their implications for information retrieval are discussed in Zhang's monograph on the topic [14].

Cross-references

► Information Retrieval
► Information Visualization

Recommended Reading

1. Asahi T., Turo D., and Shneiderman B. Using treemaps to visualize the analytic hierarchy process. Inf. Syst. Res., 6(4):357–375, 1995.
2. Dearholt D.W. and Schvaneveldt R.W. In Pathfinder Associative Networks: Studies in Knowledge Organization, R.W. Schvaneveldt (ed.). Ablex Publishing, Norwood, NJ, 1990, pp. 1–30.
3. Grokker. Available online at: http://www.grokker.com/ (retrieved October 29, 2007).
4. Inxight. Available online at: http://www.inxight.com/ (retrieved on October 29, 2007).
5. Kartoo. Available online at: http://www.kartoo.com/ (retrieved on October 29, 2007).
6. Korfhage R.R. To see or not to see – is that the query? In Proc. 14th Annu. Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, 1991, pp. 134–141.
7. Lin X. Map displays for information retrieval. J. Am. Soc. Inf. Sci., 48(1):40–54, 1997.
8. Nuchprayoon A. and Korfhage R.R. GUIDO: a visual tool for retrieving documents. In Proc. 1994 IEEE Comp. Soc. Workshop on Visual Languages, 1994, pp. 64–71.
9. Olsen K.A., Korfhage R.R., Sochats K.M., Spring M.B., and Williams J.G. Visualization of a document collection: the VIBE system. Inf. Process. Manag., 29(1):69–81, 1993.
10. Robertson G.G., Card S.K., and Mackinlay J.D. Information visualization using 3D interactive animation. Commun. ACM, 36(4):57–71, 1993.
11. Spoerri A. InfoCrystal: a visual tool for information retrieval and management. In Proc. Second Int. Conf. on Information and Knowledge Management, 1993, pp. 11–20.
12. White H.D. Visualizing a discipline: an author co-citation analysis of information science, 1972–1995. J. Am. Soc. Inf. Sci. Technol., 49(4):327–355, 1998.
13. Zhang J. TOFIR: a tool of facilitating information retrieval – introduce a visual retrieval model. Inf. Process. Manag., 37(4):639–657, 2001.
14. Zhang J. Visualization for Information Retrieval. Springer, Berlin, 2008.
15. Zhang J. and Korfhage R.R. DARE: distance and angle retrieval environment: a tale of the two measures. J. Am. Soc. Inf. Sci., 50(9):779–787, 1999.

Visualization Pipeline

Helwig Hauser[1], Heidrun Schumann[2]
[1]University of Bergen, Bergen, Norway
[2]University of Rostock, Rostock, Germany

Synonyms

Visualization reference model

Definition

The *visualization pipeline* is a general model for a typical structure of a visualization process, which has been abstracted from earlier works in visualization in the late 1980s [3], and then adapted and extended several times [1, 2,...]. Starting out from *data* to be visualized and a particular visualization *task* at hand, a number of steps are processed along the visualization pipeline, including *data enhancement*, *visualization mapping*, and *rendering*, to eventually achieve a

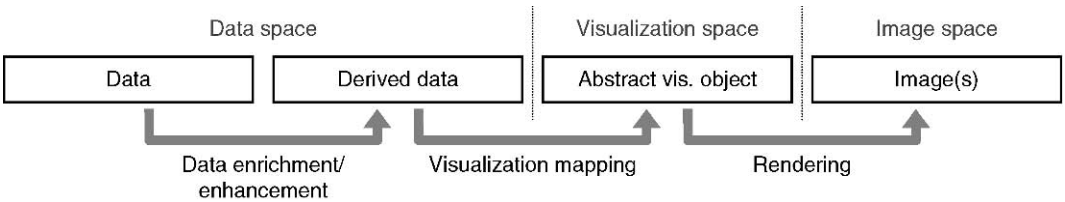
visualization of the data with the purpose of serving the given visualization task through effectiveness, expressiveness, and appropriateness [4].

Key Points

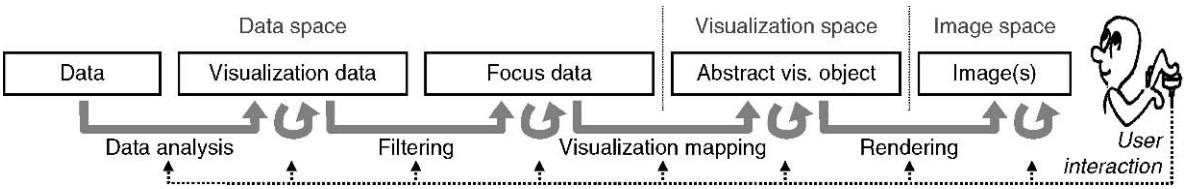
Data visualization is a part of computer science which provides expressive visual representations of data – through the appropriate application of computer graphics, often also in an interactive form – with the intention to effectively aid specific user tasks, including the exploration, analysis, and presentation of data. At this point only graphical data visualization is considered and other forms of data visualization such as sonification or the visualization through haptics are disregarded. Visualization establishes an efficient link between the user and her or his data, utilizing the enormous strengths of the human visual system with perception and cognition, and leading to a better understanding of the data through information extraction and knowledge crystallization.

To do so, the data is transformed from its original space, i.e., the *data space*, to an abstract *visualization space*, usually in two or three dimensions, where the visualization representation of the data is constructed. To eventually link up to the user, the visualization is rendered to an image or animation which is then displayed to the user. At least two mappings, i.e., the *visualization mapping* (from data space to visualization space) and *rendering* (from visualization space to image space) are required to fully accomplish this process. An example is the geometric representation of a 3D scalar data distribution by a discretized iso-surface, i.e., a mesh of triangles, which is then rendered to the screen by the use of computer graphics algorithms for visible surface detection and shading. In almost all cases, however, it is also necessary to first prepare and transform the data according to the needs of the visualization task at hand, such that visualization mapping and rendering can be applied afterwards. Haber and McNabb have already described this principal visualization model (as illustrated in Fig. 1) in 1990 [3].

About ten years after Haber and McNabb, Chi formulated the data state reference model for visualization [1], which formulates the visualization pipeline on the basis of transformation and stage operators, which – depending on their combination – result in various different possible instances of the visualization pipeline model. In 2004, dos Santos and Brodlie concluded that it is more meaningful to split the first step of the visualization pipeline into two [2]: *data analysis* and *filtering*. While the original data is initially prepared (once) for visualization, e.g., through interpolation, i.e., in a data-centric fashion, thereafter the user selects which data to actually visualize in the highly interactive and thus user-centric filtering step. Especially in the case of explorative visualization, where users search for priori unknown features in the data, the flexible interaction with the filtering step is crucial.



Visualization Pipeline. Figure 1. The visualization pipeline according to Haber and McNabb [3].



Visualization Pipeline. Figure 2. The visualization pipeline based on operators [1], with two steps in data space [2], and with user interaction.

The consideration of user interaction in general, altering operator parameters at all stages of the visualization pipeline, is an important aspect of the modern visualization pipeline (as also shown in Fig. 2).

Cross-references

► Data Visualization
► Visual Data Mining

Recommended Reading

1. Chi E.H. A taxonomy of visualization techniques using the data state reference model. In Proc. IEEE Symp. on Information Visualization, 2000, pp. 69–75.
2. dos Santos S. and Brodlie K. Gaining understanding of multivariate and multidimensional data through visualization. Computer & Graphics, 28(3):311–325, 2004.
3. Haber R.B. and McNabb D.A. Visualization idioms: A conceptual model for scientific visualization systems. In Visualization in Scientific Computing, G.M. Nielson, B. Shriver, L.J. Rosenblum (eds.). IEEE Computer Society, WA, USA, 1990, pp. 74–93.
4. Schumann H. and Müller W. Visualisierung – Grundlagen und allgemeine Methoden (in German). Springer-Verlag, Berlin, 2000.

Visualization Reference Model

► Visualization Pipeline

Visualizing Categorical Data

Ali Ünlü, Anatol Sargin
University of Augsburg, Augsburg, Germany

Synonyms

Visualizing categorical data; Graphics for discrete data; Visual displays of nonnumerical data; Plots for qualitative information

Definition

Categorical data are data recorded about units on variables which take values in a discrete set of categories. Examples of categorical variables are gender, citizenship, or number of children. Categorical variables can be dichotomous (two categories; e.g., gender) or polytomous (more than two categories; e.g., citizenship), and nominal (unordered categories; e.g., gender) or ordinal (ordered categories; e.g., number of children). Categorical data can be in case form (individual raw data vector recorded about each unit) or frequency form (tabulated data counting over the categories of the variables), and univariate or multivariate (including bivariate). Quantitative variables can be discretized to become categorical variables (e.g., using child and adult instead of exact age). Strictly speaking, all data may be considered categorical because of limited precision of measurement.

Visualizing categorical data, indeed visualization of data in general, seeks to (i) summarize the data, (ii) expose information and structure in the data, (iii) supplement the information available from analytic measures on the data, and (iv) suggest more adequate analytics for the data.

Key Points

Graphics for univariate categorical data are barcharts and stacked barcharts (for vertically drawn bars, all bars have the same width), and spineplots (modified barcharts where, for vertically drawn bars, all bars have the same height). The area of a bar represents the count for its category. Whereas (stacked) barcharts enable a good comparison of (cumulative) absolute counts, spineplots enable a direct comparison of proportions. When interactively highlighting a subgroup of units, spineplots allow visually comparing the proportions in different categories by looking at the heights of the highlighted areas [3]. Another popular, yet widely criticized, graphic is the pie chart (with partial, exploded, or perspective variants), for displaying shares.

Graphics for multivariate categorical data are mosaic plots and their variations equal binsize and doubledecker plots (for comparing highlighted proportions), fluctuation diagram (for identifying common combinations), and multiple barcharts (for comparing conditional distributions) [2]. Mosaic matrices, as a categorical analog of the scatterplot matrix, consist of all pairwise mosaic plots for two-way subtables [1]. Other powerful graphics are treemaps and trellis displays [2,3], for visualizing hierarchies and conditional structures, respectively. There are graphics specifically designed for bivariate categorical data, for instance, fourfold displays and sieve diagrams (parquet diagrams) [1]. For 2×2 tables, the fourfold display visualizes the association between variables in terms of the odds ratio, with confidence rings providing a visual test of whether

the odds ratio differs significantly from 1. Sieve diagrams provide displays of the pattern of association in general $r \times c$ tables.

Influence and diagnostic plots for such categorical data models as logistic regression, loglinear, and logit models (e.g., half-normal probability plots) provide examples of model-based visualizations of categorical data [1,2]. These plots help to evaluate model fit. Dimension reduction methods such as correspondence analysis and biplots provide visualizations of associations in n-way tables in a small number of dimensions [1].

Cross-references

- ▶ Data Visualization
- ▶ Multivariate Visualization Methods
- ▶ Visual Analytics
- ▶ Visualizing Quantitative Data

Recommended Reading

1. Friendly M. Visualizing Categorical Data. SAS Institute, Cary, NC, 2000.
2. Hofmann H. Mosaic plots and their variants. In Handbook of Data Visualization, C.H. Chen, W. Haerdle, A.R. Unwin (eds.). Springer, Berlin Heidelberg New York, 2008.
3. Unwin A.R., Theus M., and Hofmann H. Graphics of Large Datasets. Springer, Berlin Heidelberg New York, 2006.

Visualizing Clustering Results

Alexander Hinneburg
Martin-Luther-University Halle-Wittenberg, Halle/Saale, Germany

Synonyms

Dendrogram; Heat map

Definition

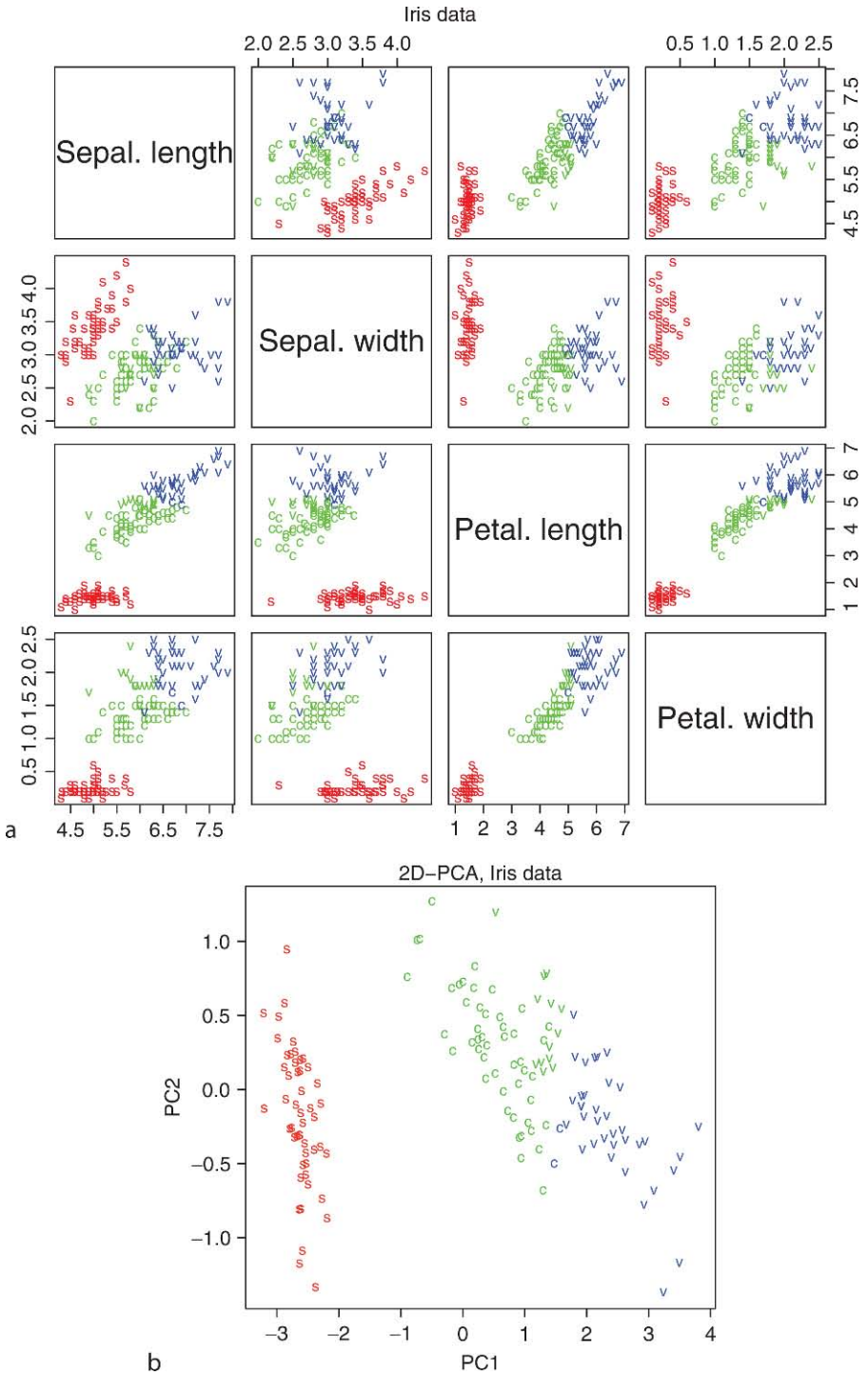
Visualizing clusters is a way to facilitate human experts in evaluating, exploring or interpreting the results of a cluster analysis. Clustering is an unsupervised learning technique, which groups a set of n data objects $D = \{x_1,...,x_n\}$ into clusters, so that objects in the same cluster are similar, and objects from different clusters are dissimilar to each other. The data can be available (i) as $(n \times n)$ matrix of similarities (or dissimilarities), and (ii) as $(n \times d)$ data matrix, which describes each data object by a d-dimensional vector. The second form has to be accompanied by a suitable similarity or dissimilarity measure, which computes for a pair of d-dimensional vectors a (dis)similarity score. A typical example of such a measure is the Euclidian metric. Clustering results may come in different forms: (i) as a partition of D, (ii) as a model, which summarizes properties of D and (iii) as a set of hierarchically nested partitions of D. Visualizations of those results focus one or several properties like the similarity relations between clusters and/or the underlying data objects, the components of the clustering model, or the representation of the data objects. Cluster visualization serves to determine semantics of clusters relevant to the application at hand by inspection, to check whether the cluster model and its parametrization (e.g., number of clusters) fits the data in a meaningful way, or to explore hierarchical relationships between clusters.

Historical Background

Clustering has been used as a technique to explore the structure of data. However, the results are abstract in nature, and in general the found cluster structure is not directly accessible to human experts as an overall picture. Data exploration is also facilitated by directly visualizing the data itself without clustering it.

Simple examples include the scatter plots technique of vector data, which shows 2D projections to all possible $d(d-1)/2$ pairwise attribute combinations. Independently computed results of partitioning clustering algorithms can be shown by labeling the points in the scatter plots according to cluster membership. An example is shown in Fig. 1a.

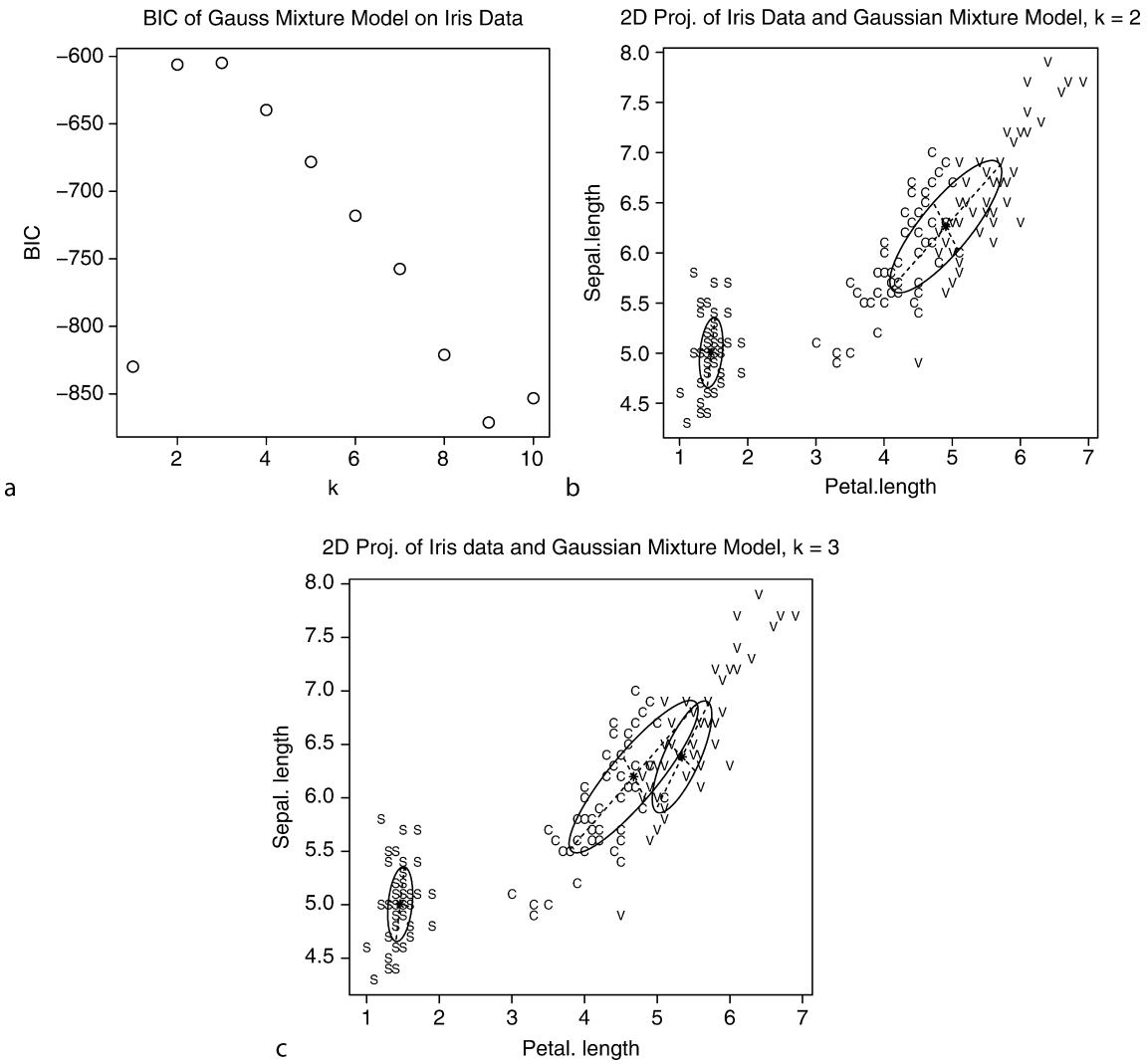
As the display of all 2D scatter plots requires quite a large view space, alternative techniques turn to dimensionality reduction to derive a single 2D projection of the vector data, which (approximately) optimizes some criterion. Dimensionality reduction techniques for data visualization include principle component analysis (PCA) using the first two principal components, classic multi-dimensional scaling (MDS), which performs PCA on the (dis)similarity matrix, isoMDS which minimizes the sum of squared differences between the original (dis)similarities and those in the projected 2D-space. New techniques of this sort include, FASTMAP [5] and latent probabilistic models like probabilistic and Bayesian PCA [4]. The optimization criteria differ from technique to technique but the results are always sets of 2D points. The main property is that



Visualizing Clustering Results. Figure 1. (a) Scatter plot the four-dimensional Iris data, (b) 2D-PCA of the Iris data. The classes Seritosa (S), Versicolor (C) and Virginica (V) are coded by letter and the membership of the clusters is coded by color *red*, *green*, *blue*.

dimensionality reduction and clustering are independently applied to the original data, and the results of both are combined in the visualization.

Another need for visualizing clustering results beside data exploration stems from cluster model evaluation and selection. Clustering methods require parameters, e.g., some algorithms need the number of clusters k as an input parameter. There is a large number of proposed validation measures to tackle the particular problem of choosing the right value for k. A typical example is the Bayesian information criterion (BIC), which balances the number of parameters with the model performance measured as likelihood. An example of BIC for a Gaussian mixture model applied to the Iris data is shown in Fig. 2a. Gaussian mixture models are a probabilistic extension of k-means. Note, that BIC gives no clear indication, that $k = 2$ or $k = 3$ is better. However, despite that measures like BIC are good indicators for model selection, they may also mislead the user [5] in cases where the model oversimplifies the underling data. In general, there is no broadly accepted measure, that solves the problem of model selection. So, cluster analysis often requires manual tuning of the algorithms to the application at hand. Visualization serves as a tool in the process of evaluating the performance of a clustering algorithm. However, due to its subjective nature, visualization is mainly used here as an auxiliary evaluation method.



Visualizing Clustering Results. Figure 2. (a) BIC of a Gaussian mixture model fitted to the Iris data for $k = 1,...,10$. The shown BIC-values are averages of 10 runs (higher BIC-values is better). (b,c) show clusterings for $k = 2,3$.

Foundations

A simple but powerful method to visualize the results of clustering is to visualize the cluster model itself. A straightforward example is *k*-means, which has been used to determined the clusters (shown by color) in Fig. 1a, b. Note, that in this case the cluster model, which consists of the cluster centers only, is shown indirectly by color coding the cluster membership. The cluster centers, which are the means of the clusters, can be visually approximated. An explicit visualization would be required to show the coordinates of the cluster centers and perhaps the decision boundaries of the clusters, which correspond to the edges of the Voronoi diagram induced by the cluster centers.

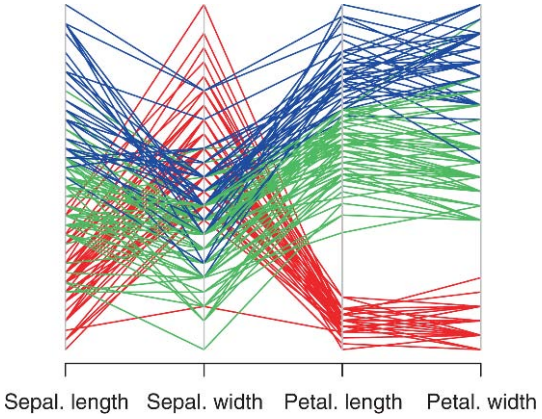
A more complex cluster model is the Gaussian mixture model, whose components consist of multi-dimensional Gaussian distributions. Such a distribution also has in addition to the center coordinates, parameters for the covariances between the dimensions. Figures 2b, c show some 2D-projections of the Gaussian models mixture models for two and three components fitted to the Iris data. Each component is visualized as an ellipse, whose center corresponds to the center of the Gaussian, and the orientation of the ellipse reflects the correlations between dimensions. Specifically, each ellipse shows the points which have the same probability density with respect to the corresponding Gaussian. The figures do not show the cluster memberships of the data points, because the Gaussian mixture model only gives a posterior distribution $P(c|\vec{x})$ for a point $\vec{x}$ and a cluster c, which allows several options for associating points to clusters. The most common method is to associate a point to the cluster with the maximum posterior. The shown visualizations combine both data and cluster models. However, there are many cases, when this scenario is not directly applicable. The reasons might be that the data does not allow a direct visualization in a scatter plot style, or the dimensionality of the vector data is too high to be easily reduced to 2D-projections by picking dimensions by hand. A typical example is document clustering, where the data objects are documents. Documents are either represented as sets of words or high dimensional word-count vectors. Both cases do not allow a straightforward two-dimensional visualization of the document collection.

A general method is parametric embedding (PE) [10], which takes the posterior distributions of the data objects with respect to some computed clusters, and fits a Gaussian mixture model of two-dimensional data points, so that the posteriors of the mixture model reflect the original posteriors. The model fitting of parametric embedding is done by minimizing the sum of Kullback–Leibler divergences between original posterior distributions and those from the two-dimensional Gaussian Mixture model. Note, that parametric embedding determines both the parameters for the two-dimensional Gaussian components of the mixture model, as well as the positions of the two-dimensional data points onto which the original data objects are mapped to. This technique allows the visualization the clustering results of most probabilistic clustering algorithms, regardless of the type and dimensionality of the data.

In the case where posterior probabilities are not available from the clustering algorithm, neighborhood component analysis (NCA) [8] can be applied for cluster visualization. NCA assumes vector data with given cluster (or class) labels and minimizes a stochastic version of the leave-one-one k-nearest-neighbor classification by learning a Mahalanobis metric. Therefore, NCA falls into the class of metric learning. Cluster visualization is achieved by restricting the rank of the learned Mahalanobis matrix to two or three. In such settings, the linearly transformed data can be visualized as scatter plots in two- or three-dimensional space.

An improvement of the PE technique is the combination of clustering and visualization [11], where the two-dimensional visualization acts like a regularizing prior distribution for the clustering model. The improvement is that the cluster membership values for each object are not fixed during the calculation of the visualization. So, both the parameter set for the clustering model and the parameters for the visualization are estimated simultaneously. In the case of high-dimensional data, the regularizing visualization component helps to avoid overfitting of the clustering model. Although, the experiments are done with a simple Gaussian mixture model (e.g., see Fig. 2), the improved technique could be used in combination with any probabilistic clustering model.

Instead of using scatter plots, which represent multi-dimensional data by two-dimensional points, parallel coordinates [1] can be used, which have no need for dimensionality reduction (e.g., see Fig. 3). Parallel coordinates visualize a multi-dimensional vector as a sequence of consecutive line segments, and visually scale up to 15 dimensions. The coordinates



Visualizing Clustering Results. Figure 3. K means clustering ($k = 3$) of the iris data shown in parallel coordinates.

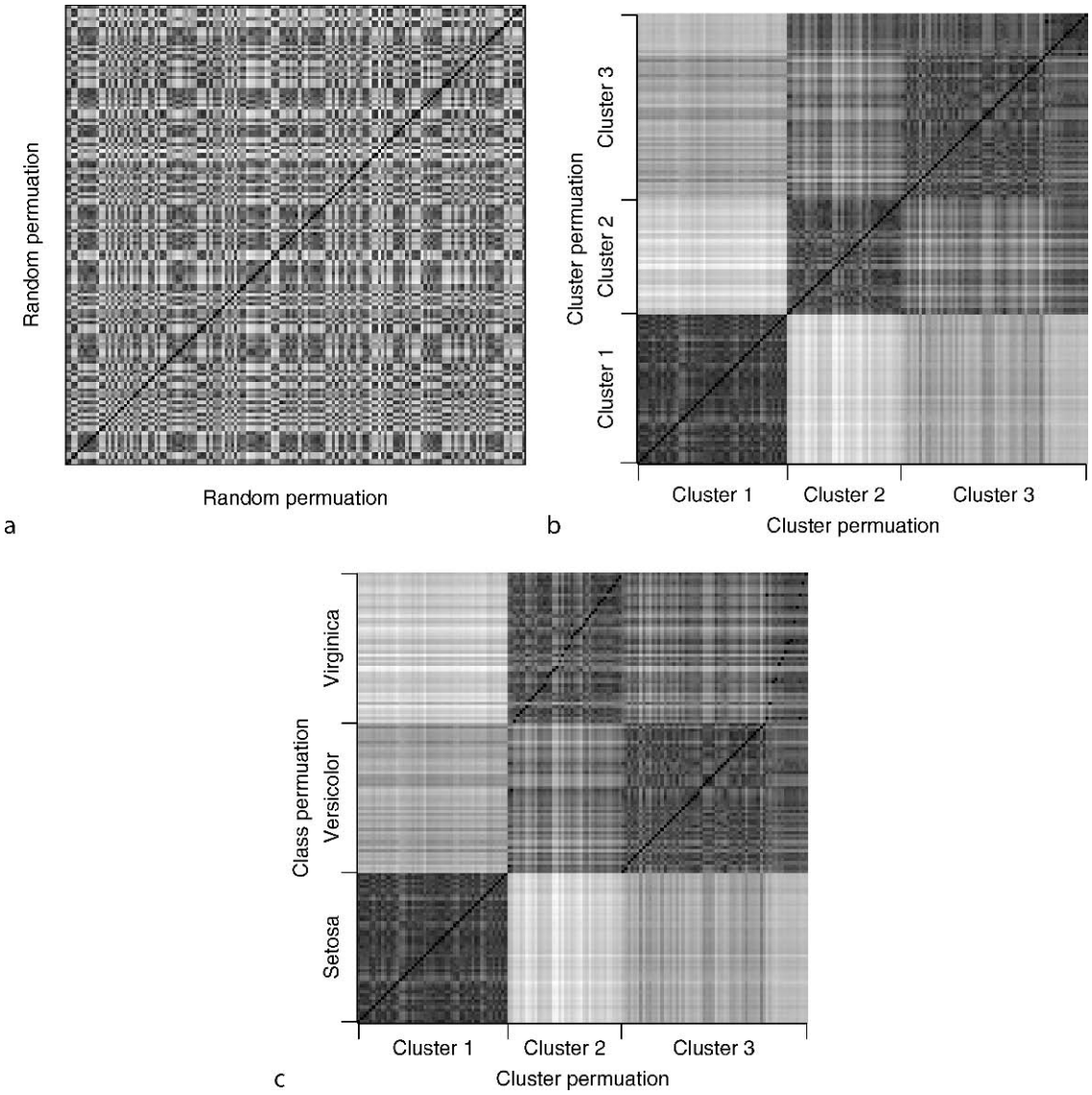
axes are represented by vertical lines. Simple visualizations show the clustering by color coding the cluster membership. A more advanced technique is proposed in [7]. Cluster centers are visualized by parallel coordinates. Additionally, the spans of the clusters projected to the attributes are also visualized by drawing opaque tubes of varying thickness around the line segments of the cluster centers.

So far, clusters have been visualized by showing a possibly transformed version of the data objects itself, together with some information about the clustering and/or the cluster model. A principle alternative to that approach is to visualize distance or similarity relations between the data objects and/or the derived clusters. In the next paragraphs, we explain the methods wrt. to object distances to make the discussion simpler, however, similarity measures could be used as well. The amount of distances between n data objects scales quadratically in n, which makes it difficult to visualize all pairwise distances of large data sets (e.g., $n \gg 1{,}000$). On the other side, visualizing distances or similarities is applicable to a much wider spectrum of data types, in contrast to the previous approaches, which assumed a given or derived representation of the data objects as vectors. Thus, visualizing relative information is attractive for looking at clusterings of sequences, trees, graphs, sets, and other non-vector like data.

The direct visualization of the distance matrix $D \in \mathbb{R}^{n \times n}$ of a data set codes the values of a matrix entry d_{ij} by color or gray value of a pixels, while the two indices i and j are coded by the pixel's position. In order to visualize a clustering, the columns and rows are partially ordered into blocks according to cluster membership. In a case where the clustering is meaningful, the change of the column and row permutation from a random to the partial ordering induced by the clustering has a huge visual impact. In the ideal case, clusters form blocks which are placed along the diagonal of the matrix. Possible relations between clusters show up as off-diagonal blocks in such a plot. This technique also allows the comparison of a clustering with some ground truth, if available, by ordering the rows and columns according to clustering and class labels, respectively. Examples are shown in Fig. 4. As the distance values sometimes have some skewed distribution, it improves the visual results to apply a monotone sub(super)-linear transformation to the distance values before mapping them to color to increase (decrease) the visual contrast. In Fig. 4, the square roots of the distances are shown. Non-clustering methods to find a good permutation of the distance matrix are discussed in [9] under the term seriation.

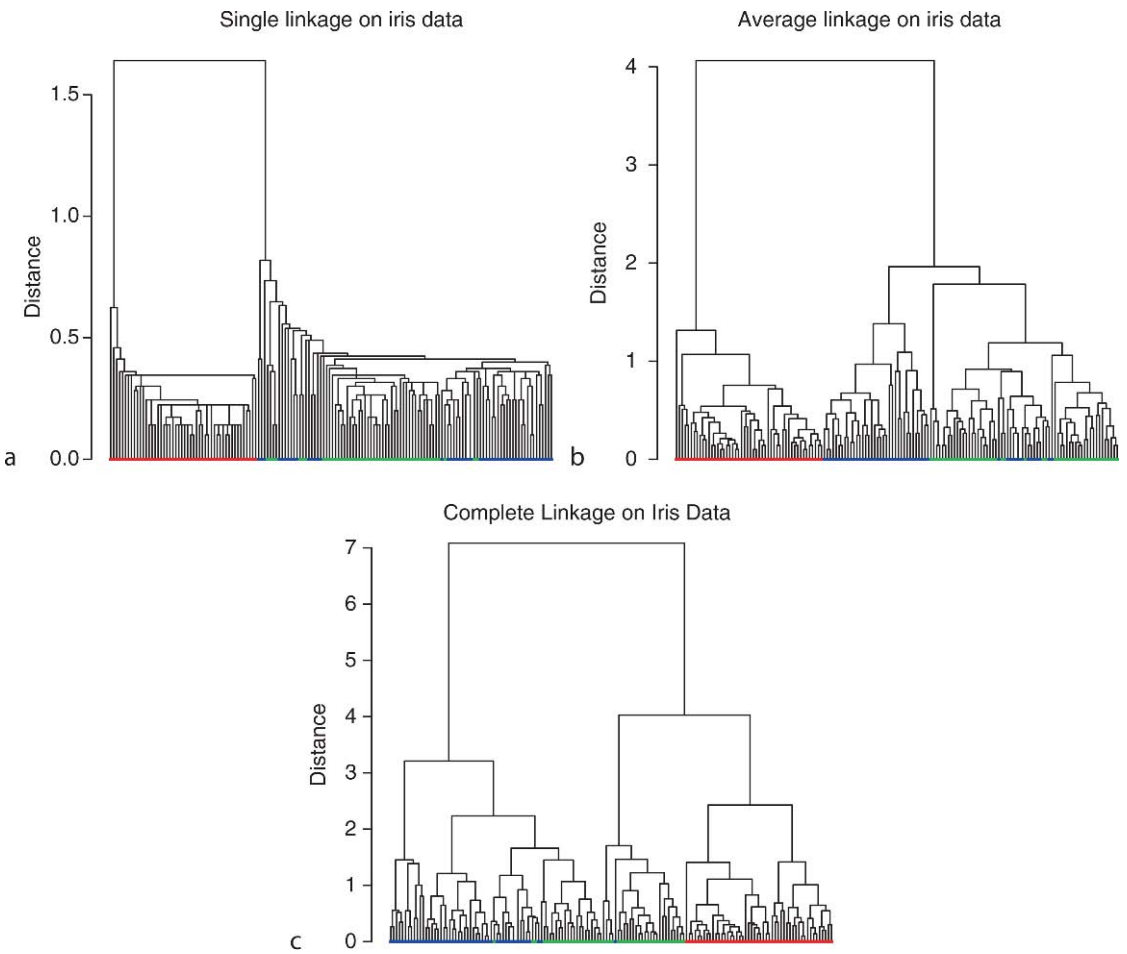
Hierarchical clustering is one of the earliest clustering methods. The generic, agglomerative, bottom-up algorithm starts with a clustering where each data object is its own cluster. In each step, the closest pair of clusters is found (i.e., the pair of clusters with smallest distance) and merged to form a new cluster. The distance between clusters C and C′ is induced by the distance between objects in different ways, e.g., single linkage $d(C, C') = \min_{x \epsilon C, x' \epsilon C'} d(x, x')$, average linkage $d(C, C') = 1/|C| \cdot |C'| \Sigma_{x \epsilon C} \Sigma_{x' \epsilon C'} d(x, x')$ or complete linkage $d(C, C') = \min_{x \epsilon C, x' \epsilon C'} d(x, x')$. After $n - 1$ steps, a hierarchy of nested clusterings is determined, which is represented as a binary tree. The visualization of such a tree is called a dendrogram, when the height of an inner node is the distance between the merged clusters. Leaf nodes have zero heights. The height of a complete linkage dendrogram is the maximal distance between two objects, the height of a single linkage dendrogram is the maximal edge in the minimal spanning tree of the data, and the height of an average linkage dendrogram is something between the two other dendrograms. Figure 5 shows different dendrograms of the iris data.

Overall, a dendrogram shows $n - 1$ distances between clusters, which is much less distance information than the original distance matrix. So, a dendrogram can be seen as a lossy compressed form of



Visualizing Clustering Results. Figure 4. Distance matrix of the iris data: (a) in random order, (b) columns and rows are ordered according to a k-means clustering with $k = 3$, and (c) rows are ordered according to a k-means clustering with $k = 3$ and columns are ordered by ground truth class labels.

the distance matrix. The number of possible visualizations of the same dendrogram is 2^{n-1}, which can be seen by noting that the orientation of each inner node has two possible states. So, the super-exponential problem of finding a good visualization of the similarity matrix among $n!$ possibilities, is reduced to the exponential problem of finding a good dendrogram visualization among 2^{n-1}. Typically, heuristics are used to find a useful orientation of a dendrogram, but in [3,12] more principled optimization techniques are used to that end. The derived permutation of the leafs, which correspond to the data objects, is also used as a meaningful ordering of the rows and columns of the distance matrix [14]. In the special case of vector data, a similar visualization is derived by hierarchically clustering the rows and columns of the data matrix, which is built by concatenating the d-dimensional data vectors $x_1,...,x_n$ to a $n \times d$ matrix. The respective dendrograms for the row and the column set are shown at the borders of the matrix. Both variants are called heatmap. Examples of the iris data are shown in Fig. 6. These kinds of



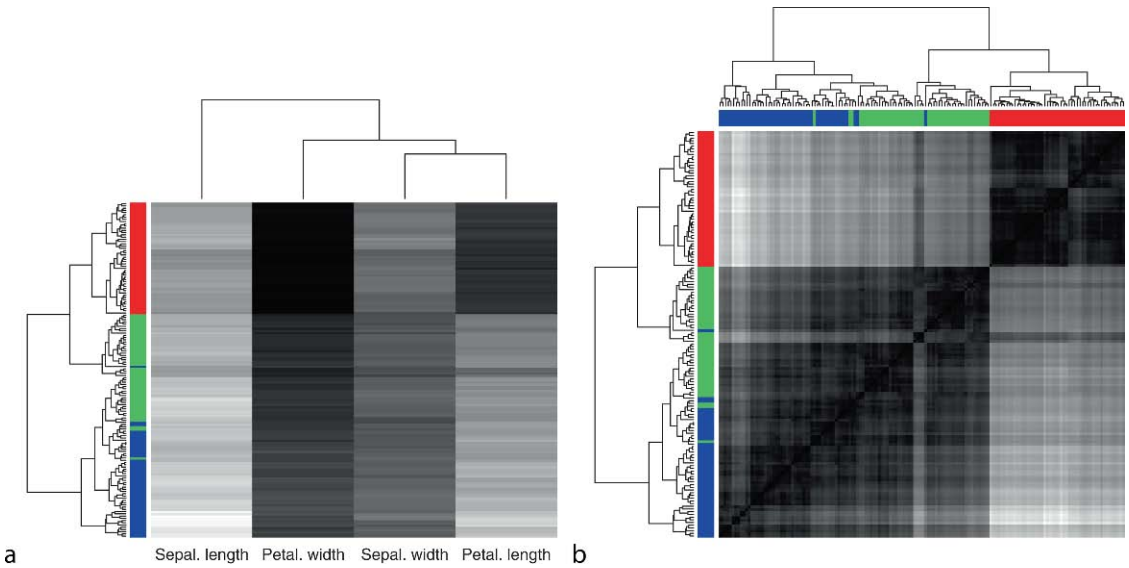
Visualizing Clustering Results. Figure 5. Dendrograms of the Iris data.

visualizations are especially popular in gene expression analysis.

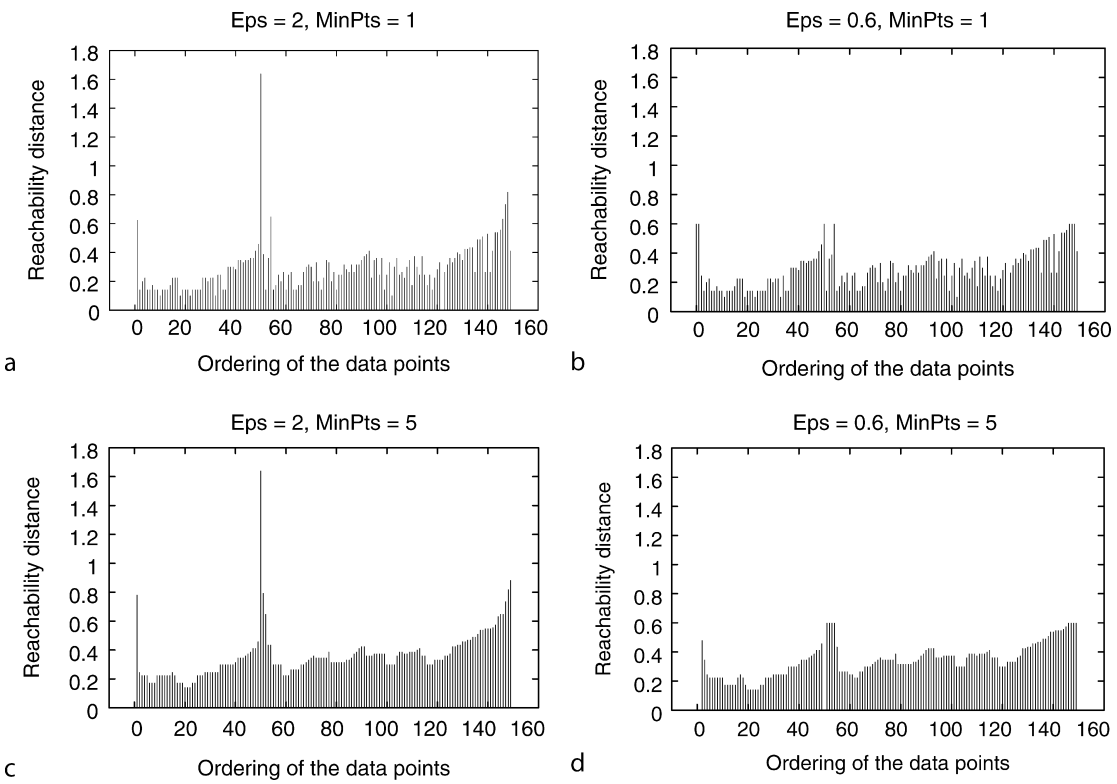
As the width of a dendrogram scales linearly with the data size, the clustering of a large data set is difficult to visualize with that technique. Therefore, heuristics are used to compute a horizontal cut in the dendrogram, and only the upper part is visualized. However, cutting at a constant height does not always produce the best result, thus new more flexible heuristics are discussed in [13].

In the case of single-linkage-like cluster hierarchies, the OPTICS approach [2] visualizes the lower part of the hierarchy with an alternative technique. OPTICS extends flat single linkage clustering. For visualization purposes it computes an ordering of the data points, which reveals nested clusters.

Flat single linkage clustering sees the data as a graph, with the data objects as nodes. An edge connects two nodes, when the corresponding data objects are closer than a given threshold (*EPS*). Clusters are the connected components of the graph. A drawback of single linkage clustering is that well separated clusters may be connected by chains of outliers and are merged by the algorithm. Therefore, OPTICS extends the basic single linkage approach by introducing an additional linkage constraint, the core object condition. The condition says that every object in a cluster is linked to at least one core object. A core object has at least *MINPTS* ≥ 1 other data objects in the neighborhood of a radius *EPS*. Graph theory says that connected components can be found by two equivalent algorithms, namely depth first search (DFS) and breath first search (BFS). BFS is mostly implemented by a first in first out (FIFO) queue. OPTICS uses a priority queue instead of a simple FIFO queue, which sorts the internal objects by ascending reachability distance. The reachability



Visualizing Clustering Results. Figure 6. Heatmaps of the Iris data showing (a) the data matrix and (b) the Euclidian distance matrix. The used dendrograms are computed by complete linkage.



Visualizing Clustering Results. Figure 7. OPTICS on Iris data.

distance of an object is defined as the distance to the *MINPTS*-nearest neighbor in the case of core objects, as the distance to the nearest core object in the case of border objects, which are linked to a core object, and infinity in the case of outliers. A good intuition of the reachability distance is to think of it as the inverse of data density. Objects with a low reachability distance have many neighbors, thus are in areas of high density. As the algorithm uses a priority queue, which outputs objects with a low reachability distance first, it focuses

on objects in high density areas first. The mode of operation of OPTICS is that it starts randomly somewhere in a cluster, is then lead to the clusters center and works its way to the border. After a cluster is finished, it starts with the next one in the same manner.

The visualization technique of OPTICS displays the objects in the order they are processed by the algorithm, and shows the reachability distance for each object. By the mode of operation of OPTICS, this methods shows a cluster as a valley. The left border of the valley is formed by the random start, which is with high probability an object with some larger reachability distance. The bottom of the valley is formed by the center objects of the cluster, which have the smallest reachability distance. The right border of a valley is formed by the border objects of the cluster. In that case, the cluster is perfectly separated from the rest of the data, the priory queue becomes empty and the algorithm jumps randomly to some still unlabeled object and processes the next cluster. Nested cluster structure shows up as dents in the bottom of a larger valley. As OPTICS never links objects further away than *EPS*, it does not determine links between clusters, which are separated by more than *EPS*. Thus, the visualization shows only the lower part of the hierarchy. Examples of OPTICS plots are shown in Fig. 7.

Key Applications

The visualization of clustering results is mainly used for interpreting the results of clustering and presenting them to people from the application side. It is a good tool to put the derived clusters into the context of the application at hand by annotating the visualization with meta-information. Visualizing clustering results also facilitates the exploration of the data and the tuning of clustering algorithms.

URL to Code

The images in this article (except the OPTICS plots) are produced by the statistics package R. The code for the drawings can be found at http://www.informatik.uni-halle.de/~hinnebur/clusterVis.tar.gz.

Cross-references

- ▶ Clustering Overview and Applications
- ▶ Clustering Validity
- ▶ Visual Classification
- ▶ Visual Clustering
- ▶ Visual Data Mining

Recommended Reading

1. Alfred Inselberg. Parallel coordinates: VISUAL multidimensional geometry and its applications, Spring, 2007.
2. Ankerst M., Breunig M.M., Kriegel H.-P., and Sander J. Optics: Ordering points to identify the clustering structure. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1999, pp. 49–60.
3. Bar-Joseph Z., Gifford D.K., and Jaakkola T.S. Fast optimal leaf ordering for hierarchical clustering. Bioinformatics, 17(90001):22–29, 2001.
4. Bishop C. Pattern Classification and Machine Learning. Springer, New York, 2006.
5. Domingos P. Occam's two razors: The sharp and the blunt. In Proc. 4th Int. Conf. on Knowledge Discovery and Data Mining, 1998, pp. 37–43.
6. Faloutsos C. and Lin K.I. Fastmap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1995, pp. 163–174.
7. Fua Y.-H., Rundensteiner E.A., and Ward M.O. Hierarchical parallel coordinates for visualizing large multivariate data sets. In Proc. IEEE Conf. on Visualization, 1999.
8. Goldberger J., Roweis S.T., Hinton G.T., and Salakhutdinov R. Neighbourhood components analysis. In Advances in Neural Inf. Proc. Syst. 18, Proc. Neural Inf. Proc. Syst., 2005, pp. 513–520.
9. Hahsler M., Hornik K., and Buchta C. Getting Things in Order: An introduction to the R package seriation. http//:cran.at.r-project.org/web/packages/seriation/vignettes/seriation.pdf.
10. Iwata T., Saito K., Ueda N., Stromsten S., Griffiths T.L., and Tenenbaum J.B. Parametric embedding for class visualization. Neural Comput., 19(9):2536–2556, 2007.
11. Kaban A., Sun J., Raychaudhury S., and Nolan L. On class visualisation for high dimensional data: Exploring scientific data sets. In Proc. 9th Int. Conf. on Discovery Science, 2006.
12. Koren Y. and Harel D. A two-way visualization method for clustered data. In Proc. 9th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, 2003, pp. 589–594.
13. Langfelder P., Zhang B., and Horvath S. Defining clusters from a hierarchical cluster tree: the Dynamic Tree Cut package for R. Bioinformatics, 24(5):719–720, 2008.
14. Strehl A. and Ghosh J. Relationship-Based Clustering and Visualization for High-Dimensional Data Mining. INFORMS J. Comput., 15(2):208–230, 2003.

Visualizing Hierarchical Data

Graham Wills
SPSS Inc., Chicago, IL, USA

Synonyms

Hierarchical graph layout; Visualizing trees; Tree drawing; Information visualization on hierarchies; Hierarchical visualization; Multi-level visualization

Definition

Hierarchical data is data that can be arranged in the form of a tree. Each item of data defines a node in the tree, and each node may have a collection of other nodes as *child nodes*. The relationship between the parent nodes and the child nodes forms a *tree network*. The formal definition of a tree is that the graph formed by the nodes and edges (defined between parent and child node) is both connected and contains no cycles. The following properties of a tree are of more practical use from the point of view of displaying visualizations:

- One node, called the *root* node, has no parent.
- All other nodes have exactly one parent.
- Nodes with no children are termed *leaf* nodes. Nodes with children are termed *interior* nodes.
- For all nodes in a tree, there is a single unique path up the tree going from parent to parent's parent and so on, which will terminate in the *root* node.
- The number of nodes on the path from a node to the root is termed its *depth*.

Although not strictly required, the vast majority of hierarchical data, and the main application area, consists of trees where the parent nodes define some form of aggregation on the child nodes, so that the data for the parent is equal to, or is expected to be close to, the aggregation of the data for that node's children. The relationship between parent and child is often described in terms of inclusion; it is common to state that nodes *contain* their children. This aspect is often brought out in visualizations.

A simple example of hierarchical data would consist of populations for the world, hierarchically broken down into sub-regions. At the top level, the root would consist of the world, with data being the population of the world. The next level could be the four main continents, each with their individual population, and each continent would have countries as children, each with their population counts. In each case, one would expect that the population of the parent node would roughly equal the population of the child nodes. If the data were collected at slightly different times, or from different sources, the populations for the continents might not exactly equal the sum of their children's populations, but one would expect it to be very close.

Note that in this example, every *leaf* node has the same *depth*. Although this is a common property in many applications, it is not a required property. In fact, if Antarctica is counted as a continent, it would have no contained countries, and the hierarchy would lose this property. If the level of the hierarchy is further increased, by dividing countries into regions, one would have very different levels. The United States might be divided into states and then into zip codes, whereas Vatican City would not be divided up at all.

Historical Background

Leonhard Euler is regarded as the founder of graph theory, publishing initially in the 1730s, but displays of trees as predate even this, especially displays of family trees, some of which survive from significantly earlier. However, the discipline of visualizing hierarchical data in a systematic fashion dates back only to the availability of computers. In 1992, the conference *International Symposium on Graph Drawing* commenced meeting and their conference proceedings, available from 1994, provide an excellent overall reference to the state of the art in this subject, although limited mainly to static graphs. From around 1990 onwards, increased attention has been paid to *interactive*, or *dynamic* visualization of hierarchical data, although information on such techniques is scattered across many disciplines. User interface controls for interacting with hierarchies became common with the advent of windowed operating systems; trees of folders and files are commonplace and techniques for filtering and pruning such views have seen significant research and user testing.

Foundations

There are a number of reasons why hierarchical data might be visualized, and it is important to identify the goal of any visualization, as different visualization techniques serve different goals. Common reasons to view hierarchies are:

- *Understanding Structure*: The goal is to understand the structure of the hierarchy; in the world population example, one might want to know how countries are distributed within continents, or to see how many major regions are within countries.
- *Understanding Data*: The goal is to understand the distribution of data across the hierarchy. In the example, one might want to know if each continent has a similar distribution of population, or if the lowest levels have similar populations (do zip codes

in the US, on average, have the same populations as the Vatican City?), or similar questions. Often, the goal is to understand the data within the context of the structure.

- *Summarizing large amounts of data*: The goal is to reduce information overload by providing a summary of low-level data into aggregated data. A hierarchy allows the user of a visualization to set the level of detail they want, by only showing data up to a certain level in the hierarchy.

There are two basic branches of visualization techniques for hierarchies. The first is based on a node-edge graph-layout approach which focuses attention on the structure and relationships, and the second on space-filling approaches, which focus attention on the relative sizes of nodes in the hierarchy. Each is discussed below.

Node-Edge Layouts

These displays are essentially a specialization of general graph layout techniques. Each node in the hierarchy is displayed as a small glyph, commonly a circle or a square. Data on the node can be represented by changing aesthetic attributes of the node such as its size, color, pattern, etc. Fig. 1 below gives an example of a traditional node-edge display for hierarchical data. Each node represents a country, and they are aggregated into a hierarchy where the net level up is a sociological grouping. The populations have been aggregated by a mean average, so the node representing the "new world" countries has been given the mean population of countries in the new world. It is clear that there is little difference between populations of countries when they are aggregated into these groupings.

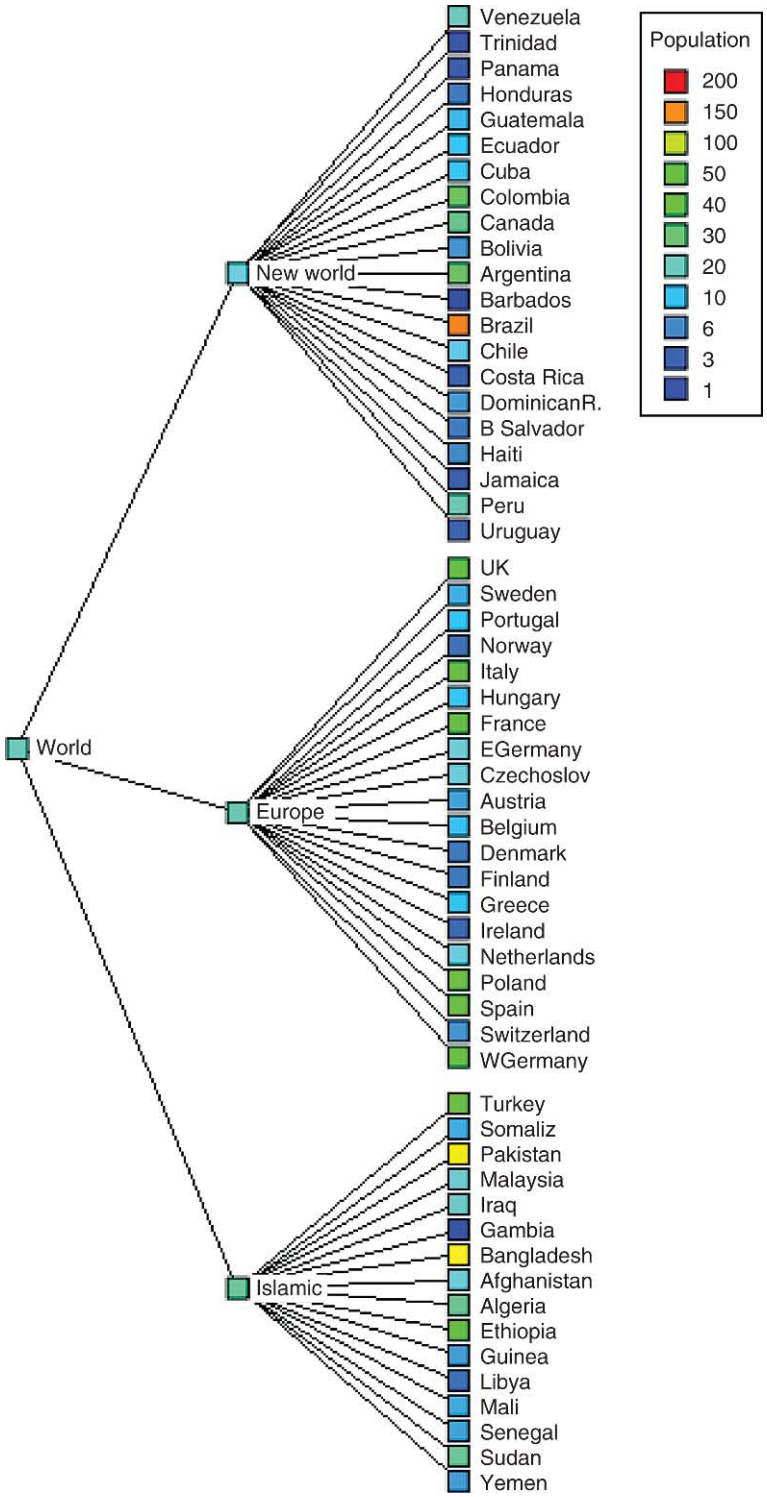
This visualization has the advantage that it is good for showing the *structure* of the hierarchy – one can see clearly how each many countries are in each group, and the distribution of the variable of interest. However, the display is not compact – a lot of space is wasted showing links having little information. Modifications to this basic display include the following:

- *Ordering the nodes within each parent.* The child nodes for each parent could be sorted by a variable. In Fig. 1, it would make sense to sort by the population to create a better overview of the distribution of that variable across countries within a group.
- *Displaying information on the edges.* In the above figure, the edges could be coded according to the similarity between the country and its group, using any of a number of statistical techniques to define such similarity. One such technique starts simply with data items and then generates a hierarchy by successively clustering items into groups based on similarity. This technique is called hierarchical clustering and is often visualized using a dendrogram as in Fig. 2. The dendrogram shows when groups are formed using the vertical dimension; the lower on the vertical dimension, the more similar the groups that were merged were. In Fig. 2, F and G were merged first and because they care the most similar pair, then A and B were formed into a group, then D and E. Then C was added to {A, B}, following which {D, E} was merged with {F, G}. The resulting groups {A, B, C}, {D, E, F, G}, {H} are only merged together at a much higher level of dissimilarity. The dendrogram therefore lets one see not only what clusters were created, but it also, by placing the merge information at a location in the vertical dimension proportional to the similarity of the groups being merged, allows one to see how good the resulting clusters are.

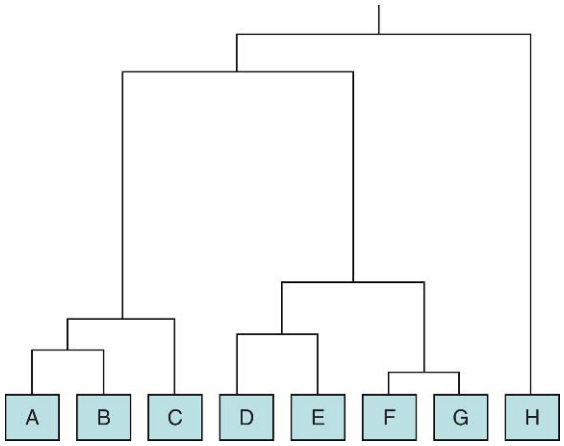
Space-Filling Layouts

In contrast to node-edge layouts, space-filling layouts take explicit advantage of the hierarchical nature of data directly. A space-filling layout is usable only when the main variable of interest is *summable*. As these techniques lay out areas in proportion to sizes, and parents visually include their children, a variable that can be summed is necessary. It is not possible to use space-filling layouts directly to show means, minimums or similar statistics. The base layout is limited to sums. It is, of course, possible to color or use some other non-size aesthetic for such techniques, but then the essential space-filling nature of the display is of little value.

Figure 3 shows the example data with a radial space-filling layout. Each level in the hierarchy is represented by a band at a set radius, with the root note in the center, and children outside their parents. The angle subtended by a node is proportional to the percentage of the entire population that this node represents. Children are laid out directly outside their parents, so parents "divide up" the space for their children according to the child sizes.. This is a typical



Visualizing Hierarchical Data. Figure 1. Standard Node-Edge layout for a hierarchical network.



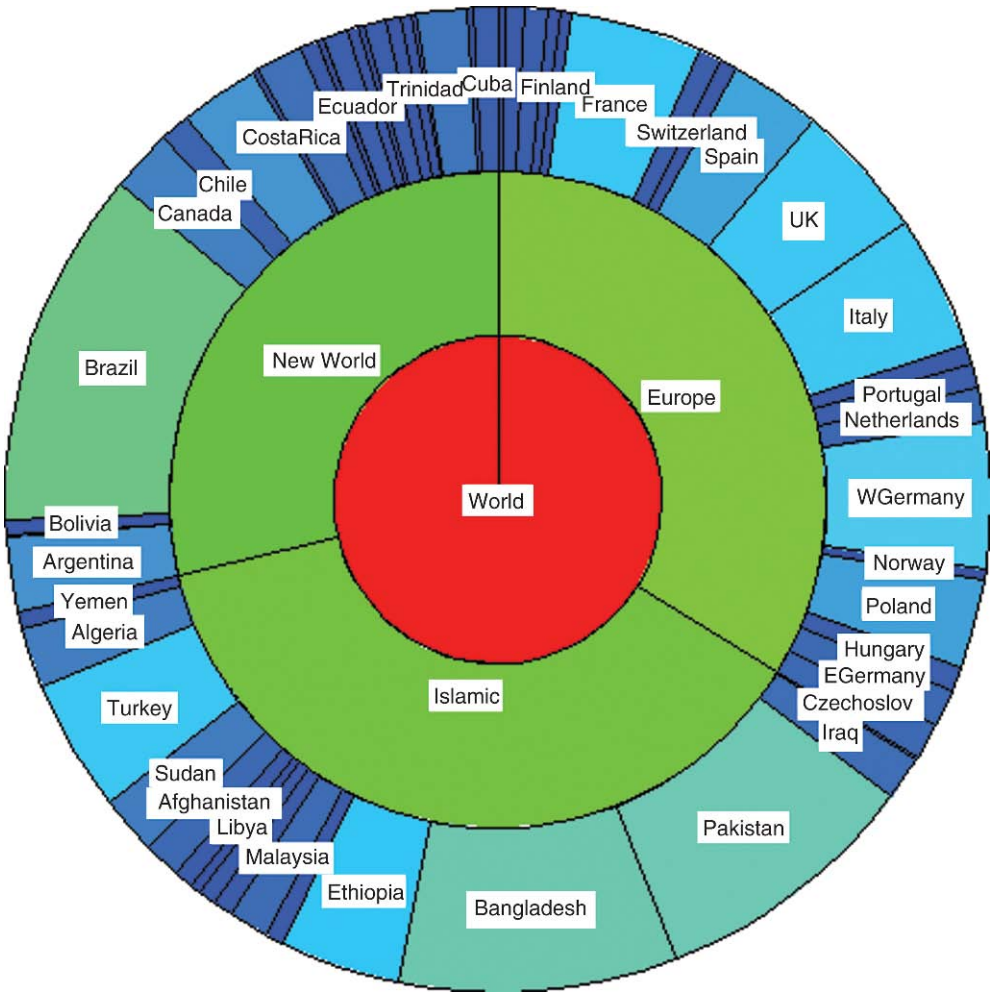
Visualizing Hierarchical Data. Figure 2. Dendrogram of a hierarchical clustering.

space-filling technique, many of which have been discovered in various disciplines. They share the following characteristics:

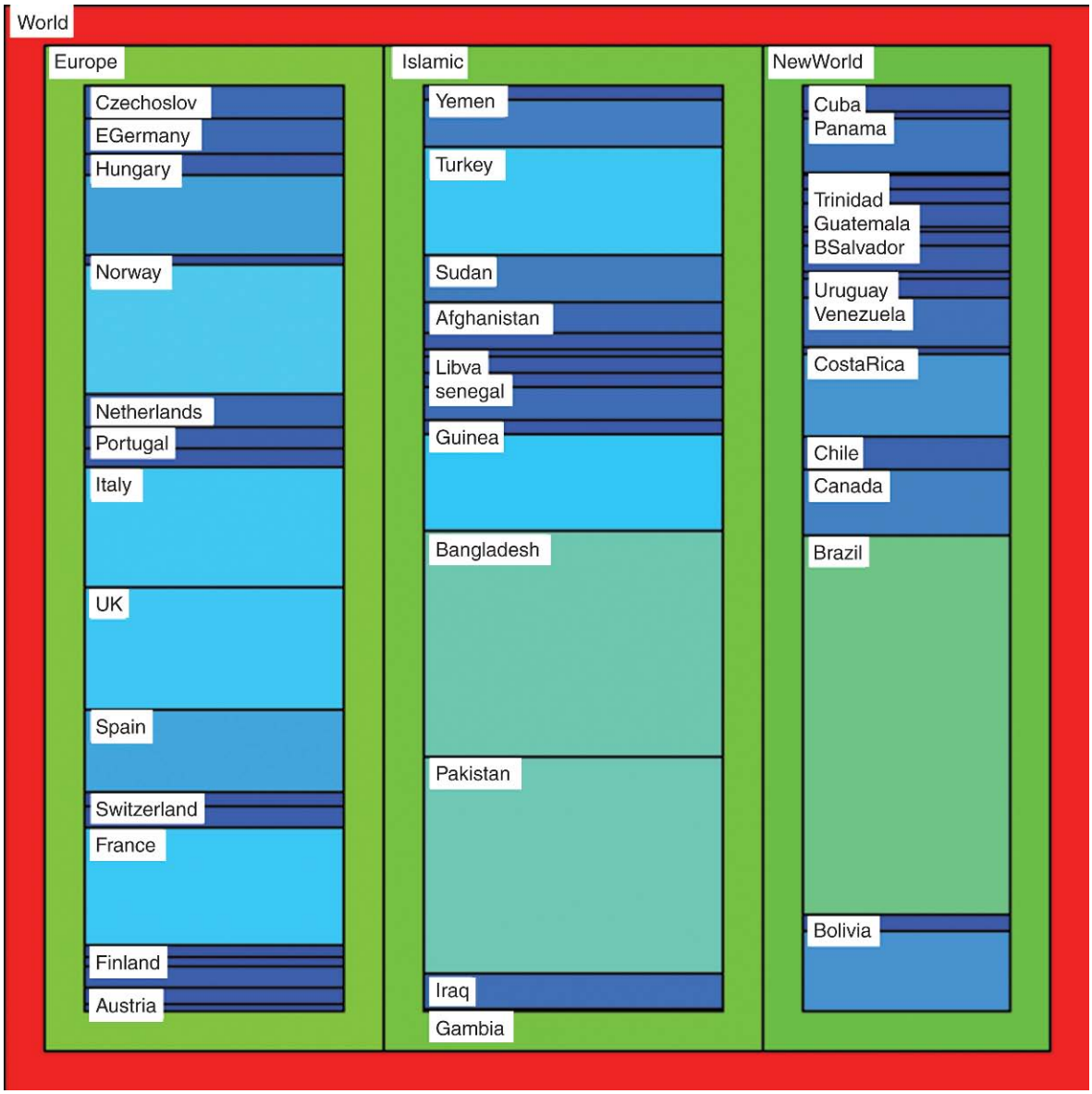
- The root node occupies 100% of the space, or dimension of interest (in this case, the radial dimension).
- Each node partitions its space according to the relative sizes of its children.

Figure 3 uses space more economically than Fig. 1, and also allows one to see sizes more clearly. It is generally to be preferred if the data support it. Compare this figure with Fig. 4, which uses a style of layout popularized under the name "TreeMap."

The treemap, instead of allocating space for child nodes outside the parent node, lays them out directly



Visualizing Hierarchical Data. Figure 3. The Population data of Fig. 1, this time showing summed population, using a space-filling radial layout.



Visualizing Hierarchical Data. Figure 4. TreeMap version of Fig. 3.

on top of the parent, thus making a maximally compact representation. This causes the immediate problem that there is then no way to distinguish the tree structure, since the children completely occlude the parents. Typically, as in Fig. 4, a small border is left around the child nodes to allow the tree structure to be ascertained, although that does distort the overall relationship between visual size and the sizes of the hierarchical data points. The TreeMap is not a good technique for learning about *structure* because of this issue, but because it is very compact it can perform quite well in the domain to which it is applicable: displaying relative sizes of items in the hierarchy when the structure of the hierarchy is well-known or of little interest. There are numerous different ways to render this figure, and care should be taken to avoid situations where some rectangles are skinny and others are flat; it is harder to make size judgments based on areas with widely differing aspect ratios than it is on arcs with different angles as in Fig. 3, for example. In general, the TreeMap should

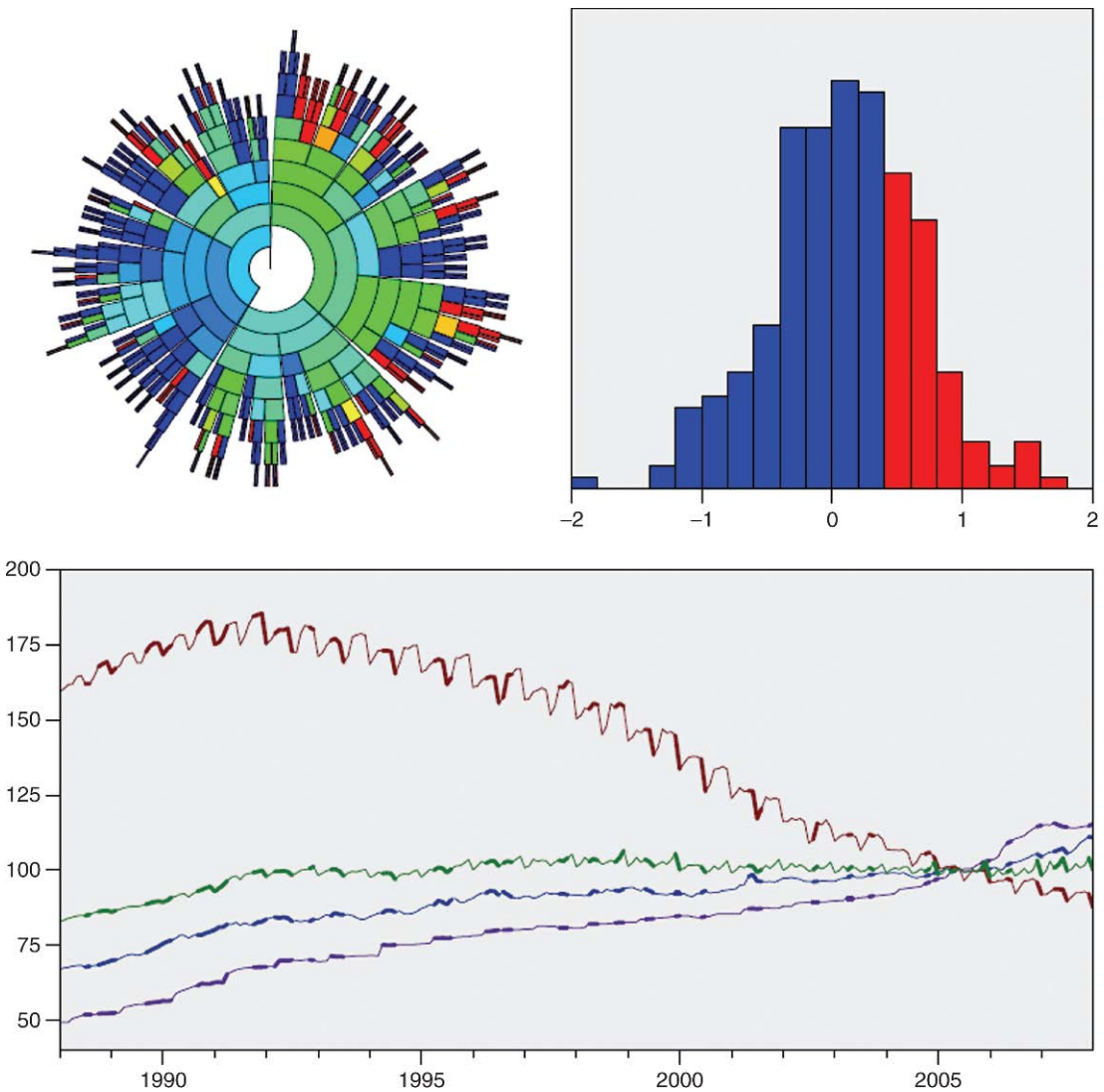
be treated as a specialist tool for hierarchies to which it is applicable, not as a general purpose hierarchical visualization.

Interactive Visualization of Hierarchical Data

The entry on Visualization of Network Data indicated some techniques for interacting with general networks. These can be applied to the special case of hierarchies, and of these the most applicable and important technique is that of brushing and linking.

Figure 5 demonstrates a number of techniques that have been detailed earlier, and adds interaction to the visualization. The base data are a set of consumer price indices (CPI) from the United Kingdom, collected monthly. A hierarchical clustering has been performed on the months, based on all the CPIs in the data set except the overall CPI. At the bottom is multiple time series chart showing four indices (food, clothing, housing, furniture). At the top right is a histogram showing residuals from a fitted time series mode of the overall CPI. Each view is linked to each other view, so that selecting a region of one chart highlights the corresponding months in all other views.

In this figure, the higher residuals have been selected, denoting months where the actual CPI was higher than the model predicted. This is shown in red in the



Visualizing Hierarchical Data. Figure 5. Linked Views of UK CPI data.

histogram, and in the time series view the segments corresponding to those months are shown with a thicker stroke. It appears that these residuals occur mainly on the downward part of regular cycles on the topmost time series (representing the clothing CPI).

In the hierarchical view, a generalization ahs been made to the linking. As the months are aggregated at higher levels of the hierarchy, these interior nodes in the hierarchical tree map may be partially selected. In this figure the selection percentage is shown with a rainbow hue map, with blue indicating completely unselected, red completely selected, and green half-selected. Intermediate hues indicate intermediate selection percentages. At the outer levels some fully selected clusters can be seen, but more interestingly, looking at the inner bands, there are strong differences in color, indicating some relationship between the distributions of the residuals for the overall CPI, and the distribution of clusters of the other CPIs. This indicates some form of inter-dependence between them should be investigated to improve the model.

Key Applications

Hierarchies are a very common data structure. Application areas include geographical data, which are invariably organized in hierarchies, data on organizations, such as businesses, military organizations, and political entities. Financial data is also often suitable. As well as *a priori* hierarchies, it is very common to generate hierarchies so as to aggregate data and allow higher level information to be viewed. Database systems like OLAP and other roll-up techniques can create hierarchies and allow users to move rapidly between levels to investigate data.

Data Sets

UK figures for CPI were retrieved from http://www.statistics.gov.uk/cpi/, and similar statistics should be generally available from most countries National Statistics office.

The world population data are a subset taken from the CIA's world fact book, available at https://www.cia.gov/library/publications/the-world-factbook/.

Cross-references

► OLAP
► Visualizing Network Data

Recommended Reading

1. Di Battista G., Eades P., Tamassia R., and Tollis I. Graph Drawing: Algorithms for the Visualization of Graphs. Prentice-Hall, Englewood Cliffs, NJ, USA, 1999.
2. Friendly M. The Gallery of Data Visualization. http://www.math.yorku.ca/SCS/Gallery/.

Visualizing Network Data

Graham Wills
SPSS Inc., Chicago, IL, USA

Synonyms

Graph layout; Network topology; Graph drawing; Information visualization on networks

Definition

A network is a set of nodes with edges connecting the nodes. When the graph defined by that set of nodes and edges has other associated data, the result is termed "network data." Visualizing Network Data is the process of presenting a visual form of that structure so as allow insight and understanding.

Historical Background

Although examples of informal drawing of networks and hand-drawn graph layouts can be found stretching back many decades, the discipline of visualizing network data in a systematic fashion dates back only to the availability of computers, with an early paper by Tutte in 1963 titled "How to Draw a Graph" being a prime example. In 1992, the conference *International Symposium on Graph Drawing* commenced meeting and their conference proceedings, available from 1994, provide an excellent overall reference to the state of the art in this subject. From around 1990 onwards, increased attention has been paid to *interactive*, or *dynamic* visualization of Network Data. These techniques have surfaced in a variety of different fields, including Statistical Graphics, Information Visualization, and many applied areas.

Foundations

The basic goal behind a successful static visualization of network data is simply stated: Producing a layout of nodes and edges with a bounded 2-dimensional (or, less commonly, 3-dimensional) region such that the

resulting display portrays the structure of the network as clearly as possible. To achieve this goal the following topics must be addressed:

- Techniques for representing the nodes and edges
- Aesthetic criteria that define what is meant by a clear representation
- Layout Techniques
- Extensions to layout techniques to incorporate data on nodes and edges
- Interactive techniques to augment static layouts

Node and Edge Representation

The simplest form of display for a node is as a small glyph, commonly a circle or a square. It is simple and compact. Further, it is easy to add extra information to, such as color, pattern, label or other aesthetics. These can represent data on the nodes. The first three example layouts Fig. 1 below show nodes as circles. The last layout is different; it uses an extended representation to display a node, allowing the edges to be displayed as vertical lines. This representation is most common when the graph is *tree-like*, or more generally is a *hierarchical* network. When there is special known structure for a graph like this, it is possible to use representations that help elucidate such properties. For general network data, however, simple nodes are the most common and suitable choice.

Edge representation allows more freedom. If the edges are *directed*, so that they have a definite "from" and "to" node, then they are usually portrayed with arrows at the ends, unless the direction is obvious from the layout. Other common representational techniques are:

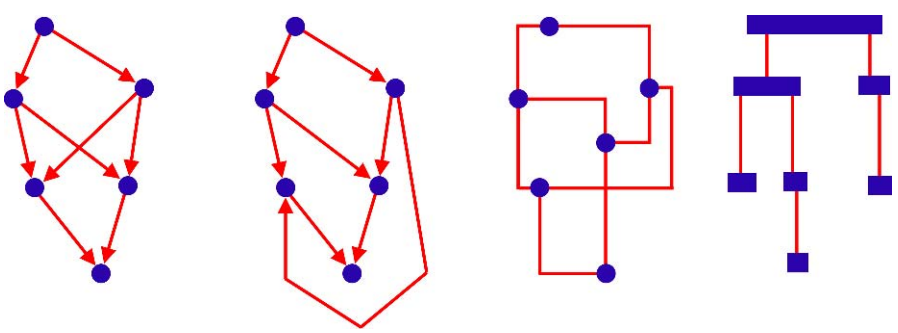
- *Straight Edge*. Figure 1(a) and Figure 1(d) shows edges as straight lines directly linking nodes. This has the advantage of being simple and making connections clear, but tends to result in more edge crossings.
- *Polyline/Paths*. Using paths instead of straight edges allows a reduction in the number of edge crossings in Fig. 1(c), but the resulting display is more complex.
- *Orthogonal Paths*. A style of representation where paths consist of orthogonal lines. This form of representation harkens back to printed circuit board layouts, an early application of graph layout.

Quality Criteria

The question of what makes a good layout has been approached by many authorities. In practice trade-offs must be made, and specific applications stress some criteria more than others. Those criteria that are most often considered important are given below:

- Minimize edge crossings
- Minimize the area needed to display
- Maximize the symmetry of the layout, both globally and locally
- Minimize number of bends in polyline layouts
- Maximize the angle between edges, both at nodes and when they cross
- Minimize total path lengths

There are also criteria that are suitable for specific graph types. For example, in a graph that is *directed* and *acyclic* (there is no path from nodes following edges that loops back on itself), one strong criterion is that the edges generally head in the same direction (all upward, or all downward, as in Fig. 1(a). Note that Fig. 1(b) does not exhibit this quality, an example of the trade-offs made when considering different layout algorithms.



Visualizing Network Data. Figure 1. Representations: (a) Straight-edge, (b) Polyline, (c) Orthogonal, (d) Hierarchical.

Layout Techniques

Layout techniques for general networks employ a variety of techniques, the most common of which are:

Planar Embeddings. A planar graph is one that has a 2-dimensional representation that has no edge crossings. It is also possible to represent any planar graph using only straight-line edges. Although testing and subsequent layout can be done in linear time, the algorithms for doing so are complex. When each vertex has at most four edges, orthogonal representations are possible, but minimizing the number of bends is *NP-hard*, and approximate algorithms are employed in practice.

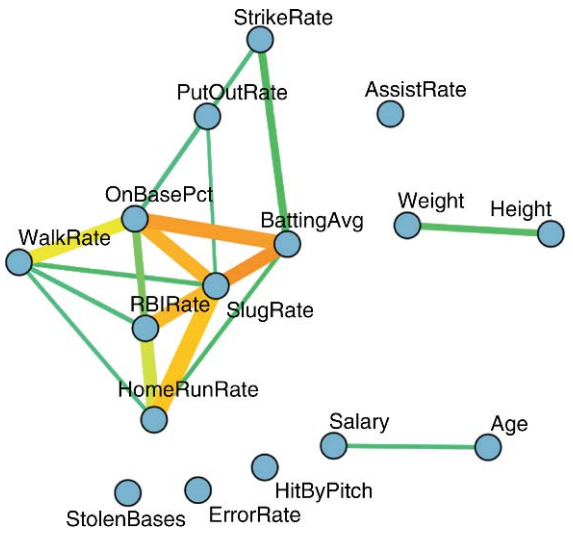
Planarization. If a graph is not planar, it can be made planar by adding "fake" nodes at the crossing points. One such technique would be to find the maximal planar subgraph and laying it out, then adding the additional edges and inserting the additional nodes at crossing points. The resulting graph is laid out using straight edges and then the fake nodes removed, leaving polylines connecting some remaining nodes.

Directed Embeddings. A similar technique is to extract a directed graph from the overall graph (by orienting the edges if necessary) and use a hierarchical drawing technique to place the nodes. A simple example would be finding a minimum spanning tree within the graph and laying it out directly.

Force-directed. Using a mixture of aesthetic criteria, the "energy" of a layout can be defined where low energy corresponds to a good layout. The resulting energy function can be minimized using techniques including randomization, simulated annealing, steepest descent, and simulation. Force-directed techniques are very general, and often can be implemented using iterative algorithms, which makes them amenable to a choice of stopping criteria that can deal with large networks more easily.

Layouts for Networks with Data

If, in addition to the network connections, one also has data on either or both of the nodes or the edges, standard *Information Visualization* techniques can be used to augment the network visualizations. The simplest augmentation is to map a variable for nodes or edges onto an aesthetic, such as color, size, shape, pattern, transparency or dashing. The basic act of labeling nodes is already an example of this use of aesthetics to convey information. Figure 2 shows an application to understanding correlation patterns between variables in a data set consisting of information



Visualizing Network Data. Figure 2. Correlations between Variables; baseball player data, 2004.

on Major League Baseball players in 2004. Only correlations passing a statistical test of adequacy have been retained, resulting in a non-connected graph. The edges contain data on two measures of association between then variables connected by the edge, which have been mapped to aesthetics as follows:

- *Color*: The color represents the statistical significance of the association, with green being weak and red being strong.
- *Size*: The width of the edges indicates how strong the association is in the sense of how much of the variation with one variable can be explained by the other.

The overall layout has a straight-edge representation, and has been arrived at via a force-directed algorithm.

When the data are similar to the above, where one has a measure (or, in this case, two measures) of an edge's strength, the algorithms used should be modified so as to take the strength into account. Although this is possible for all algorithms, it is most simple to implement using force-directed techniques. The goal is to ensure that nodes that are highly associated with each other are close together, and that leads to the criterion that path length should be inversely proportional to edge weight. Reviewing the quality criteria above, some of the criteria can be modified for weighted networks as follows:

- Minimize the weighted summed deviation between path lengths and inverse edge weights.

- Penalize edge crossings with crossings involving strong edges penalized more than weak edges.
- Maximize the angle between edges, both at nodes and when they cross, penalizing angles involving strong edges more than weak edges.

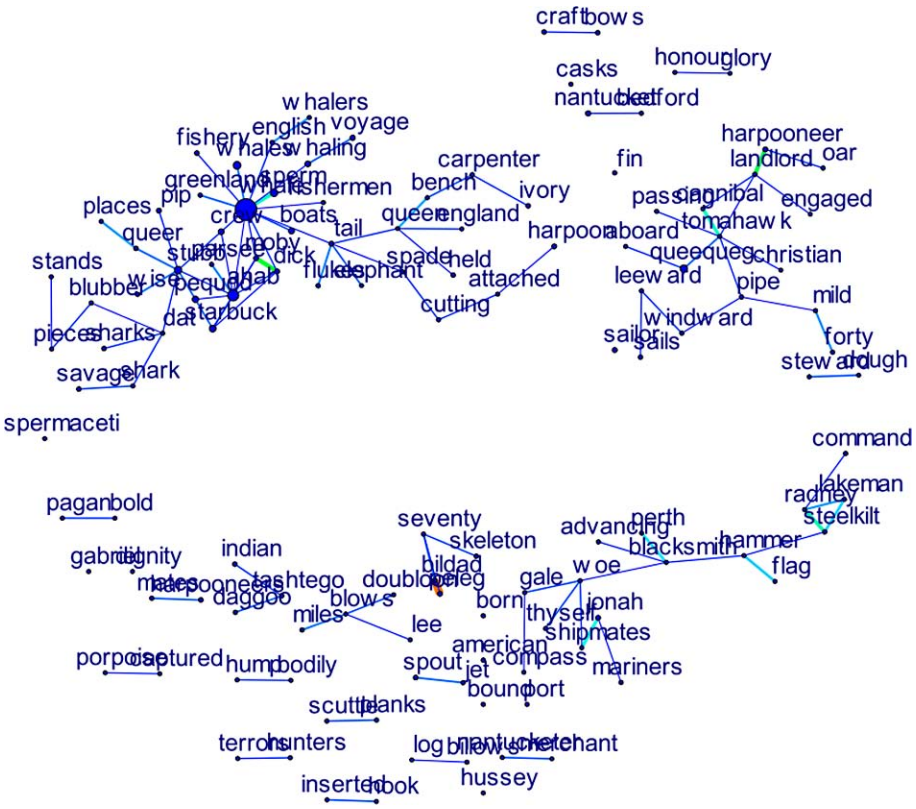
The layout of Fig. 2 was achieved by a force-directed algorithm under these constraints. The measures of association were derived by *Scagnostic* algorithms of Wilkinson and Wills.

Interactive Augmentations

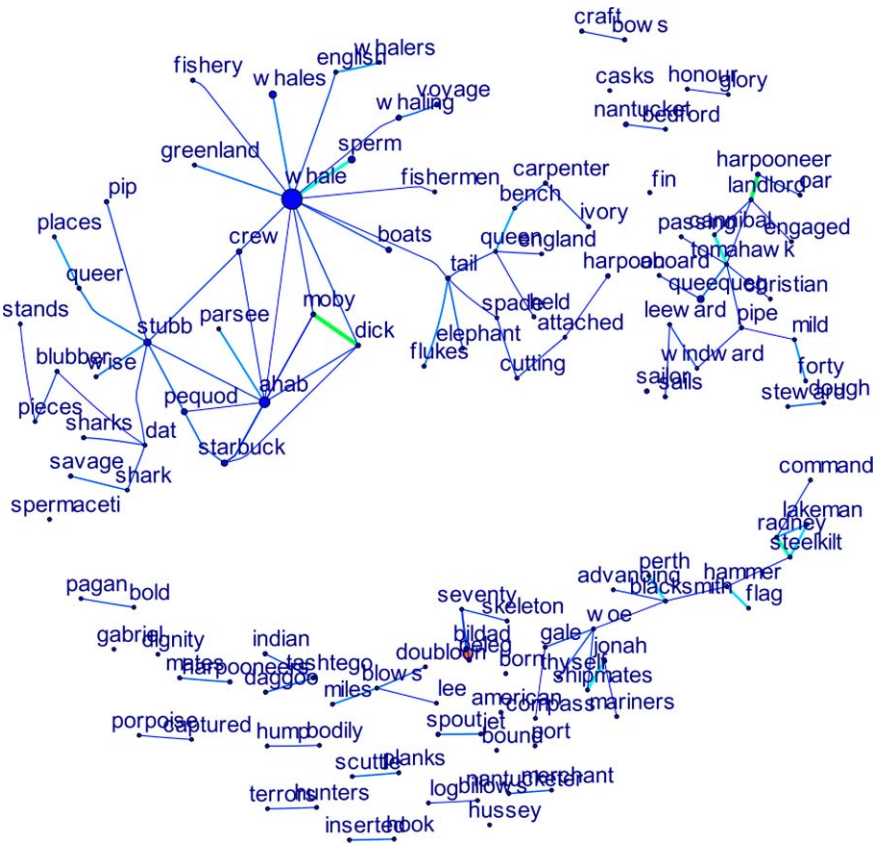
For small and medium-sized static networks, static layouts are adequate, but for larger data sets and for time-varying data, different techniques are required. These can roughly be divided into the following categories:

Distortion techniques. Suitable for large but not huge networks, distortion techniques allow users to place a focus point on a region of interest of the display, and the display redraws so as to magnify the area of interest and de-magnify the rest of the display. Since usable magnification levels can go no higher than a factor of about 5, this technique can improve the number of nodes that can be *visualized* by a maximum of about 25. Specific versions of these techniques include *fisheye* and *hyperbolic* transformations. Figure 3 and Figure 4 below illustrate a network showing associations between words in Melville's *Moby Dick*. The size of the nodes indicates word frequencies, and the links color indicates strength of association. The cluster at the top left is cluttered in Fig. 3, so an interactive fisheye is applied and the focus point is dragged to that cluster, allowing one to see that cluster in more detail, as shown in Fig. 4.

Brushing/Linking. Brushing and Linking techniques are generally applicable techniques for using one visualization in conjunction with another. In the basic implementation a binary pseudo-variable is added to the data set indicating the user's degree of interest. This variable is mapped to an aesthetic in each linked chart, and the user is allowed to drag a brush, or click on



Visualizing Network Data. Figure 3. Force-directed layout of important words in *Moby Dick*. Node sizes indicate word frequencies. Edges link words are commonly found close to each other.



Visualizing Network Data. Figure 4. Force-directed layout of important words in *Moby Dick*. A fisheye transformation has been placed over the dense cluster around the word "whale," magnifying that cluster.

areas of interest in one chart so as to set corresponding values of the "degree of interest" variable and highlight corresponding parts of all linked charts.

This technique is of particular value in the visualization of network data, as it allows any graph layout to be augmented with additional visualizations of data on nodes and links. A simple system for reducing visual complexity is to link histograms, bar charts or other summarized charts to a graph layout display, with the visibility of nodes and links based on the degree of interest. This allows users to select, for example, high values of one variable and then refine the selection by clicking on a bar of a categorical chart. The network display will then show only those items corresponding to the visually selected subset of rows.

A trivial application of this is simply to provide sliders for each variable associated with nodes and links, and by dragging the sliders define a subset of the data which is then displayed as a network display.

Animation. When data on networks varies over time, animating the results can provide insight into the nature of the variation. Visualization is particularly suitable for such data as modeling dynamically evolving networks is a hard problem, with no general models currently available. Hence visualization becomes an important early step in understanding the problem. Technically, animation is similar to brushing, and can be simulated in a brushing environment by making selections on a view of the time dimension. The main differences are in internal techniques to optimize for animation, and in the user interfaces provided for each.

Key Applications

Network Data Visualization is widely applicable. Communication networks such as telephony, internet and cellular are key areas, with particular emphasis on network security issues such as intrusion detection and fraud monitoring.

Future Directions

The field of network data visualization has been evolving steadily since its inception. The classic problem

remains open; providing high-quality layouts for networks. It is known that most problems in this area are NP-hard, and so ongoing research will focus on approximate algorithms. Application areas are increasingly providing large networks with sometimes millions of nodes, which require new algorithms for such data. Further, evolutionary networks, in which the topology of the network itself changes over time, have become more important, and algorithms for evolving a layout smoothly from an old state to accommodate a new state are needed.

Data Sets

The baseball data can be found online at the baseball archive: http://www.baseball1.com. This contains a wealth of tables, and the example data used above was extracted from the tables found there. The text of *Moby Dick* can be found at many sites, including Project Gutenberg: http://www.gutenberg.org.

Cross-references

► Visualizing Hierarchical Networks

Recommended Reading

1. Battista G., Eades P., Tamassia R., and Tollis I. Graph Drawing: Algorithms for the Visualization of Graphs. Prentice-Hall, 1999.
2. Herman I., Melancon G., and Marshal M.S. Graph visualization and navigation in information visualization: a survey. IEEE Trans. Vis. Comput. Graph., 6(1), 2000.
3. International Symposium on Graph Drawing, http://graphdrawing.org/

Visualizing Quantitative Data

Anatol Sargin, Ali Ünlü
University of Augsburg, Augsburg, Germany

Synonyms

Visualizing quantitative data; Graphics for continuous data; Visual displays of numerical data

Definition

Quantitative data are data that can be measured on a numerical scale. Examples of such data are length, height, volume, speed, temperature or cost.

A quantitative variable can be transformed into a categorical variable by grouping, for example weight can be divided into underweight, normal weight and overweight. The inverse transformation may not be possible.

Quantitative data can be categorical or continuous. This entry only concentrates on continuous data. Visualization in general means the graphical or visual display of data or relations.

Key Points

Univariate graphics are the most basic ones and display exactly one variable. There are three plot types that are commonly used: dotplots, boxplots and histograms.

Dotplots draw every data point along one axis. This graphic shows clusters and gaps in the data. Overplotting can be a serious problem, because many points can be crowded around one position. Jittering of the points is then a possible solution. This means, that the points are diffused along a second axis.

Boxplots are also aligned on one axis, but show summary statistics (median, lower and upper hinges) and emphasize outliers. A boxplot gives a broad overview of the structure of the data and sets of boxplots are good for comparing several variables.

Histograms are displayed using two axes. The values of the variable, grouped in bins, are on the abscissa and the frequencies of the values on the ordinate. Histograms can be very informative but strongly depend on the choice of the anchorpoint (start of the first bin) and binwidth.

Bivariate data are typically drawn as scatterplots, which show the structure and dependencies of two variables. The two variables are plotted against each other in an orthogonal coordinate system. The points can be drawn in different colors or symbols to show special groups of the data. Scatterplots are good for detecting one- and two-dimensional outliers and can, together with interactive methods, also be used for large datasets.

Scatterplot matrices, also known as splom, are a generalization for n variables. The (i,j)-th entry is the scatterplot of variable i against variable j. Variable labels are often given in the matrix diagonal. Scatterplot matrices are only efficient for a small number of variables. For more variables parallel coordinates are a better choice. This space-saving graphic shows all variables in one display and hence can visualize the whole dataset.

Cross-references

- ▶ Data Visualization
- ▶ Multivariate Visualization Methods
- ▶ Parallel Coordinates
- ▶ Visualizing Categorical Data

Recommended Reading

1. Cleveland W. Visualizing Data. Hobert, Summit, NJ, USA, 1993.
2. Unwin A., Theus M., and Hofmann H. Graphics of Large Datasets. Springer, Berlin Heidelberg New York, 2006.

Visualizing Spatial Data

- ▶ Visual Interfaces for Geographic Data

Visualizing Trees

- ▶ Visualizing Hierarchical Data

Volume

Kaladhar Voruganti
Network Appliance, Sunnyvale, CA, USA

Synonyms

Logical volume; Physical volume

Definition

The storage provided by a storage device is offered in the form of volumes. A volume corresponds to a logically contiguous piece of storage. The volumes offered by storage devices are typically known as physical volumes.

Key Points

Storage controllers usually create volumes by stripping them across multiple disks that belong to a RAID group. RAID 1 (mirroring), RAID 5 (use of parity, but no dedicated parity disk), and RAID 6 (can tolerate two failures) are the most common RAID types. These volumes get mapped into LUNs by host OS, or aggregated into logical volumes by a logical volume manager, or aggregated into virtual volumes by virtualization software. A file system and its associated files ultimately reside in a physical volume on a storage device. Database table's data also ultimately resides in a physical volume on a storage device. Storage vendors typically use Volume as the unit of management. That is, they allow volume level data copy, migration, snapshots, access control, compression, and encryption.

Cross-references

- ▶ Logical Unit Number (LUN)
- ▶ Logical Volume Manager
- ▶ LUN
- ▶ LUN Mapping
- ▶ RAID

Volume Set Manager

- ▶ Logical Volume Manager (LVM)

Voronoi Decomposition

- ▶ Voronoi Diagram

Voronoi Diagrams

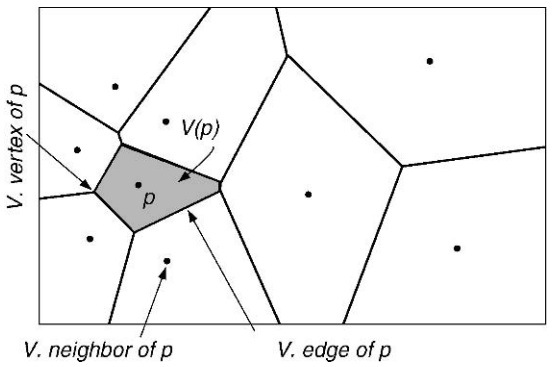
Cyrus Shahabi[1], Mehdi Sharifzadeh[2]
[1]University of Southern California, Los Angeles, CA, USA
[2]Google, Santa Monica, CA, USA

Synonyms

Voronoi tessellation; Voronoi decomposition; Dirichlet tessellation; Thiessen polygons

Definition

The Voronoi diagram of a given set $P = \{p_1,...,p_n\}$ of n points in $\mathbb{R}^d$ partitions the space of $\mathbb{R}^d$ into n regions [2]. Each region includes all points in $\mathbb{R}^d$ with a common closest point in the given set P according to a distance metric $D(.,.)$. That is, the region



Voronoi Diagrams. Figure 1. The ordinary Voronoi diagram.

corresponding to the point $p \in P$ contains all the points $q \in \mathbb{R}^d$ for which the following holds:

$$\forall p' \in P, p' \neq p, \ D(q,p) \leq D(q,p')$$

The equality holds for the points on the borders of p's and p''s regions. Incorporating arbitrary distance metrics $D(.,.)$ results in different variations of Voronoi diagrams. As an example, *Additively Weighted* Voronoi diagrams are defined by using the distance metric $D(p, q) = L_2(p, q) + w(p)$ where $L_2(.,.)$ is the Euclidean distance and $w(p)$ is a numeric weight assigned to p. A thorough discussion on all variations is presented in [7]. Figure 1 shows the *ordinary* Voronoi diagram of nine points in $\mathbb{R}^2$ where the distance metric is Euclidean. The region $V(p)$ containing the point p is denoted as its Voronoi *cell*. With Euclidean distance in $\mathbb{R}^2$, $V(p)$ is a convex polygon. Each edge of this polygon is a segment of the perpendicular bisector of the line segment connecting p to another point of the set P. These edges and their end-points are called *Voronoi edges* and *Voronoi vertices* of the point p, respectively. For each Voronoi edge of the point p, the corresponding point in the set P is called a *Voronoi neighbor* of p. The point p is referred to as the *generator* of Voronoi cell $V(p)$. Finally, the set given by $VD(P) = \{V(p_1),...,V(p_n)\}$ is called the Voronoi diagram of the set P with respect to the distance function $D(.,.)$.

Key Points

Voronoi diagrams exhibit the following properties [9]:

Property 1: The Voronoi diagram of a set P of points, $VD(P)$, is unique.

Property 2: Given the Voronoi diagram of P, the nearest point of P to point $p \in P$ is among the Voronoi neighbors of p. That is, the closest point to p is one of generator points whose Voronoi cells share a Voronoi edge with $V(p)$.

Property 3: The average number of vertices per Voronoi cells of the Voronoi diagram of a set of points in $\mathbb{R}^2$ does not exceed six. That is, the average number of Voronoi neighbors of each point of P is at most six.

Empowered by the above properties, different variations of Voronoi diagrams have been used as index structures for the nearest neighbor search. Hagedoorn introduces a directed acyclic graph based on Voronoi diagrams [3]. He uses the data structure to answer exact nearest-neighbor queries with respect to general distance functions in $O(\log^2 n)$ time using only $O(n)$ space. In [6], Maneewongvatana proposes a hierarchical index structure for point data, termed overlapped split tree (os-tree). Each os-tree node is associated with a convex polygon referred as the *cover*, which includes all the points in space whose nearest neighbor is a data point associated with the node. The same principal used in Voronoi diagrams is utilized in os-trees to partition the space using subset of Voronoi edges into regions, each having different sets of nearest neighbors. In [11], Xu et al. study indexing location data in location-based wireless services. They propose the D-tree, an index structure that can be used to efficiently process NN queries (planar point queries in their terminology). D-tree simply indexes the point data using the subsets of the points' Voronoi edges that partition the entire set into two subsets. This partitioning principle is used in all levels of the tree.

Many studies also focus on utilizing individual Voronoi cells for query processing. Korn and Muthukrishnan [5] describe four examples of the Voronoi cell computation problem, drawn from different spatial/vector space domains in which the influence set of a given point is required. Stanoi et al. in [10] combine the properties of Voronoi cells (influence sets in their terminology) with the efficiency of R-trees to retrieve reverse nearest neighbors of a query point from the database. Zhang et al. [12] determine the so-called *validity region* around a query point as the Voronoi cell of its nearest neighbor. The cell is the region within which the result of the nearest neighbor query remains valid as the location of the query point is changing. To provide an efficient similarity search mechanism in a peer-to-peer data network, Banaei-Kashani and Shahabi propose that each node maintains its Voronoi cell based on its local neighborhood

in the content space [1]. As a more practical example, Kolahdouzan and Shahabi [4] propose a Voronoi-based data structure to improve the performance of exact k nearest neighbor search in spatial network databases. Sharifzadeh and Shahabi [8] utilize Additively Weighted Voronoi diagrams to process a class of nearest neighbor queries.

Cross-references

► Road Networks
► Rtree

Recommended Reading

1. Banaei-Kashani F. and Shahabi C. SWAM: a family of access methods for similarity-search in peer-to-peer data networks. In Proc. Int. Conf. on Information and Knowledge Management, 2004, pp. 304–313.
2. de Berg M., van Kreveld M., Overmars M., and Schwarzkopf O. Computational Geometry: Algorithms and Applications, 2nd ed. Springer, Berlin Heidelberg New York, 2000.
3. Hagedoorn M. Nearest neighbors can be found efficiently if the dimension is small relative to the input size. In Proc. 9th Int. Conf. on Database Theory, 2003, pp. 440–454.
4. Kolahdouzan M. and Shahabi C. Voronoi-based K nearest neighbor search for spatial network databases. In Proc. 30th Int. Conf. on Very Large Data Bases, 2004, pp. 840–851.
5. Korn F. and Muthukrishnan S. Influence sets based on reverse nearest neighbor queries. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2000, pp. 201–212.
6. Maneewongvatana S. Multi-Dimensional Nearest Neighbor Searching with Low-dimensional Data. PhD thesis, Computer Science Department, University of Maryland, College Park, MD, USA, 2001.
7. Okabe A., Boots B., Sugihara K., and Chiu S.N. Spatial Tessellations, Concepts and Applications of Voronoi Diagrams, 2nd edn. Wiley, Chichester, UK, 2000.
8. Sharifzadeh M. and Shahabi C. Processing optimal sequenced route queries using voronoi diagrams. Geoinformatica 12(4), Springer Netherlands, December 2008, pp. 411–433.
9. Sharifzadeh M. Spatial Query Processing Using Voronoi Diagrams. PhD thesis, Computer Science Department, University of Southern California, Los Angeles, CA, 2007.
10. Stanoi I., Riedewald M., Agrawal D., and El Abbadi A. Discovery of influence sets in frequently updated databases. In Proc. 27th Int. Conf. on Very Large Data Bases, 2001, pp. 99–108.
11. Xu J., Zheng B., Lee W.-C., and Lee D.L. The D-tree: an index structure for planar point queries in location-based wireless services. IEEE Trans. Knowl. Data Eng., 16(12):1526–1542, 2004.
12. Zhang J., Zhu M., Papadias D., Tao Y., and Lee D.L. Location-based spatial queries. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 2003, pp. 443–453.

Voronoi Tessellation

► Voronoi Diagram

VP

► Virtual Partitioning

VSM

► Vector-Space Model