Fuzzy Signatures Organized Using S-Tree

Vaclav Snasel, Zdenek Horak, Milos Kudelka, Ajith Abraham
Department of Computer Science
FEI, VSB - Technical University of Ostrava
17. listopadu 15, 708 33, Ostrava - Poruba, Czech Republic
Email: zdenek.horak@vsb.cz, milos.kudelka@vsb.cz, vaclav.snasel@vsb.cz, ajith.abraham@ieee.org

***Abstract*—In this paper we explore the possibility of an efficient organization of fuzzy signatures using the so-called S-Tree. We illustrate the usefullnes of the presented approach on a real-world example of a content-based image retrieval system. Images from the dataset are described using a fuzzy set of features. This description can be translated into a fuzzy signature and these signatures can be stored in a tree structure – similar to the B$^+$tree – that allows efficient retrieval. Several variants are considered and evaluated.**

***Index Terms*—fuzzy sets, data processing**

I. Introduction

There are many areas where objects with very complex and sometimes interdependent features need to be classified; similarities and dissimilarities need to be evaluated. This makes a complex decision model difficult to construct effectively. Images and videos will be captured, manipulated, stored, searched, and reproduced much the way we manipulate text today. It is extremely important to develop technology for the management of large archives of visual information. In particular, image and video data need to be organized efficiently. Fast query mechanisms are required to perform content based retrieval of the stored information. The most difficult part of the problem is to find feature vectors. This vector represent image archives as close as possible and data structures that organize the feature vector space efficiently (thus speeding up the search process). In addition, a feature vector has to be computationally inexpensive to facilitate query processing in real time.

Signature files are one of the methods used for full text retrieval. While some authors suggest that their practical usage is limited to special cases in which they outperform inverted files, the most commonly used method, we believe that exploring the idea of signature files can lead to interesting new techniques for retrieval of not only text documents, but also of other types of documents mentioned earlier.

In the context of our search for new ways of looking at signature methods we tried to apply a fuzzy approach to signature construction and manipulation [1], [7], [11]. Fuzzy signatures can also be used in cases where data is missing. Hierarchical Fuzzy Signatures are generalizations of the Vector Valued Fuzzy Set concept which has been introduced in [5]. The results were the definition of fuzzy signatures which we present in this paper.

This paper is a direct continuation of our previous work published in [9]. We further develop our idea, explore new possibilities and provide experimental results.

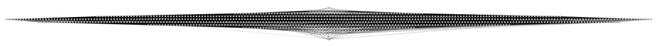
Fig. 1. Concept lattice constructed from color features detected in the dataset

II. Motivation

In our older paper [3] we presented our content-based image retrieval system. This system can detect multiple features in an image and therefore create a description of such image. These features are constructed to correspond human intuition so they can be used for search and similarity queries.

To organize the description of images we have used a formal concept lattice which can identify all natural groups of images and their features which are present in the data. This approach is suitable for investigating the inner structure of the data, but has several practical drawbacks, which limits the use of this method for several purposes. Figure 2 illustrates the structure of the image dataset, with the color features detected. We have to employ reducing methods to obtain a maintable, but still useful, structure analysis. The structure becomes even more complex if we consider fuzzy degrees of detected features.

Practical drawbacks

- formal concept lattice construction is a time and space demanding task (see fig. 1), especially in the fuzzy environment (exponential complexity)
- the complexity can be handled by reduction methods, but most of them depend on some apriori known parameters (see fig. 2)
- navigation (search) in the reduced lattice is not guaranteed to provide the same results as the navigation in the original lattice

To address these problems and obtain a structure for navigation (search) in feature descriptions, we have decided to adopt signature techniques – respectively their fuzzy extension – in combination with a tree-like organization. The aim of our experiment is the practical evaluation of the Fuzzy S-Tree and its variants over the real image dataset.

III. Brief description of signature files

Before we introduce the definition of fuzzy signatures, we would like to mention the basic principles of all signature methods.

Information retrieval methods are used to separate a subset of relevant documents within the finite set of all documents $D = \{D_1, D_2, \ldots, D_k\}$. The extracted documents are said to

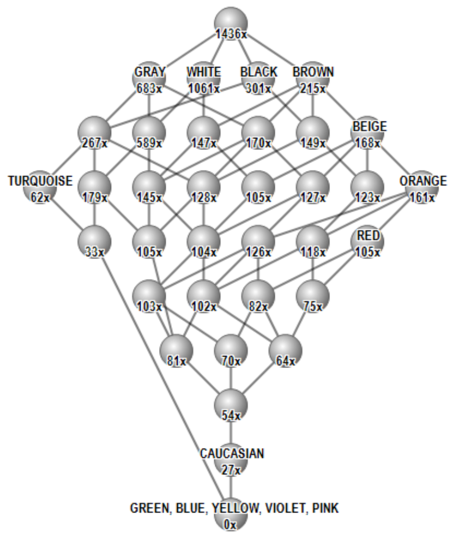

Fig. 2. Concept lattice constructed from color features detected in the dataset, reduced to rank 3

be relevant to a given query Q. A certain amount of work may be required before any query can be evaluated. During this phase, auxiliary data structures are created on disk. These structures enable later evaluation of any given query. In signature methods, signatures are auxiliary structures and are stored in a signature file on disk.

An ordinary signature is a bit string $s_1 s_2 \dots s_n$, of a fixed length n. Signatures of all documents in a set are created and can be used for query evaluation as the signature of the query is compared with them. The lengths of the document signature S_i and the query signature S_Q are the same. Document D_i is relevant only if its signature S_i contains ones in all positions in which ones are encountered in the query signature S_Q.

An important issue is the way the signatures are created. There are two basic possibilities: superimposed coding and concatenation. Since superimposed coding is the only way actually used, we only describe this method. The signature of a document is created in the following way:

In the beginning, the bit string of signature S_i contains zeros in all positions. Let's assume that document D_i contains a finite number of distinct words $\{w_1, w_2, \dots, w_l\}$. Then m positions in the signature bit string are selected for each word. If the bit string contains zeros in these positions, they are replaced with ones. These positions should be distributed evenly throughout the full length of the signature. They are typically chosen as a result of a hash function which takes the word w_j as an argument. The resulting effect is the same as if one signature S_{w_j} was created for each word w_j. This signature would contain up to m ones. The signature of the whole document would be the result of superimposing all signatures $S_{w_1}, S_{w_2}, \dots, S_{w_l}$.

The query signature is created in the same way. The query contains a finite number of words which are used as arguments for the same hash function which was used for the construction of document signatures.

Another important issue is the organization of signatures in the signature file. It affects the efficiency of query evaluation. We will discuss possible organizations in the part devoted to fuzzy signature files.

IV. Fuzzy signatures

As hash functions are for signature extraction, collisions are bound to occur. This results in the selection of the same position p for the distinct words w_i, w_j. The bit in the position p, set to one, indicates the presence of the word w_p in the corresponding document. The word w_p belongs in the set M of all words, for which the hash function has the same result p. That is why the value in position p is the truth value of the statement that the corresponding document belongs to the set of all documents containing any word from the set M. This value is expressed using two numeric values, 0 and 1.

The ideal situation for signature methods occurs when each position in the signature corresponds to exactly one word. In this case, the set M contains just one word.

This interpretation of values in signature string positions is just one step from using fuzzy sets and fuzzy logic for signature extraction. Assuming that the values in signature positions represent degrees of membership in certain sets, we can extend these degrees using numbers from the interval $\langle 0, 1\rangle$.

Definition The fuzzy signature F is a vector $(f_1, f_2, \dots, f_n)$, where $f_i \in \langle 0, 1\rangle \forall i = 1, 2, \dots, n$.

Provided that we have created fuzzy signatures for all documents in the set D and that we have the fuzzy signature F_Q of the query Q, we can use the operation of conjunction to find the relevant documents. The operation is defined in the same way as in fuzzy logic.

Definition The conjunction of fuzzy signatures F_i and F_j is the fuzzy signature

$$F_i \bigwedge F_j = (f_{i_1} \wedge f_{j_1}, f_{i_2} \wedge f_{j_2}, \dots, f_{i_n} \wedge f_{j_n})$$

The operation $\wedge$ is defined for all elements of the fuzzy signature as

$$f_{i_r} \wedge f_{j_r} = \min\{f_{i_r}, f_{j_r}\}$$

In order to find all documents relevant to the given query Q we have to find all documents D_i which satisfy the formula $F_i \bigwedge F_Q = F_Q$. This actually means that for a document to be relevant to the query, all the elements of its signature must be equal to or greater than all the corresponding elements in the query signature.

The operation of disjunction can be defined in a similar way:

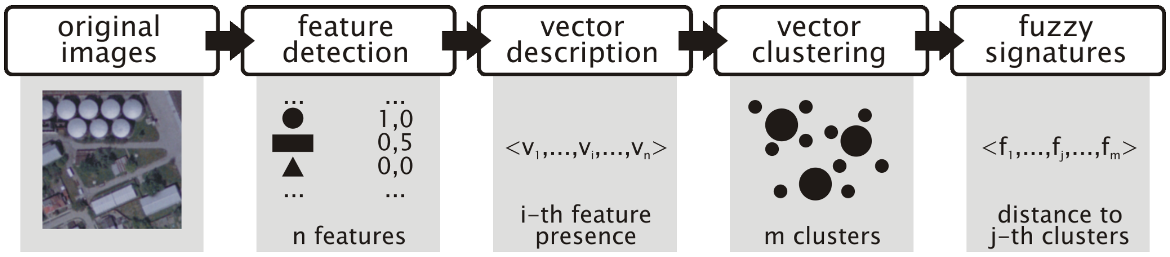

Fig. 3. Illustration of the feature extraction and fuzzy generation process

Definition The disjunction of fuzzy signatures F_i and F_j is the fuzzy signature

$$F_i \bigvee F_j = (f_{i_1} \vee f_{j_1}, f_{i_2} \vee f_{j_2}, \ldots, f_{i_n} \vee f_{j_n})$$

The operation $\vee$ is defined for all elements of the fuzzy signature as

$$f_{i_r} \vee f_{j_r} = \max\{f_{i_r}, f_{j_r}\}$$

These definitions of logical operations with fuzzy signatures reflect the common definitions of logical operations in fuzzy logic. The definitions can be found in [4].

We have described the method of evaluating a query by examining the fuzzy signatures of individual documents and the signature of the query itself, but we have not discussed how fuzzy signatures are actually constructed. We assume that the elements f_i of the string F are degress of membership of the feature M_i. But if the number of features is too large, we may need the signatures to be shorter. We can cluster the documents, according to their features, to some predefined number of clusters (equal to the required length of the signature). The fuzzy signature of one particular document is then given by a vector, whose components are distances from the calculated clusters (see fig. 3).

The application of fuzzy signatures is not limited just to text documents but that they could also be applied to multimedia databases. It is possible to use this method for describing geometrical objects. We could specify certain features of these objects then check for the presence of these features in each object. The elements of the fuzzy signatures would express our degree of confidence that a particular feature is present in a particular drawing.

V. Organization of fuzzy signatures

Another issue to be resolved is the way fuzzy signatures are organized in the signature file. The simplest way would be to store the fuzzy signatures in sequential order. This method is not efficient if the time required for query evaluation is concerned. That is why we have modified the data structure called S-tree, which is traditionally used to store ordinary signatures. In the next section we will sum up the specification of the original S-tree. Following that we will present the modification which enables to use this data structure to store fuzzy signatures. An alternate way of considering weights in signatures is discussed in [8].

A. *Fuzzy S-tree*

We can obtain a data structure for the storing of fuzzy signatures by a modification of the S-tree.
Fuzzy signatures of documents will be stored in leaf pages rather than ordinary signatures. In the non-leaf pages there will be fuzzy signatures too but each fuzzy signature in a non-leaf page will correspond to another page at the lower level. These signatures are created as disjuctions of all fuzzy signatures in the corresponding pages. The operation of disjunction was defined in IV.

S-tree is a balanced tree which uses similar principles as the well known B-tree or its variation, the B$^+$-tree. S-tree is a data structure which allows to search for, insert and remove signatures.

Fuzy signatures are created using a feature extraction function applied to some objects in the database. They are then stored in the leaf pages of the tree. Each signature is accompanied by a link which points to the object described by the signature, or by the object itself. The pages at higher levels contain signatures too, but these signatures are created by superimposing all signatures in the pages of their corresponding successors. This means that each record in a non-leaf page has one whole page assigned at the lower level. In the non-leaf pages there are links to successor pages rather than database objects. Several rules similar to B-tree are defined, for the implementation of the above operations.

1) Each path from the root to any leaf has the same length h.
2) The root page contains the minimum of 2 records and the maximum of K records, except for the cases when it is a leaf page at the same time.
3) Each page except for the root page contains the minimum of k records and the maximum of K records.

k and K are constants.

The major advantage of the S-tree structure is the reduction of the number of signatures which must be searched during the evaluation of a query. In an ideal case, this number would be proportional to the height of the tree. However, this situation is very unlikely as we will explain in the section devoted to splitting of tree pages.

B. Operations enabled by S-tree

1) Searching: Searching is the most common operation. It gives the possibility to identify signatures which are relevant to a given query. The query is transferred into a query signature using the same hash function which was used for the extraction of object signatures. The query signature is used for navigation through the tree from the root to a leaf in the following way: All signatures In the current page are examined to identify all signatures which contain ones in all positions in which there are ones in the query signature.

There can be more than one signature which satisfies this condition. If the current page is a leaf page, objects described by the signatures identified are added to the result of the whole search. If it is not a leaf page, the search continues in the pages of all successors corresponding to the signatures identified.

2) Inserting a signature: Insertion of a record is the next most commonly used operation in most applications. The record contains the signature of the inserted object and a link to the object, or the object itself. The record must be placed in a leaf page and the procedure of insertion must follow the defined rules. The algorithm of insertion starts in the root page again and continues down the tree until it reaches a leaf page. In each step, the algorithm selects one page from the successors of the current page. The signatures in the selected page must be as similar to the inserted signature as possible. To specify the similarity, several measures can be used. Two most commonly used methods are as follows:

1) Hamming metrics δ

$$\delta(S,S^{'}) = \gamma(S \vee S^{'}) - \gamma(S \ \wedge S^{'})$$

where S and $S^{'}$ are signatures of the page and of the inserted object,
L is the length of the signatures
$\gamma(S) = \sum_{i=1}^{L} s_i$ is the weight of the signature S

2) increase of weight ϵ

$$\gamma(S \ \vee S^{'}) = \gamma(S) + \epsilon(S,S^{'})$$

$$\epsilon(S,S^{'}) = \gamma(S \ \vee S^{'}) - \gamma(S)$$

This measure is not commutative and therefore it is not a metric. Nevertheless, it proves to be more suitable for our purposes because we try to minimize the number of ones in the signature of the whole page as described in detail in the following paragraph.

3) Splitting of pages: If the page selected for insertion of a new record is already full (i.e. it contains K records), then it is necessary to split this page into two new pages containing k and $(k+1)$ records. The record of the original page is replaced in the page of the predecessor by two records of the newly created pages. In case the predecessor page is also full, the splitting continues until it eventually reaches the root page. If the root page is split, the height of the tree will increase by one. This is the only way how the tree can grow.

To preserve the logarithmical class of all operations, it is important that the number of pages which must be searched after the the current page is left is as close to one as possible. That is why we try to maintain the weight of all pages at minimum. The higher the number of ones in the signature of a page, the more likely it is that the signature will match the query signature even though its combination of ones resulted from superimposing several signatures with a lower weight.

Signature size	Splitting method	Distance measure	Seek Page ratio
16	PairSplit	WeightIncrease	1,42
32	PairSplit	WeightIncrease	1,29
16	PairSplit	Hamming	2,17
32	PairSplit	Hamming	1,94
16	PivotSplit	WeightIncrease	1,31
32	PivotSplit	WeightIncrease	1,23
16	PivotSplit	Hamming	1,90
32	PivotSplit	Hamming	1,65

TABLE I
SUMMARIZED RESULTS OF EXPERIMENT OVER DATASET WITH 1,600 ITEMS AND 62 FEATURES

It is our aim to find a way of splitting pages which would preserve low weight of the newly created pairs of pages while maintaining high Hamming distance between the pages in the pairs. There is no optimum algorithm at the moment and that is why heuristic approach is used. The simplest choice is to split the page into half w.r.t. some signature order (we call it PivotSplit in the rest of the text). The following way of splitting the pages is suggested by Depisch in [2].
The signature with the highest weight is identified among the signatures within the original page, including the newly inserted signature. The record of this signature is marked as seed α and it is stored in the first of the two new pages. The record whose signature has the largest distance from α is marked as seed β and it is stored in the second page. The remaining records are divided between the new pages depending on whether their distance to seed α is larger than the distance to seed β (we call it PairSplit in the rest of the text).

4) Deletion of a record: The operation of deletion is relatively rare and can be implemented in the following way. First we find the leaf page which contains the record to be removed. The record is deleted from the page. If the number of records in the page has not dropped below the constant k, the operation is completed. Otherwise it is necessary to reconstruct the whole tree to make sure that it satisfies the defined rules. This can be achieved by removing the whole affected leaf page and inserting its records back using the usual procedure for inserting records into the tree. The only exception is that the reinserted records must stay at the same level where they had been before they were removed from the tree.

VI. EXPERIMENT

To evaluate the presented approach, we have used an image dataset containing 1,600 images. Using our content-based image retrieval system, we have detected 62 fuzzy features in these images. For each image, we have therefore obtained a vector $v_i = (v_{i,1}, \ldots, v_{i,j}, \ldots, v_{i,62})$, where $v_{i,j} \in \langle 0, 1 \rangle$ denotes the presence of j-th feature in the i-th image. These vectors have been clustered (with the number of constructed clusters equal to the length of the signature) and the fuzzy signature of the corresponding vector has been created as a coordinate in the vector space induced by the clusters (the whole process is illustrated in fig. 3). These signatures have then been inserted into the Fuzzy S-Tree (with different parameters and measures). Then we performed several search queries and recorded the process of searching in the constructed tree.

Results of this experiment are summarized in table I. In the classical B^+ tree, there is always, at maximum, one way to navigate when searching for an element. As mentioned before, this is not generally true in the S-Tree, nor in the Fuzzy S-Tree. The efficiency of this type of tree is given by the number of alternate ways which must be searched in each step – the closer the number is to one, the better.

In the experiment we have tested the two aforementioned distance measures (Hamming distance and WeightIncrease) and two page-splitting strategies (PivotSplit and PairSplit). From the presented results it is evident that the WeightIncrease measure provides better results than the Hamming distance. Contrary to this, the improvement expected by using the more sophisticated PairSplit page-splitting method (which was proved to be useful for binary signatures) has not been observed in our experiment. This fact should be further investigated.

VII. Conclusion and Future Works

The presented results indicate the viability and usefulness of our approach. Due the limited scope of this paper, we are unable to present further details, applications and evaluation of this approach.

In future work we would like to focus on adopting and evaluating more page-splitting approaches from binary signatures (e.g. approaches mentioned in [10]).

Acknowledgment

This research is partially supported by the Grant Agency of the Czech Republic under the grants No. GA 102/09/1494.

References

[1] A. Ballagi and L. T. Koczy, *Robot cooperation by fuzzy signature sets rule base*, Applied Machine Intelligence and Informatics (SAMI), 2010 IEEE 8th International Symposium on Applied Machine Intelligence and Informatics, pp.37-42, 28-30, 2010.

[2] U. Deppisch, *S-tree: A Dynamic Balanced Signature Index for Office Retrieval*, Proc. of ACM Research and Development in Information Retrieval, pp. 77–87, 1996.

[3] Z. Horak, M. Kudelka and V. Snasel, *FCA as a Tool for Inaccuracy Detection in Content-Based Image Analysis*, IEEE International Conference on Granular Computing, pp. 223–228, 2010.

[4] G. J. Klir, S. U. Clair and B. Yuan, *Fuzzy set theory: foundations and applications*, Prentice-Hall, 1997.

[5] L. T. Koczy, *Vector Valued Fuzzy Set*, Busefal, pp. 1–57, 1980.

[6] W. Y. Ma and B. S. Manjunath, *Pictorial queries: Combining feature extraction with database search*, Technical Report 18, University of California at Santa Barbara, Dept. of Electrical Engineering, 1994.

[7] B. S. U. Mendis, T. D. Gedeon, J. Botzheim, L. T. Koczy, *Generalised Weighted Relevance Aggregation Operators for Hierarchical Fuzzy Signatures*, Computational Intelligence for Modelling, Control and Automation, 2006 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, pp. 198, 2006.

[8] P. Moravec, J. Pokorny and V. Snasel, *Vector query with signature filtering*, Proceedings of 6th BIS conference, 2003.

[9] V. Snasel, *Fuzzy Signatures for Multimedia Databases*, Advances in Information Systems, pp. 257–264, Springer, 2000.

[10] E. Tousidou, A. Nanopoulos and Y. Manolopoulos, *Improved methods for signature-tree construction*, The Computer Journal, vol. 43, 2000.

[11] T. Vamos, L. T. Koczy and G. Biro, *Fuzzy signatures in data mining*, IFSA World Congress and 20th NAFIPS International Conference, vol. 5, pp. 2842–2846, 2001.