A WAVELET DATA MODEL FOR IMAGE DATABASES

Simone Santini

Praja, Inc.
San Diego, CA
ssantini@praja.com

Amarnath Gupta

San Diego Supercomputer Center
San Diego, CA
gupta@sdsc.edu

ABSTRACT

This paper defines wavelet transforms as a datatype suitable for inclusion in databases and an algebra for the manipulation of this data type.

1. INTRODUCTION

Of the several trends in the development of modern database systems, one of the most interesting is towards the inclusion of data types not contemplated by traditional models and which can, with a certain amount of impropriety, be called *multimedia data*. From a database point of view, a multimedia object is usually characterized by a set of visual properties represented with structurally complex features, and not by any property related to "multiple media".

Images, in particular, can be described using a number of different features. Image databases researchers have used color histograms [7], structure graphs [5], transforms [3] or other, more ad hoc, features. These structural features have remained, for the most part, "black boxes" inside which no access operations are permitted. In the case of images, for example, a feature is treated as a collection of opaque elementary components, arranged in some ad hoc serialized data structure, which is retrieved as an atomic unit such that some in-memory algorithm operates, to compare it with another, similar feature. All the problematic of considering an image feature as a *data type*, with the consequent requirements that we will consider shortly, have been largely absent from the bulk of image database research. This has generated a situation in which databases of images have been either simplified into simple memory or linear indices, implemented with ad hoc solutions, or built on top of databases that were not specifically designed to include them, with a consequent loss of efficiency.

We believe that the study of image features as transparent, explicitly manipulable, data types in a database will be of paramount importance for developing efficient and robust systems for storing and searching multimedia data. In this paper we continue our previous work [2] toward a feature-based image data models, by presenting a data type and an algebra for one of the most widely used features in image analysis: multi-resolution transforms. While we present only a first treatment of an algebra of multi-resolution tranforms, the the algebra and the approach are general enough to express most queries of practical interest, and they can be well integrated into a more complete model of feature databases.

2. THE WAVELET DATA TYPE

Our wavelet data model is based upon the concept of complex object, extensively studied in the database literature [9]. If T is an uninterpreted data type (which can be either a basic type like integer or boolean, or an abstract data type created by a type constructor function), the common complex data types built on T are *sets* $\{T\}$, *lists*, $[T]$ and *tuples* $\times t_i$ where t_i is of some type T_i. Arrays are usually considered as a special case of lists. As Libkin and Wong [4] observed, though, in applications that make heavy use of arrays, it is convenient to define them as an independent complex data type. In their definition, a k-dimensional array is a function from a rectangular subset of $\mathbb{N}^k$ (the index set) to a data type T. Such an array is indicated as $[\![T]\!]_k$. If $U^k \subset \mathbb{N}^k$ is the data type of all k dimensional index rectangles aligned with the origin of $\mathbb{N}^k$, then $[\![T]\!]_k$ is equivalent to the function type $U^k \rightarrow T$. In addition to the elementwise application of all the operations defined for the data type T, Libkin and Wong define four functions in their array algebra:

1. $[\![e | i_1 < e_1, \ldots, i_n < e_n]\!]$ builds a k dimensional array with index limits $e_1, \ldots, e_k$ such that the element of indices $i_1, \ldots, i_k$ has the value $\lambda e.i_1 \ldots i_k$;
2. the indexing function $e[i]$, with $i \in \mathbb{N}^k$, which evaluates the array for a particular value of the subscript;
3. the function $dim(e) \in \mathbb{N}^k$, which returns the dimension of an array; and
4. the function $index_k(e) : \{\mathbb{N}^k \times T\} \rightarrow [\![T]\!]_k$, which converts an indexed set into an array.

A Wavelet transform can be seen as a list of two-dimensional arrays of varying dimensions, with the following additional structural constraints:

1. All the arrays in the transform have dimension 2^k for some k and, for all $k < k_0$, there is at least one array with dimension 2^k;
2. the arrays are *index-constrained*.

Two arrays A and B are index constrained if there is a function f that maps subsets of indices of B to indices of A. In other words, let $I_A \subseteq \mathbb{N}^2$ be the set of indices of A, and $I_B \subseteq \mathbb{N}^2$ the set of indices of B, then the function f is defined as $f : 2^{I_B} \rightarrow I_A$. The idea of this function is that it encodes the structural relationship whereby a pixel in a coarser scale of a wavelet transform represents the same spatial location as a set of pixels at a finer scale. The function $map(R, A_i, A_j)$, given a sub-array of A_i returns the corresponding sub-array of A_j. Note that the index-constraint relation is transitive and (trivially) reflexive.

If P is a set of arrays such that any pair of members is index-constrained by some index mapping function, then P is an *index-constrained* set. A wavelet transform is an index-constrained set W of arrays such that:

1. There is a single array A such that every array in W index-constrains A, and
2. there are no loops in the index-constraint relation (that is, it is never the case that A constrains B, B constrains C, and C constrains A).

The operations defined on the wavelet data type are the same defined on its constituent arrays, the function map, and its inverse $imap$. All the operations introduced in the following for the wavelet data type can be defined in terms of these basic operations. In the following, the wavelet data type will be indicated as $\mathbb{W}$, regions (whose definition is standard) as ($\mathbb{R}$), and shapes as ($\mathbb{S}$).

The multi-band operator $\mathcal{M}()$ is defined as follows: given a region r and a wavelet w, the operator

$$m : \mathbb{R}^m = \mathcal{M}(w, r) \tag{1}$$

creates a multi-band region with the same structure as the wavelet w. The multi-band region thus created has components in all the bands of the wavelet transform. It is sometimes necessary to restrict a region to only certain bands of the wavelets. This can be done using the *slice* operator $\mathcal{S}$, which extracts the region component on a single band of the transform (regions composed of multiple bands can be defined as a partially ordered composition of single-band regions). The form of the operator is

$$r_1 : \mathbb{R}^m = \mathcal{S}(r : \mathbb{R}^m, i_1, i_2) \tag{2}$$

where i_1 and i_2 are the indices of the band that will be extracted from the region.

3. GENERAL ORGANIZATION OF A FEATURE DATABASE

Features are defined as a data type in order to be inserted in a database. In this section we will give some general concepts of a possible organization of a relational database that include features. An important difference between feature databases and other relational databases is that the latter do pure *matching* queries while, in the case of images and other multimedia data described by features, the imprecision with which the features describe the data, and the inherent semantic polysemy of the data themselves make it necessary to resort to *similarity* queries. Similarity queries require several extensions of traditional database concepts, including the possibility to handle scoring functions, and a scoring-function dependent join that will not be considered at all in these brief consideration. The interested reader can find more details in [6].

In a database, images are organized in *tables* containing an identifier of the image and a number of attributes, some of which can be features:

id	author	width	height	w
1	Simone Santini	640	480	w_1
2	Amarnath Gupta	640	480	w_2

The non-feature part of this table is managed using typical database operations, so we will ignore it in the following, and consider tables containing only the image identifier and a feature, which we will indicate as $T(id : \texttt{int}, val : \mathbb{W})$. Each row has an implicit read-only attribute, called *score*, and indicated as ς. As a notational matter, the score of the i^{th} row of table T will be indicated as $T[i].\varsigma$, and a similar notation applies to the other attributes.

In traditional databases, queries are implemented using the *select* operator σ: the operation $\sigma_T(Q)$ returns a table composed of all the rows that match the query Q. In the case of a similarity database there are two select operators, which we indicate with σ^ρ and $\sigma^{|k|}$. The operator $\sigma^\rho_T(Q)$ returns all the rows of the table T that receive from the query Q a score of at least ρ. The score field of the returned rows is set to the score with respect to the query Q.

The operator $\sigma^{|k|}_T(Q)$ returns a table composed of the k rows that better match the query Q. The score field of the returned rows is set to the score with respect to the query Q.

The projection operator π is defined exactly as in the case of standard databases. The definition of the join operator $\bowtie$ requires handling the scoring functions of the tables to which it is applied, and will not be considered here.

4. OPERATIONS ON WAVELETS

As mentioned above, a wavelet transform is created from an image using a constructor $\mathfrak{W}()$. The arguments of the constructor specify the image to be transformed, the type of the transform, the number of resolution levels of the transform, and any other necessary parameter depending on the particular transform. For example, the constructor $w = \mathfrak{W}(img, \texttt{GABOR}, 8, 16)$ creates a gabor transform with 8 resolution levels and 16 direction at every level. A special form of the constructor is used in order to create an empty wavelet, in which every coefficient is the `null` value of the type μ: $w = \mathfrak{W}(\mu, p, \texttt{GABOR}, 8, 16)$, where the null image from which the transform is derived has size $2^p \times 2^p$. Two wavelets w_1 and w_2 are said *isomorphic* if they have the same dimension, number of bands and coefficients data type.

The slicing operator extract a band from the wavelet transform given its band indices. The result is a matrix of suitable size with elements of type μ:

$$m : [\mu] = \mathcal{S}(w : \mathbb{W}, i_1, i_2). \tag{3}$$

Note that this operator is identified by the same symbol as the similar operator defined for regions. There is however no ambiguity, since the signature of the operator is different in the two cases. The inverse of the slicing operator replaces a band in a given wavelet transform with the data of a matrix of suitable size:

$$w = \mathcal{S}^{-1}(w : \mathbb{W}, m : \mu, i_1, i_2) \tag{4}$$

Given a function $f : \mu \to \lambda$ and a wavelet w, the apply operator [] returns a wavelet isomorphic to w, with elements of type λ obtained applying the function f to all its coefficients. The operation is written $[f]w$. Given a function $f : \mu \times \mu \to \lambda$ and two isomorphic wavelets w_1, w_2, the apply operator will return a wavelet isomorphic to w_1 and w_2 but with elements of type λ in which coefficients are obtained by applying the function f to the corresponding coefficients of w_1 and w_2; the operation is written $w_1[f]w_2$.

The *map* operator takes an associative and commutative function $f : \mu \times \mu \to \mu$ and makes a "running application" of the function between all coefficients of the wavelet. The operator is written $f\backslash w$

The *mask* operator takes a multi-band region and a wavelet and restricts all operations on the wavelet to the portion included in the region. Masking the wavelet w with the region q is indicated as ${}^q w$.

4.1. Comparison Operators

One of the most important operations in a database is to compare two features to determine the degree to which they match. The comparison operator "=" determines the degree by which two wavelet transforms match. The syntax of the operator is $w_1 = w_2$, which returns the similarity between the two wavelets in the interval $[0, 1]$. This general operator does, in practice, take three different form depending whether a mask is applied to one of the two wavelets. Let $r : \mathbb{R}^m$ be a region and $s : \mathbb{S}^m$ be a shape. Then the semantics of the comparison operator is the following:

$w_1 = w_2$ returns the similarity between the wavelet transforms w_1 and w_2.

${}^r w_1 = w_2$ returns the similarity between the region r of w_1 and the same region of w_2.

${}^r w_1 = {}^s w_2$ returns the similarity between the region s of w_1 and the corresponding region of w_2 that better matches it, independently of its location.

4.2. Examples

The following simple examples should give a general idea of the behavior of the operations introduced so far.

- $w = w_1[+]w_2$: The wavelet w is the sum of the wavelets w_1 and w_2.
- $v = +\backslash w$: Computes the sum of all coefficients of a wavelet.
- $v = +\backslash[\texttt{norm}](w_1[-]w_2)$: Computes the Euclidean distance between two wavelets (`norm` is the function that computes the norm of a coefficient of type μ).
- $v = +\backslash[\texttt{norm}]^q(w_1[-]w_2)$: Computes the Euclidean distance between the portion of the wavlets w_1 and w_2 included into the region q.
- $w = w_1[+]^q w_2$: w is equal to w_1 outside of the region q, and equal to $w_1[+]w_2$ inside the region q.

For examples of queries, assume that the database is composed of a single table $T(id : \texttt{int}, w : \mathbb{W})$. The simplest query possible in this database involving image contents is query by example: given a wavelet transform w, find the k images closest to w. This is written as

$$\sigma_T^{|k|}(w = T.w) \tag{5}$$

The following query finds the images that are similar to a given image in the region r:

$$\sigma_T^{|k|}({}^r w = T.w). \tag{6}$$

Some queries require, in order to be executed, the definition of logic operator on scores. We will not consider such operators here: we will just assume that they are defined

as in Fagin's paper [1] or as in [6]. Some of the following queries will also require the definition of auxiliary functions, which are included into the query language using a syntax derived from that of the ML programming language [8]. The query itself is created as the body of an implicit function `query` which returns an ordered table. For example, the query (6) is translated into the function

$$\texttt{fun query } (w, r) \texttt{ = } \sigma_T^{|k|}({}^r w = T.w)$$

Note that all the free variables in the query will be mae into arguments of the query function.

It is possible to use variables in a query by inserting them in the body of a function with a `let` statement. The following query retrieves the images similar to a given image in general shape (that is, in the content of the lowest resolution bands) and in the high frequency component taken in a given fixed rectangle:

$$\begin{array}{l}\texttt{let}\\ \quad r_1 : \mathbb{R}\texttt{; } w : \mathbb{W}\texttt{; } m_1, m_2, m, m_t : \mathbb{R}^m\\ \quad r_1 = \texttt{rectangle}(100, 100, 50, 20)\\ \quad w_1 = \mathfrak{W}(\texttt{int}, 8, \texttt{GABOR}, 8, 16)\\ \quad m_t = \mathcal{M}(w_1, r_1)\\ \quad m_1 = \mathcal{S}(m_t, 8, *) \cup \mathcal{S}(m_t, 7, *)\\ \quad m_2 = \mathcal{S}(\mathcal{M}(w_1, \texttt{all}), 0)\\ \texttt{in}\\ \quad \sigma_T^{|k|}({}^{m_1} w = T.w \wedge^{m_2} w = T.w)\\ \texttt{end}\end{array}$$

The definition of the region is the most cumbersome part of this function definition but, in practice, libraries of macros are available to make such definition easier.

5. SOME INDEXING NOTES

One of the advantages of defining a feature as a data type and restricting its manipulation to a finite set of operations is the possibility of an efficient implementation of such operations which, in turn, would allow an efficient evaluation of all the queries made using such operators. Implementing a feature algebra efficiently requires two categories of techniques: algebraic manipulation and low-level indexing. The algebraic manipulation subsystem performs some simple query rewriting in order to increase the evaluation efficiency and to traslate the high level algebraic operators into lower level operators, which can be implemented more efficiently. The most important case in point is the grouping of the selection and equality operations into a single indexing operator: the group $\sigma_T^{|k|}(w = T.w)$ is implemented as a single operator, and so are the region-masked operator group $\sigma_T^{|k|}({}^{r:\mathbb{R}^m} w = T.w)$ and the shape-masked operator group $\sigma_T^{|k|}({}^{s:\mathbb{S}^m} w = T.w)$.

This re-definition allows us to implement the operator in an efficient way. This is important especially for databases of large images, such as satellite or medical images, for which loading a whole transform in memory is a time consuming operation. Wavelet are stored using a quadtree structure for every band, so that region operations can be implemented efficiently.

6. REFERENCES

[1] Ronald Fagin. Combining fuzzy information from multiple systems. In *Proceedings of the 15th ACM Symposium on Principles of Database Systems, Montreal*, pages 216–226, 1996.

[2] Amarnath Gupta and Simone Santini. Toward feature algebras in visual databases. In *5th IFIP 2.6 Working Conference on Visual Database Systems, Fukuoka, Japan*, May 2000.

[3] Andrew Laine and Jian Fan. Texture classification by wavelet packet signature. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 15(11):1186–1191, November 1993.

[4] Leonid Libkin, Rona Machlin, and Limsoon Wong. A query language for multidimensional arrays: Design, implementation, and optimization techniques. In *Proceedings of ACM SIGMOD Conference*, pages 228–239, 1996.

[5] Euripides Petrakis and Christos Faloutsos. Similarity searching in medical image databases. *IEEE Transactions on Knowledge and Data Engineering*, 9(3):435–447, 1997.

[6] Simone Santini. *Content Based Retrieval in Image Databases*. Academic Press, 2001. (in press).

[7] Markus Stricker and Markus Orengo. Similarity of color images. In *Proceedings of SPIE, Vol. 2420, Sorage and Retrieval of Image and Video Databases III, San Jose, USA*, pages 381–392, Feb 1995.

[8] Jeffrey Ullman. *Elements of ML Programming*. Prentice Hall, 1994.

[9] Philip L. Wadler. Comprehending monads. In *Proceedings of the 1990 ACM Conference on Lisp and Functional Programming, Nice, France*, 1990.