Principles of Schema Design for Multimedia Databases

Simone Santini, *Member, IEEE,* and Amarnath Gupta

***Abstract*—This paper presents the rudiments of a theory of schema design for databases containing high dimensional features of the type used for describing multimedia data. We introduce a model of multimedia database based on tables containing feature types, and the concept of schema design, which is based on splitting tables depending on the functional relations between different parts of the features.**

We show that certain relations between substructures of a same feature structure can lead to schemas for which efficient algorithms for k-nearest neighbor and range searches can be defined.

I. Introduction

In THE general constellation of problems in which the multimedia research community is engaged at this developmental juncture, efficient storage and retrieval of complex descriptions of data plays a central role. The flurry of activity around content based image retrieval and content based video retrieval in the last five years is a testimony of this centrality.

Most of the research in these areas has focused around the construction of powerful features to describe the contents of an image or a video [3], [9] [12], [17], and on the use of similarity functions to rate the relevance of an image for a query [4][8][18]. Once this is done, the feature vectors are stored in a database, which also provides the query facilities for their retrieval [16], [20]. In most cases, features are stored in the database as “black boxes,” or “blobs” of whose internal structure nothing is known, the only operation defined on them being the measurement of similarity.

As the size of the multimedia heritage increases with utmost rapidity, two issues are becoming preponderant. On one hand, the need to manage a large quantity of data efficiently; on the other hand the need to integrate “sensorial,” or *prosymbolic* data (such as images, video, or audio) into organizational frameworks which include structured and semistructured data of a more symbolic nature. In other words, the databases of the future should (and will) be not just video or image repositories, but comprehensive information systems in which data are aggregated and searched across media, independently of whether information is retrieved from a video, a relational table, or a semistructured web page [10], [14]. This need for cross-mediality generates pressing requirements for the integration of multimedia data into heterogeneous data models, and problems of an exquisitely database oriented nature.

From a database point of view, current multimedia features can be described as opaque data types, whose algebra consists of a single operation: the determination of the similarity between two instances of the data type. There are several negative consequences of this stance.

Most features extracted from images are structured entities, and part of their semantics is captured by their structure. For example, in a wavelet transform, the different resolution levels represent the same image structure at different scales. This scheme is encoded in the structural relation between different levels of the transform but, unless the structure of the wavelet is made explicit for the database to manipulate, one cannot take advantage of it [13].

This paper presents the rudiments of a theory of *schema design* [24] oriented toward storage and retrieval of images into a multimedia database. This orientation entails that our view of features is somewhat different from that of the image analyst. In particular, this paper will not be concerned with the expressive power of features, or the quality with which certain features capture the semantics of the data that they describe. Rather, we will be interested in two problems.

1) Given a set of features that describe certain multimedia data, is it possible to exploit structural relationships between different parts of the features to organize them in a more efficient database schema, in particular one that allows efficient processing of k-nearest neighbor queries [21]?
2) Given a feature design problem, what database issues should the designer consider (next, of course, to all the semantic considerations proper of a feature design problem) so that the resulting feature can be mapped efficiently into a database schema?

The ultimate goal of the methods introduced in this paper is to exploit relations between different parts of a feature structure in order to design a database schema to increase search efficiency. This goal resembles superficially certain dimensionality reduction techniques, like principal component analysis or multidimensional scaling [22], which have been widely used to reduce the dimensionality of feature vectors to a more manageable size. In spite of the superficial similarity, we regard the work presented here as conceptually orthogonal and technically independent of such methods, for a number of reasons.

1) Methods like principal component analysis are used for feature design, rather than schema design. In the schema

Manuscript received received April 17, 2001; revised February 26, 2002. The associate editor coordinating the review of this paper and approving it for publication was Dr. Sankar Basu.

S. Santini is with the National Center for Microscopy and Imaging Research, University of California at San Diego, La Jolla, CA 92093-0608 USA (e-mail: ssantini@praja.com).

A. Gupta is with the San Diego Supercomputer Center, University of California at San Diego, La Jolla, CA 92093 USA.

Publisher Item Identifier S 1520-9210(02)04870-8.

design phase, we assume that the feature designer has already applied the opportune dimensionality reduction methods.

2) Design-oriented reduction methods often entail a certain loss of information. For instance, methods based on principal component analysis prescribe the elimination of a certain numbers of the smallest eigenvalues. It is up to the designer to determine whether this loss is unacceptable, acceptable or, as sometimes happens, beneficial. The methods presented here, on the other hand, work by organizing features in a certain schema, without discarding any information.
3) Schema design relies on properties of the distance functions used by the database. On one hand, this entails that schema design is done without transforming the features, but just by breaking their structure among different tables. So, unlike principal component analysis, schema design can be made based on features defined on a metric space and, unlike multidimensional scaling, it does not transform the features in vectors. On the other hand, schema design can be applied to features of different nature (e.g., color and shape) in a way that depends on the predicted distribution of queries (in which case, of course, less likely queries will be answered less efficiently).

This paper is organized as follows. In Section II we present the database model that we consider in this paper, which is a simple extension of the standard relational model, and which was chosen because it can easily accommodate more powerful models; in particular it can easily accommodate models based on functional languages for query specification and data manipulation simply by encapsulating the relational table into a suitable data type of the functional language [1].

Section III introduces the problem of multimedia database schema design in rather general terms, and highlights the importance of functional dependencies among substructures of a feature for database design problems. Section IV introduces the main classes of dependencies that we consider in this paper, and presents algorithms that take advantage of such dependencies for fast indexing. Section V deals briefly with the case in which the functional dependencies between features is probabilistic in nature. Section VI casts the previous considerations and algorithms into a formal schema design problem, and presents some examples of design. Conclusions are drawn in Section VII.

II. Preliminaries

An image database is formed by one or more relations, or *tables,* of the form $T(h : \mathbb{N}, x_1 : X_1, \ldots, x_n : X_n)$, where h is a unique image handle, $X_1, \ldots, X_n$ are feature types, and x_i is the name of the ith field or *column* of the table. For the sake of simplicity, we will assume that every table contains only features. In practical applications, of course, tables will contain features as well as other information about the images, but the extension to this more general case of the techniques presented in the following is immediate. The *signature* of the table is the sequence of data types ($\mathbb{N}$, $X_1, \ldots, X_n$), and its schema is the sequence of names $(h, x_1, \ldots, x_n)$ with their associated data types. The elements of a schema are called the *explicit fields,* or simply the fields, of the table.

The kth row of the table is indicated as $T[k]$, and it contains the handle some features relative to one of the images stored in the database. $T[k].h$ is the handle of the image, and $T[k].x_i$ is the value of the ith feature descriptor of the image. In addition to the explicit fields, each row of the table has a *score field* ς (of type $\mathbb{R}$) such that $T[k].\varsigma$ is the distance between the image in the kth row and the current query.[1] The value of the field ς is assigned by the scoring operator Σ introduced later on. If h_0 is a handle, the notation $T[h_0]$ will be used to indicate the row k for which $T[k].h = h_0$.

Many image queries are based on distance measures in feature spaces or, equivalently, on similarity functions. In the following, we will always talk in terms of distance functions but it should be understood that the same considerations apply, mutatis mutandis, to similarity function, given the duality existing between distance and similarity [22]. Given a feature type X, let $\mathfrak{D}(X)$ be the set of distance functions defined on X. All distance functions considered in this paper take values in the interval [0, 1] and are *curried,* that is, they are of type d: $X \to X \to [0, 1]$. Given an element $x : X$, and $d \in \mathfrak{D}(X)$, the function $d(x)$: $X \to [0, 1]$ assigns to every element of X its distance from x. Such a function is called a *scoring function,* and the set of all scoring functions for a feature type X is indicated as $\mathfrak{S}(X)$. Each table T has associated a distance, indicated as $T.d$, such that, if the signature of T is ($\mathbb{N}$, $X_1, \ldots, X_n$), then $T.d \in \mathfrak{D}(X_1 \times \cdots \times X_n)$. Each row of the table has associated a scoring function

$$T[k].d = T.d(T[k].x_1, \ldots, T[k].x_n) \in \mathfrak{S}(X_1 \times \cdots \times X_n) \tag{1}$$

that measures scores with respect to the image described by the row.

Moreover, a library of distance combination operators is defined. These are based on scoring combination operators $\diamond$: $[0, 1] \times [0, 1] \to [0, 1]$, defined as in [11], [7]. Each one of these operators induces an operator on distance functions as follows. Let $d_1 \in \mathfrak{D}(X)$ and $d_2 \in \mathfrak{D}(Y)$, then the distance operator $\diamond$: $\mathfrak{D}(X) \times \mathfrak{D}(Y) \to \mathfrak{D}(X \times Y)$ is the operator that makes the following diagram commute:

$$\begin{tikzcd}
\mathfrak{D}(X) \times \mathfrak{D}(Y) \arrow[r, "\diamond"] \arrow[d, "\mathrm{eval}\times\mathrm{eval}"'] & \mathfrak{D}(X, Y) \arrow[d, "\mathrm{eval}"] \\
{[0, 1] \times [0, 1]} \arrow[r, "\diamond"] & {[0, 1].}
\end{tikzcd} \tag{2}$$

That is, for $x : X$ and $y : Y$, it is

$$(d_1 \diamond d_2)(x, y) = d_1(x) \diamond d_2(y). \tag{3}$$

The combination operators will be assumed to be symmetric and Lipschitz, that is

$$x \diamond y = y \diamond x \tag{4}$$

$$|x \diamond y - x \diamond z| \leq L|y - z| \tag{5}$$

[1] The use of the term *score* implied in this definition is a bit anomalous. In normal discourse, a score is positively correlated with significance, so that significant images have high score. In this case, however, score is a distance, which means that significant images have low scores.

for some constant $L > 0$. (The definition establishes that the operator is Lipschitz on the second argument only; the property on the first argument follows from symmetry.)

Operators on Tables: Given a scoring function s, and a table $T(\mathbb{N}, X_1, X_2, \ldots, X_n)$ the *scoring operator* $\Sigma_T(s)$ assigns a score to all the rows of T using the scoring function s. That is, $\Sigma_T(s)$ is a table with the same signature as T and

$$(\Sigma_T(s))[k].\varsigma = s(T[k].x_1, \ldots, T[k].x_n). \tag{6}$$

Given a table $T(\mathbb{N}, X_1, X_2, \ldots, X_n)$, the k lowest distances operator $\sigma_k^{\#}$ returns a table with the k rows of T with the lowest distance from the query. The operators $\sigma_k^{\#}$ and Σ_T are generally used together: the operator $\sigma_k^{\#}(\Sigma_T(s))$ is called the k-nearest neighbors operator for the scoring function s.

The operator $\sigma_\rho^{<}$ returns all the rows of a table T with a distance less than ρ. The operator $\sigma_\rho^{<}(\Sigma_T(s))$ is called the range query operator for the scoring function s.

The operator σ_P is the usual predicate selection operator on a table T. In the databases that we consider here, P has either the form $h = h_0$, where $h_0 \in \mathbb{N}$ is a handle, or $h \in H$, where H is a set of handles. Note that the notation $T[h_0]$ introduced above is a shorthand for $\sigma_{h=h_0}(T)$ which, because of the unicity of the handles, always returns a table with a single row.

Finally, the $\diamond$-join $\underset{\diamond}{\bowtie}$ is a join operator in which two tables T and Q are joined on their handle field to form a new table $W = T \underset{\diamond}{\bowtie} Q$. If $T = T(h : \mathbb{N}, x_1 : X_1, \ldots, x_n : X_n)$ and $Q = Q(h : \mathbb{N}, y_1 : Y_1, \ldots, y_n : Y_n)$, then

$$W = W(h : \mathbb{N}, x_1 : X_1, \ldots, x_n : X_n, y_1 : Y_1, \ldots, y_n : Y_n). \tag{7}$$

The row q such that $W[q].h = h_0$ is obtained by collecting the features of the rows $T[i]$ and $Q[j]$ such that $T[i].h = Q[j].h = h_0$. The table W has a distance function

$$W.d = T.d \diamond Q.d \tag{8}$$

and, if the qth row of W was obtained by joining the ith row of T with the jth row of Q, it has score

$$W[q].\varsigma = T[i].\varsigma \diamond Q[j].\varsigma \tag{9}$$

and

$$W[q].d = T[i].d \diamond Q[j].d. \tag{10}$$

III. Database Schema Design

Let $T(h, x_1, \ldots, x_n)$ a feature table with schema $(\mathbb{N}, X_1, \ldots, X_n)$. Assume, for the sake of simplicity, that each feature type X_i can be represented as a vector, and that its dimensionality is m_i.[2] In a multimedia database, typical operations on this table are k-nearest neighbors and range searches, which involve ordering the rows of the table using the distance $T.d \in \mathfrak{D}(X_1, \ldots, X_n)$ that is, they involve indexing a feature space of dimension $m = \sum_i m_i$.

It is a well known fact that, as the dimensionality of the feature space increases, the performance of all indexing structures declines rapidly and, even for moderate values of m, there is no solution more efficient than the trivial one of visiting all the rows in the table and measuring their distance from the query [25], [5], [19].

As long as one considers features as black boxes about which nothing is known except their distance function, there is no general solution to this problem. A solution can only derive from the integration of feature design into the general activity of database design. There are characteristics of certain features that afford a re-organization of the database in the sense of a greater efficiency, and these characteristics should be identified, pursued, and exploited.

In this paper, we will consider the problem of *schema design* for an image database and how certain characteristics of features can be used to design schemas that can be searched efficiently. The problem can be defined as follows:

Definition 3.1: Consider a set of images S, and assume that for each image the features $F = \{x_1, \ldots, x_n\}$ is available, with $x_i : X_i$. A *database schema* is a set of tables $\mathcal{T} = \{T_1, \ldots, T_q\}$ with $T_i = T_i(h, x_{i,1}, \ldots, x_{i,q_i})$ such that, if $F_i = \{x_{i,1}, \ldots, x_{i,q_i}\}$, then $F = \bigcup_i F_i$.

Using a set $\mathcal{T}$ of tables, rather than a single table T can lead to more efficient searches because in many cases one of the tables can be used as a *prototable* that can be searched to individuate a subset of the database in which the solution is guaranteed to be so that the search in the complete feature space can be limited to this subset rather than to the whole database. (A formal definition of prototable will be given later in the paper.)

A general algorithm for search in a database decomposed in a prototable K and in a set of *dependent tables* $\{T_1, \ldots, T_q\}$ is the following

1) Search the prototable K, and retrieve a set W of p images that is guaranteed to (or has a pre-specified probability to) contain the images that satisfy the query.
2) Join the set W with all the dependent tables, obtaining a table $Q = W \underset{\diamond}{\bowtie} T_1 \underset{\diamond}{\bowtie} \cdots \underset{\diamond}{\bowtie} T_q$ which contains the full feature description of the p images in the table W.
3) Execute the query on the table Q, possibly doing a linear scan of the table if its dimensionality does not allow for efficient indexing.

The only feature search that this algorithm does on the whole database involves the prototable K which may be of a much lower dimensionality than the whole database, thus allowing efficient indexing.

The possibility of dividing a database into a set of smaller tables depends on certain characteristics of the features $x_1, \ldots, x_n$ and it is a goal of the feature designer to select features that allow the design of an efficient schema. The study of the characteristics of an image feature vis-à-vis schema design will form the subject of a discipline of multimedia database design that is still, for the most part, in the making. This paper will try to provide some foundations for such a discipline by analyzing some limited and specific characteristics of features

[2] This hypothesis is not really necessary, and can be replaced by the less stringent hypothesis that X_i be a metric spece of intrinsic dimensionality d_i. The vector space hypothesis, however, allows a simpler exposition.

that can be used for schema design, in particular, we will consider *partial functional dependencies* between subsets of features.

Straight functional dependence provides a trivial way of dividing a table. Consider a table with two features: $T(h : \mathbb{N}, x_1 : X_1, x_2 : X_2)$, with $T.d = d_1 \diamond d_2$, $d_1 \in \mathfrak{D}(X_1)$ and $d_2 \in \mathfrak{D}(X_2)$. Suppose that there is a function f such that $x_2 = f(x_1)$. Then the table can be trivially divided into a prototable $K(h, x_1)$ and a dependent table $T(h, x_2)$. Assigning to K the distance function $K.d = d_1 \diamond (d_2 \circ f)$ one can ignore the table T and do the searches on the prototable K alone.

This is a trivial case in which the search algorithm assumes a particularly simple form, but it is not very realistic. In the remainder of the paper, we will consider cases of *partial* functional dependencies, which are more realistic and can still be used to split tables.

IV. Functional Dependencies

In this section we want to determine under which conditions is it convenient to separate a table $T(h, x_1, x_2)$ with two feature fields into two tables $T_1(h, x_1)$ and $T_2(h, x_2)$, in particular, we are interested in the various forms of dependencies between the scoring functions (or, equivalently, the distance functions) defined on the two features, and how those can be used to design algorithms that allow a faster computation of k-nearest neighbor queries and range queries once the tables are split.

We will assume that the distance function of the table T is obtained as the combination of the distance functions of the two features: $T.d = d_1 \diamond d_2$. We will also assume that a reference image h_0 has been selected, and that all distances are relative to this image. In order to ease the notation, we will indicate with s the scoring functions relative to this image, that is $s = T[h_0].d$ or, when referred to the features x_1 and x_2 individually, $s_1 = T[h_0].d_1$ and $s_2 = T[h_0].d_2$.

A. Search Algorithm

Before considering various forms of dependence between features, we introduce in this section the algorithm that will be used to solve the k-nearest neighbors problem in the case in which a table containing two related features has been split.

The general idea of the algorithm is as follows. Suppose we have two features x_1 and x_2, belonging to feature spaces X_1 and X_2, and that there is a relation between the two that allows us to predict, to a certain degree, the score of x_2 given the score of x_1. If the relation between the two features is $\mathfrak{r}$, we will use the notation $x_1[\mathfrak{r}\rangle x_2$. If we do a k-nearest neighbors search in the space X_1 we will not, in general, end up with the correct solution. Consider, however, the distance between the query and the kth image retrieved by the k-nearest neighbors algorithm using the feature x_1, and $\tilde{\rho}$ be this distance. One can hope that by making in the feature space X_1 a search with a radius suitably larger than $\tilde{\rho}$, one will collect a number of images $k' > k$ which will contain the k solutions of the original problem. The amount by which a given radius $\tilde{\rho}$ must be increased is a function of the relation $\mathfrak{r}$, $\zeta(\mathfrak{r})$: $\mathbb{R} \to \mathbb{R}$. The radius ρ is then written as $\rho = \zeta(\mathfrak{r})(\tilde{\rho})$. In formal terms, the algorithm can be written as follows:

Algorithm 1.

```
1. $Q = \sigma_k^{\#}(\Sigma_{T_1}(T_1[h_0].d))$;
2. $\tilde{\rho} = Q[k].\varsigma$;
3. $\rho = \zeta(\mathfrak{r})(\tilde{\rho})$;
4. $Q = \sigma_{\rho}^{<}(\Sigma_{T_1}(T_1[h_0].d))$;
5. $W = Q \underset{\diamond}{\bowtie} T_2$;
6. $R = \sigma_k^{\#}(\Sigma_W(W[h_0].d))$.
```

Step 1 does a k-nearest neighbors search of the neighbors of image h_0 in table T_1. In steps 2 and 3, the distance between the farthest away of the returned images and the query is found and increased by a suitable factor, which depends on the relation $\mathfrak{r}$ between the two sets of features. In step 4, a range query is done on table T_1 to obtain the images closer to the query than this new distance. The resulting table (Q) contains $k' > k$ entries, and is guaranteed to contain the k images that, in the original table, were closest to the query (see below for a proof of this fact). In order to determine these images, the table Q is joined to T_2 and the k images closest to the query are extracted from this table. The advantage, in this algorithm, is that the search using the complete feature space is only done on the table W, which contains only k' entries, and not the whole database.

The crucial point of the algorithm is step 3, in which the range of the query is "increased a little bit" to guarantee that the range query of step 4 will contain all the solutions to the original query. Whether this is possible, it depends on the relation $\mathfrak{r}$. In particular, $\mathfrak{r}$ must have a *distance bounding* function, defined as follows.

Definition 4.1: Let $\mathfrak{r}$ be a relation such that $x_1[\mathfrak{r}\rangle x_2$, with $x_1 : X_1$ and $x_2 : X_2$, $s_1 \in \mathfrak{S}(X_1)$ and $s_2 \in \mathfrak{S}(X_2)$ be two scoring functions for the features X_1 and X_2, and $s = s_1 \diamond s_2$ be a scoring function for a table containing both features, $\diamond$ being an arbitrary but fixed combination operator. A function $\zeta(\mathfrak{r})$ is a *bounding function* for $\mathfrak{r}$ if, for a given scoring function s, the following is true.

> For every ρ, if there are exactly k rows such that $s_1(x_1) \leq \rho$, then there are at least k rows for which $s(x) = s_1(x_1) \diamond s_2(x_2) \leq \zeta(\mathfrak{r})(\rho)$.

Then the following theorem proves the correctness of the algorithm:

Theorem 4.1: In algorithm 1, assume that the two tables are $T_1(h : \mathbb{N}, x : X)$ and $T_2(h : \mathbb{N}, y : Y)$, with $x[\mathfrak{r}\rangle y$, and that $\zeta(\mathfrak{r})$ is a bounding function for the relation $\mathfrak{r}$. Then the algorithm retrieves the k images closest to the query with respect to the scoring function $sT_1[h_0].d \diamond T_2[h_0].d$.

Proof: The proof is by contradiction. Assume that the algorithm is incorrect. Then there is an image v such that $s(v)$ is one of the k lowest distances, but which is not returned by the algorithm.

If this is true, then the image v was left out of the range query in step 4 since, had it been part of the table Q, it would have been picked up in step 6. To see this, consider that step 5 is a join between the identifiers in Q and a superset of the same identifiers contained in T_2; as such, the join will drop no entries from Q. Step 6, on the other hand, is a k nearest neighbors query based on the scoring function s and since, ex hypothesis, image v is one of the k images with the lowest distance, it will return it.

If v was not returned by the range query of step 4, then it is $T_1[h_0].d(v) > \zeta(\mathfrak{r})(\tilde{\rho})$ that is $s(v) > \zeta(\mathfrak{r})(\tilde{\rho})$. But, since $\zeta(\mathfrak{r})$ is a bounding function for $\mathfrak{r}$ and, by virtue of step 1, there are k images u_i such that $T_1[h_0].d(u_i) \leq \tilde{\rho}$, it follows from the definition that there are at lest k images w_j for which $s(w_j) \leq \zeta(\mathfrak{r})(\tilde{\rho})$, which contradicts the hypothesis that v was one of the k images closest to the query. □

This is the basic algorithm that we will use in the remainder of the paper. In the following section, we will present three relations (the first two of which will turn out to be special cases of the third) with the necessary requirements for applying the algorithm.

B. Scale Difference

Two features x_1 and x_2 are in r-scale difference dependence, written $x_1[|r\rangle x_2$ if there is an $r \geq 1$ such that, for all images u, it is $s_2(x_2(u)) \leq s_1(x_1(u))^r$. This relation can be used to decompose a table into a normal form, as described by the following definition.

Definition 4.2: A set of tables is said to be in *z-scale-normal form* if, whenever $x_1[|r\rangle x_2$ with $r \geq z$, x_2 does not belong to the prototable. In other words, the prototable contains no feature that is r-scale dependent on another feature with $r \geq z$.

The idea behind the definition z-scale-normal form is that if a part of a feature gives a small contribution to the distance between the query and the elements in the database, then there should be the possibility of searching only on the part of the feature structure that gives the greatest contribution. In other words, if $x_1[|r\rangle x_2$ with high r then, presumably, the distance between two images with respect to x_1 is highly indicative of the distance between the two images with respect to the whole feature vector. It should then be possible to use x_1 to filter out from the database images that cannot possibly be returned as part of a given query.

In order to show that the algorithm of the previous section applies, we need to find a bounding function for this relation. This is done by the following lemma:

Lemma 4.1: The function $\zeta([|r\rangle)(\rho) = \rho \diamond \rho^r$ is a bounding function for the relation $[|r\rangle$.

Proof: Let $T_1(h : \mathbb{N}, x : X)$ and $T_2(h : \mathbb{N}, y : Y)$ be the two tables such that $x[|r\rangle y$, and let $s_1 = T_1[h_0].d$ and $s_2 = T_2[h_0].d$. We need to prove that, if there are k images such that $s_1(u) > \rho$, then there are at least k images such that $s(u) \geq \rho \diamond \rho^r$. This is a simple consequence of the dependence between the two features. Since $x[|r\rangle y$, it is $s_2(u) \leq s_1(u)^r$ and, since the combination operators are monotonically increasing in their arguments, it is

$$s(u) = s_1(u) \diamond s_2(u) \leq s_1(u) \diamond s_1(u)^r. \tag{11}$$

Therefore, at least the k images for which $s_1(u) \leq \delta$ have $s(u) \leq \delta \diamond \delta^r$. □

Given the existence of the bounding function, Algorithm 1 can be applied: if two features x and y are such that $x[|r\rangle y$, they can be divided in two different tables $T_1(h : \mathbb{N}, x : X)$ and $T_2(h : \mathbb{N}, y : Y)$ in such a way that a complete search needs to be done only on T_1.

Transitive Closure Properties: In schema design, dependencies are used to split features into several tables so that each table will have only features of a limited dimensionality and, therefore, can be indexed more efficiently. Dependencies among parts of a feature structure can either be proved *a priori*, based on the semantics of the feature extractor, or through statistical measurements. In addition, the structure of the scale dependence itself allows to extend the relation using a form of transitive closure so that, given scale functional dependencies among certain features, other dependencies can be inferred. The transitive closure properties of the scale functional dependence are the following.

Self-dependence: For all features x, $x[|1\rangle x$.

Anti-symmetry: If $x[|r\rangle y$, $r > 1$, then there is no $q > 1$ such that $y[|q\rangle x$.

Transitivity: If $x[|r\rangle y$ and $y[|q\rangle z$, then $x[|rq\rangle z$.

The proof of these properties is immediate from the definition. These properties are used to compute the *q-transitive closure* of the features that depend on a given feature x, defined as follows:

Definition 4.3: Let $X = \{x_1, x_2, \ldots, x_n\}$ be a set of features computed on the same database (or a decomposition of a single feature computed on the database). Given a feature $x_i \in X$, the q-transitive closure of x_i is the set:

$$[x_i]_{|r} = \{x_k \in X : x_i[|r\rangle x_k,\ r \geq q\}. \tag{12}$$

The value q in the definition of q-transitive closure has a direct relation with the cost of Algorithm 1, since the largest is q in a relation $x[|q\rangle y$, the cheaper it is to execute Algorithm 1, as will be shown in the following. Therefore, given a cost objective, an acceptable value of q can be derived. The goal of the designer is then to find the smallest feature x such that $[x]_{|r} = X$. The details of this operation will be covered later, after introducing two additional forms of dependence.

C. Partial Functional Dependence

Partial functional dependence is a weak form of functional dependence which depends on the existence of a scoring function and allows for a bounded indetermination in the dependence itself.

Definition 4.4: Given two features x_1 and x_2, a scoring function s, a constant Δ, and a function f, the feature x_2 is (Δ, f, s)-dependent of x_1, written $x_1[\Delta|\rangle x_2$ if for all images u

$$|s(x_1(u)) - s(x_2(u))| \leq \Delta. \tag{13}$$

Similarly to the scale functional dependence, partial functional dependence gives rise to a family of normal forms for table indexed, in this case, by the parameter Δ.

Definition 4.5: A set of tables is said to be in *Δ-partial functional normal form* if, whenever $x_1[\Delta|\rangle x_2$, it is not the case that x_1 and x_2 belong to the same prototable.

As in the previous case, the utility of the concept of Δ-partial functional normal form comes from the existence of a bounding function, so that Algorithm 1 applies.

Lemma 4.2: The function $\zeta([\Delta|\rangle)(\rho) = \rho + L\Delta$ is a bounding function for the relation $[|r\rangle$, L being the Lipschitz constant that appears in (5).

Proof: Let $T_1(h : \mathbb{N}, x : X)$ and $T_2(h : \mathbb{N}, y : Y)$ the two tables such that $x[|r\rangle y$, and let $s_1 = T_1[h_0].d$ and $s_2 = T_2[h_0].d$. We need to prove that, if there are k images such that $s_1(u) > \rho$, then there are at least k images such that $s(u) \geq \rho + L\Delta$.

Because $x_1[\Delta|\rangle x_2$, and the Lipschitz property of $\diamond$, it is

$$\begin{aligned} s(u) &= s_1(u) \diamond s_2(u) \leq s_1(u) \diamond (s_1(u) + \Delta) \\ &\leq s_1(u) \diamond s_1(u) + L\Delta \end{aligned} \tag{14}$$

and, similarly, $s(u) \geq s_1(u) \diamond s_1(u) - L\Delta$; that is

$$\tilde{s}(u) - L\Delta \leq s(u) \leq \tilde{s}(u) + L\Delta. \tag{15}$$

Therefore, for all the k images for which $\tilde{s}(u) \leq \delta$, it is $s(u) \leq \delta + L\Delta$. □

Transitive Closure Properties: As in the case of scale dependence, transitivity relations between features induced by partial functional dependence can be used in the decomposition process. The following properties hold for partially functionally dependent features.

Self-dependence: For all features x, $x[0|\rangle x$.

Symmetry: If $x[\Delta|\rangle y$, then $y[\Delta|\rangle x$.

Monotonicity: If $x[\Delta|\rangle y$, then for all $\Gamma > \Delta$, $x[\Gamma|\rangle y$.

Transitivity: If $x[\Delta|\rangle y$ and $y[\Gamma|\rangle z$, then $x[\Delta + \Gamma|\rangle z$.

The proof of these properties is immediate from the definition. Transitive closure properties are used to compute the Δ-*transitive closure* of the features that depend on a given feature x, defined as follows.

Definition 4.6: Let $X = \{x_1, x_2, \ldots, x_n\}$ be a set of features computed on the same database (or a decomposition of a single feature computed on the database). Given a feature $x_i \in X$, the Δ-transitive closure of x_i is the set

$$[x_i]_{\Delta|} = \{x_k \in X \colon x_i[\Delta|\rangle x_k\}. \tag{16}$$

The use of the transitive closure of a feature is the same as in scale dependence: fixing a bound on the cost of the indexing algorithm, this will yield a bound on the acceptable Δ of the partial functional dependence. The designer will then have to find the smallest feature x such that $[x]_{\Delta|} = X$.

D. Quasi-Scale Dependence

The previous two sections have introduced two forms of dependence between the scores of pairs of features, and it is obvious to wonder whether there is a relation between the two. In other words, we have seen that, within the two dependencies, certain transitivity rules hold, namely

$$x[|r\rangle y \;\wedge\; y[|q\rangle z \implies [|rq\rangle z \tag{17}$$

and

$$x[\Delta|\rangle y \;\wedge\; y[\Gamma|\rangle z \implies x[\Delta + \Gamma|\rangle z. \tag{18}$$

Suppose now that $x[\Delta|\rangle y$ and $y[|r\rangle z$; what can be said of the relation between x and z? Since $z \leq y^r$ and $y \leq x + \Delta$, one can conclude that $z \leq (x + \Delta)^r$. As a matter of fact, one can impose a somewhat stronger condition: since $z \leq 1$ ex hypothesis, one can define the bounded sum operator $\dot{+}$ as

$$a \dot{+} b = \min(a + b, 1) \tag{19}$$

which allows to write $z \leq (x \dot{+} \Delta)^r$. This property motivates a further definition:

Definition 4.7: Given two features x_1 and x_2, a scoring function s and constants Δ and r, the feature x_2 is quasiscale dependent of x_1, written $x_1[\Delta|r\rangle x_2$ if for all images u

$$s(x_2(u)) \leq (s(x_1(u)) \dot{+} \Delta)^r. \tag{20}$$

The following lemma, whose proof is very similar to that of the corresponding lemmas in the previous sections, and is omitted, shows that there exist a bounding function for this relation.

Lemma 4.3: The function $\zeta([\Delta|r\rangle)(\rho) = \rho \diamond (\rho \dot{+} L\Delta)^r$ is a bounding function for the relation $[\Delta|r\rangle$, L being the Lipschitz constant that appears in (5).

Note that, as the notation suggests, the previous two relations can be regarded as special case of this one for $\Delta = 0$ and $r = 1$, respectively. The bound functions for the two relations are similarly special cases of $\zeta([\Delta|r\rangle)$.

Transitive Closure Properties: The definition of this new dependence naturally poses the problem of its transitive closure properties, in particular its composition law. That is, if $x[\Delta|r\rangle y$ and $y[\Gamma|q\rangle z$, what can be inferred about the relation between x and z? The following theorem provides the answer:

Theorem 4.2: If $x[\Delta|r\rangle y$ and $y[\Gamma|q\rangle z$, then $x[\Psi|s\rangle z$ with

$$s = rq \tag{21}$$

$$\Psi = (\Delta^r + \Gamma)^{1/r}. \tag{22}$$

Proof: Since $x[\Delta|r\rangle y$, it is $y \leq (x + \Delta)^r$ and, since $y[\Gamma|q\rangle z$, it is $z \leq (y + \Gamma)^q$. Putting together the two inequalities, we obtain

$$z \leq ((x + \Delta)^r + \Gamma)^q. \tag{23}$$

The theorem, on the other hand, implies that $z \leq (x + \Psi)^s$, therefore we need to find values of s and Ψ such that

$$(x + \Psi)^s \geq ((x + \Delta)^r + \Gamma)^q. \tag{24}$$

Defining $u = s/q$, and raising both sides of the inequality to the power $1/q$, one obtains

$$(x + \Psi)^u \geq (x + \Delta)^r + \Gamma. \tag{25}$$

For $x = 0$, this inequality becomes

$$\Delta^r + \Gamma \leq \Psi^u \tag{26}$$

which is true for

$$\Psi \geq (\Delta^r + \Gamma)^{1/u}. \tag{27}$$

We want to show that, if $u = r$, the inequality holds for all x. Call $f(x) = (x + \Delta)^r + \Gamma$, and $g(x) = (x + \Psi)^u = (x + \Psi)^r$. The previous inequality tells us that $g(0) - f(0) \geq 0$. Computing the derivatives one gets $f'(x) = r(x + \Delta)^{r-1}$ and $g(x) = r(x + \Psi)^{r-1}$. Noting that (26) for $u = r$ implies $\Psi \geq \Delta$, one sees that $g'(x) \geq f'(x)$ and, therefore $g'(x) - f'(x) \geq 0$. Since $g(x) - f(x)$ is positive for $x = 0$ and its derivative is positive, it is never negative. Therefore $(x + \Delta)^r + \Gamma \leq (x + \Psi)^u$ for all x. Finally, note that $u = r$ implies $s = rq$. □

The properties of the quasiscale dependence relation can then be summarized as follows:

Self-dependence: For all x, $x[0|1\rangle x$.

Transitivity: If $x[\Delta|r\rangle y$ and $y[\Gamma|q\rangle z$, then $x[(\Delta^r + \Gamma)^{1/r}|rq\rangle z$.

The transitive closure for the quasiscale dependence is defined as:

Definition 4.8: Let $X = \{x_1, x_2, \ldots, x_n\}$ be a set of features computed on the same database (or a decomposition of a single feature computed on the database). Given a feature $x_i \in X$, the (Δ, r)-transitive closure of x_i is the set:

$$[x_i]_{\Delta|r} = \{x_k \in X\colon x_i[\Gamma|q\rangle x_k,\ \Gamma \leq \Delta,\ q \geq r\}. \tag{28}$$

V. Probabilistic Dependence

All the previous cases of dependence between features were deterministic: it was possible to find a constraint between distances relative to the two features that was satisfied for all possible images. This allowed the definition of k-nearest neighbors algorithms that guaranteed that the k images closest to the query with respect to the complete distance were returned, even if the database was first filtered using only part of the features.

In some cases it is necessary to relax this assumption and define the relation between the two features x_1 and x_2 in terms of probability. Correspondingly, the k-nearest neighbors algorithm will return the k images closer to the query with a certain probability, and the cost of the algorithm will increases with the probability that the result will be correct. While in the previous case the decision of whether to separate two features was based only on cost considerations, in this case a new variable has to intervene: the probability that one or more of the k most significant images will not be returned.

A full treatment of the probabilistic case is very complex. In this section we will consider a rather schematic treatment under simplified but, hopefully, not unrealistic assumptions. As before, assume that we have a table $T(h : \mathbb{N}, x_1 : X_1, x_2 : X_2)$, where X_1, X_2 are feature types, and x_1, x_2 are features. The problem is whether the table should be split into two tables $T_1(h, x_1)$ and $T_2(h, x_2)$ with $T_1.d = d_1$ and $T_2.d = d_2$. As usual, let h_0 be the handle of the query image, and let $s_1 = T_1[h_0].d$, $s_2 = T_2[h_0].d$, and $s = T[h_0].d$.

Assume that the difference between the score of two images is normally distributed, with variance σ_1^2 for features of type ϕ_1 and variance σ_2^2 for features of type ϕ_2, that is

$$s_1(u) - s_1(v) \sim N(0, \sigma_1) \tag{29}$$

$$s_2(u) - s_2(v) \sim N(0, \sigma_2) \tag{30}$$

and, consequently, $s(u) - s(v) \sim N(0, \sigma)$, with $\sigma^2 = \sigma_1^2 + \sigma_2^2$.

The question of whether this assumption can be justified is still quite open. For Minkowski distances, the law of large numbers justifies this assumption under the (weaker) hypothesis that the feature components be identically distributed. In this case, and for an L_p distance, the values $(s(u))^p - (s(v))^p$ are normally distributed, and the considerations that follow hold using s^p as a scoring function. Empirical measures ([23], see also the Appendix) seem to validate this assumption to a certain extent.

Given a k-nearest neighbors query and a bound p on the probability of error that one is willing to accept, we need to determine the number h of images ($h > k$) such that the result of a h-nearest neighbors query on the table T_1 will contain with a probability $1 - p$ all the k images closest to the query according to the whole feature.

We start with the following problem: given two images that, in the complete table T, are at a distance α from each other, what is the probability that they will be order-swapped when the same query is made on the table T_1? In other words, consider two images u and v such that $s(u) - s(v) = \alpha$ (so that, in order of distance from the query, u comes after v), what is the probability that $s_1(u) < s_1(v)$ (so that in the query on the table T_1, u will come before v?). The relation between distances gives

$$\alpha = s(u) - s(v) = (s_1(u) - s_1(v)) + (s_2(u) - s_2(v)). \tag{31}$$

The images will then be swapped if $s_2(u) - s_2(v) > \alpha$. If $p_s(\alpha)$ is the probability of swapping two images given that their distance is α, then

$$\begin{aligned} p_s(\alpha) &= \mathbb{P}[s_2(u) - s_2(v) > \alpha] \\ &= \frac{1}{\sqrt{2\pi}\sigma_2} \int_\alpha^\infty \exp\left(\frac{x^2}{\sigma_2^2}\right) dx \\ &= \frac{1}{\sqrt{2\pi}\sigma_2} \operatorname{erf}\left(\frac{\alpha}{\sigma_2}\right). \end{aligned} \tag{32}$$

In order to solve the original problem, we still a second element, namely an answer to the following question: given the kth and the hth image in the database, what is the probability distribution of the difference in their distance from the query? That is, if x_i is the feature representation of the ith image in order of distance from the query, what is the probability distribution of $s(x_h) - s(x_k)$? The probability distribution of $s(x_i)$, for a generic i is derived in the Appendix under the hypothesis that the *a priori* distribution of $s(x)$ is uniformly distributed between two values a and b, and it turns out to be

$$p_i^n(\beta) = C_i^n \left(\frac{b - \beta}{b - a}\right)^{n-1} \log^{i-1} \frac{b - \beta}{b - a} \tag{33}$$

where n is the number of images in the database and C_i^n is a suitable normalization constant. The two values a and b can be set in such a way that the variance of the difference between two uniformly distributed distance be equal to the variance of the normal distribution hypothesized for $s(u) - s(v)$, that is, to σ. The difference between two samples taken from a uniform distribution of width Δ, has a triangular distribution of variance $2/3\Delta^2$, so that the uniform *a priori* distribution of the distances should have $\Delta = \sqrt{3\sigma^2/2}$. The center of the distribution depends on the particular distance and for now we will simply indicate it as c. Setting $a = c - \Delta/2$ and $b = c + \Delta/2$ one obtains:

$$p_i^n(\beta) = C_i^n \left(\frac{c + \frac{\sqrt{3\sigma^2/2}}{2} - \beta}{\sqrt{\frac{3}{2}\sigma^2}}\right)^{n-1} \log^{i-1} \frac{c + \frac{\sqrt{3\sigma^2/2}}{2} - \beta}{\sqrt{\frac{3}{2}\sigma^2}} \tag{34}$$

TABLE I
SUMMARY OF THE MAIN PROPERTIES OF THE DEPENDENCIES INTRODUCED IN THE PAPER

Dependence	Property	Formula
Scale Difference	Self-Dependence	$\forall x\ x\,[\|1\rangle x$
	Anti-Symmetry	$\forall x, y\ x\,[\|r\rangle y, r > 1 \ \Rightarrow \nexists q > 1 : y\,[\|r\rangle x$
	Transitivity	$\forall x, y, z\ \ x\,[\|r\rangle y\ \wedge\ y\,[\|q\rangle z\ \Rightarrow x\,[\|rq\rangle z$
Quasi-functional	Self-Dependence	$\forall x\ x\,[0\|\rangle x$
	Symmetry	$\forall x, y\ x\,[\Delta\|\rangle y \Rightarrow y\,[\Delta\|\rangle x$
	Monotonicuty	$x\,[\Delta\|\rangle y \Rightarrow\ \ \forall \Gamma > \Delta\ x\,[\Gamma\|\rangle y$
	Transitivity	$\forall x, y, z\ \ x\,[\Delta\|\rangle y\ \ \wedge\ \ y\,[\Gamma\|\rangle z \Rightarrow x\,[\Delta + \Gamma\|\rangle z$
Quasi-Scale	Self-Dependence	$\forall x\ x\,[0\|1\rangle x$
	Transitivity	$\forall x, y, z\ \ x\,[\Delta\|r\rangle y\ \ \wedge\ \ y\,[\Gamma\|\rangle qz \Rightarrow x\,[(\Delta^r + \Gamma)^{1/r}\|rq\rangle z$

with

$$c - \frac{\sqrt{3\sigma^2/2}}{2} \le \beta \le c + \frac{\sqrt{3\sigma^2/2}}{2}.$$

The distribution of the distance between the hth and the kth image in the database can then be computed as

$$p^n_{k,\,h}(\alpha) = \int p^n_k(\beta) p^n_h(\beta + \alpha)\, d\beta \tag{35}$$

with $\alpha \in [0,\, b - a]$.

The probability that images k and h will be swapped is

$$p^s_{k,\,h}(\sigma_1,\, \sigma_2) = \int p^n_{k,\,h}(\alpha) p_s(\alpha)\, d\alpha \tag{36}$$

where $p_s(\alpha)$ is the probability that two images will be swapped given that the distance between them is α. Using (32), this gives:

$$p^s_{k,\,h}(\sigma_1,\, \sigma_2) = \frac{1}{\sqrt{2\pi}\sigma_2} \int p^n_{k,\,h}(\alpha) \operatorname{erf}\left(\frac{\alpha}{\sigma_2}\right) d\alpha. \tag{37}$$

Note that this expression depends only on σ_1 and σ_2. The dependence from σ_1 is in the determination of the bounds a and b of the uniform distribution necessary for computing $p^n_{k,\,h}$, since $\sigma^2 = \sigma_1^2 + \sigma_2^2$.

If the database is large, then $p^n_{k,\,h}(\alpha) \approx \delta(\alpha - (m^n_h - m^n_k))$ and the probability to swap the kth and the hth image is

$$\begin{aligned} p^s_{k,\,h}(\sigma_1,\, \sigma_2) &= \frac{1}{\sqrt{2\pi}\sigma_2} \int \delta(\alpha - (m^n_h - m^n_k) \operatorname{erf}\left(\frac{\alpha}{\sigma_2}\right) d\alpha \\ &= \frac{1}{\sqrt{2\pi}\sigma_2} \operatorname{erf}\left(\frac{m^n_h - m^n_k}{\sigma_2}\right). \end{aligned} \tag{38}$$

Consider an algorithm like those presented in the previous sections in which one solves a k'-nearest neighbors problem (for some suitable k') in the reduced feature space in order to solve the k-nearest neighbor problem in the full feature space. In the case of a probabilistic dependence between the two features it is not possible to guarantee that the first k images will be retrieved for any value of k', but it is possible to provide a bound on the probability of error. In particular, given a probability p^0 of leaving one of the first k images out of the selection, the value k' will be the smallest value for which

$$p^s_{k,\,k'}(\sigma_1,\, \sigma_2) \le p^0. \tag{39}$$

Once this condition is set, the algorithm for dividing a feature in two part when the two have a probabilistic dependence is essentially the same as in the previous cases.

VI. Cost-Based Reduction to Normal Form

The previous sections have introduced three forms of dependence between features or parts of a feature structure (the probabilistic dependence can be reduced to a deterministic dependence by fixing an upper bound to the probability of swapping images): the scale difference dependence, the partial functional dependence, and the quasi–scale dependence, together with their properties, which are summarized for convenience in Table I.

These properties are used to derive transitive closures relative to a subset of the features that satisfy certain cost constraints. So far, these transitive closures have been defined independently for the three dependencies. In order to arrive at a unified definition of cost-bound transitive closure for a given subset of the features, it is necessary to analyze Algorithm 1 and the cost associated to it.

Assume that the database has been divided in two tables $T_1(h : \mathbb{N},\, x_1 : X_1)$ and $T_2(h : \mathbb{N},\, x_2 : X_2)$, where x_1 has dimensionality d_1, x_2 has dimensionality d_2, and the full dimensionality of the feature space is $d = d_1 + d_2$.

Let $\mathcal{K}(N,\, k,\, d)$ be the cost of a k-nearest neighbors search and $\mathcal{R}(N,\, r,\, d)$ be the cost of a range search of radius r in a space of dimensionality d for a database of N images.

The cost of the various steps of Algorithm 1 is the following:

1) a k-nearest neighbors search in T_1, with a cost $\mathcal{K}(N,\, k,\, d_1)$;
2) constant time access to the results;
3) constant time function computation;
4) a range query on the table T_1 with radius $\zeta(\mathfrak{r})(\tilde{\rho})$, with cost $\mathcal{R}(N,\, \zeta(\mathfrak{r})(\tilde{\rho}),\, d_1)$; this will result in $k' > k$ images in a table Q;
5) a join between on the identifier field of the table Q and the table T_2, whose identifiers are a superset of those of Q; using a hash join, the cost of this operation is linear in the size of Q, that is, in k';
6) a linear scan (worst case scenario) of the table resulting from the join, which contains k' entries; this also requires a time linear in k'.

The cost of the whole algorithm is then

$$\mathcal{K}(N,\, k,\, d_1) + \mathcal{R}(N,\, \zeta(\mathfrak{r})(\tilde{\rho}),\, d_1) + Ck'(\zeta(\mathfrak{r})(\tilde{\rho})) \tag{40}$$

where the notation $k'(\zeta(\mathfrak{r})(\tilde{\rho}))$ serves to remind that the number of images returned by the range query is a function of the search radius. For a given database size and feature space dimension, this cost depends on two variables: the dimension d_1 of the reduced feature space, and the radius-increasing function f. Consequently, we define the *normalized cost* of a database search as

$$\mathcal{N}(N, f, d) = \mathcal{K}(N, k, d_1) + \mathcal{R}(N, f(\tilde{\rho}), d_1) + Ck'(f(\tilde{\rho})). \quad (41)$$

If the original feature space has such high dimensionality that no indexing scheme works satisfactorily on it, no search is faster than a linear scan, which has a cost CN. In this case, normalizing the database leads to a gain $N - \mathcal{N}(N, \zeta(\mathfrak{r}), d_1)$.

The function $\zeta(\mathfrak{r})$ depends on the relation between the features that are being split and is given, for the general relation $\mathfrak{r} = [\Delta|r\rangle$, by

$$\zeta([\Delta|r\rangle)(\rho) = \rho \diamond (\rho \dot{+} L\Delta)^r. \quad (42)$$

The objective of cost-based reduction to normal form is to find a decomposition of the database tables such that $\mathcal{N}(N, \zeta(\mathfrak{r}), d)$ is as small as possible.

In order to formalize this problem we need a few more definitions. Let $\mathfrak{r} = [\Delta|r\rangle$ be any of the previous four dependence relations, and indicate with $x[\mathfrak{r}\rangle y$ a generic relation between x and y.

Definition 6.1: Let x a feature from a database. The ϕ transitive closure of x is the set

$$[x]^\phi = \{y: \exists\, \mathfrak{r}\, x[\mathfrak{r}\rangle y \ \wedge \ f_\mathfrak{r}(\rho) \le \phi(\rho)\}. \quad (43)$$

The definition is extended immediately to sets of features $Y = \{x_1, \ldots, x_l\}$ as

$$[Y]^\alpha = \bigcup_{x \in Y} [x]^\alpha. \quad (44)$$

In other words, the set $[x]^\alpha$ is the set of all the features that depend on x through a relation whose cost-increasing function is bounded by a given function ϕ.

Definition 6.2: Given a set of feature types $X = \{X_1, \ldots, X_n\}$ on a database, an α-seed of the set X is a set $Y \subseteq X$ such that $[Y]^\alpha = X$.

A key Y for a database of features X can be used to index the images in the database with a guaranteed bound on the cost of the algorithm using the methods illustrated in the previous sections.

The general table decomposition problem can then be formulated in the following way.

> Given a database composed of a table $T(\mathbb{N}, X_1, \ldots, X_m)$, find a set of tables $K(\mathbb{N}, Y_1, \ldots, Y_p)$, $T_i(\mathbb{N}, W_{i,1}, \ldots, W_{i,p_i})$ such that
> 1) $X = \bigcup_i Y_i \cup \bigcup_{i,j} W_{i,j}$;
> 2) Every X_i belongs either to K or to exactly one of the T_is;
> 3) K is an α-key for T;
> 4) The cost $\mathcal{N}(N, \zeta, d)$, where ζ is the function induced by the dependence of the T_is on K, and d is the dimension of the feature space of K, is minimized.

TABLE II
AVERAGE AND VARIANCE OF THE DISTANCE BETWEEN A QUERY AND THE IMAGES IN A DATABASE

d	μ	σ^2
2	0.3664	0.0298
5	0.3904	0.0125
10	0.4015	0.0060
15	0.4031	0.0040
20	0.4042	0.0029
30	0.4053	0.0020
50	0.4066	0.0012
100	0.4074	0.0006

A. An Example

The general table decomposition problem introduced in this section requires, for its complete solution, a cost model of k-nearest neighbors queries and of range queries in high dimensional feature spaces. Several such models have been proposed in the literature [2], [15], [6]. The subject of cost models for nearest neighbor queries is a complex one, and we will not consider it in this paper.

Rather, we will use some simple approximation of the model reported in [2]. In particular, we will assume a database of N images and a disk page size of $S = 360$ images. Under these circumstances, the data reported in [2] are reasonably well approximated assuming that a k-nearest neighbor in a feature space of d dimensions will access a number of pages given by

$$\mathcal{K}(N, k, d) = P \log(k) \frac{1 - \exp(-\lambda N/10^6)}{1 + \exp(-\alpha(d - 16))} \quad (45)$$

with $\lambda = \log(2)$, and $\alpha = \log(99)/16$. The number of pages accessed by a range query of radius r is also taken from [2], and is given by

$$\mathcal{R}(N, r, d) = \sum_{t=0}^{d'} \binom{d'}{t} V\left(S^t\left(\left[\tfrac{1}{2}, \ldots, \tfrac{1}{2}\right], r\right) \cup \Omega\right) \quad (46)$$

where $S^k([1/2, \ldots, 1/2], r)$ is the sphere of center $[1/2, \ldots, 1/2]$ and radius r in a t dimensional space, Ω is the unit cube in which the database is contained, and V is the volume operator.

In order to apply the cost formula, we still need to determine the average distance of the kth image from the query in the d_1-dimensional space. In the hypothesis that the distribution of distances is a δ-distribution, this value is simply m_k^n, given in (61). This value, however, depends on the extrema of the distribution of distances a and b which, in turn, depend on the dimensionality of the feature space. Using randomly generated points in the unit cube, one can derive the average and variance of the distance between points which, under the homogeneity hypotheses in [6], can be taken as an approximation of the distribution of the distance between a query and an image. Data relative to some dimensionality values are reported in Table II and Fig. 1. Using the relations $a = \mu - \sqrt{3}\sigma/2$ and $b = \mu + \sqrt{3}\sigma/2$, this allows the computation of the extrema a and b and, consequently, of the average distance of the kth image m_k^N for a given k and for a given dimensionality.

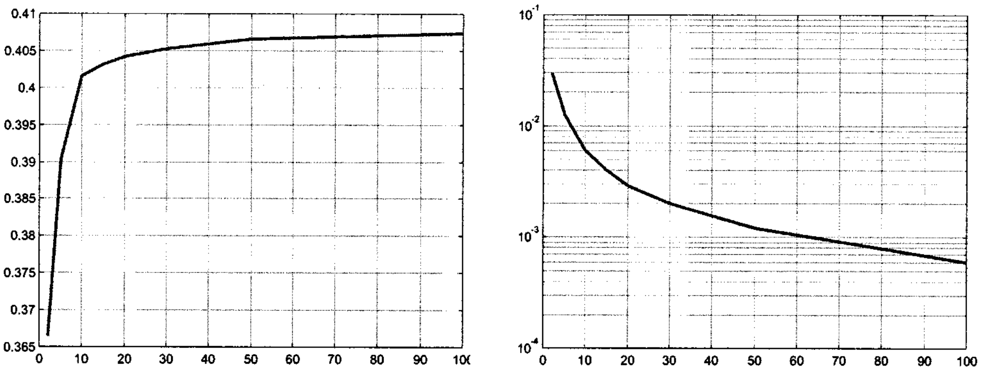

Fig. 1. (a) Average and (b) variance of the distance between points as a function of the dimensionality of the feature space.

Finally, the number of images k' returned by a query of radius r is given by the highest value h such that $m_h^N \leq r$ that is,

$$k'(r, N, d) = \left\lfloor 1 - (N-1) \log \frac{b-r}{b-a} \right\rfloor. \qquad (47)$$

Consider now a database with 10^7 images in which the features stay in certain dependence relations. We will consider, for the sake of simplicity, only limiting functions ϕ (see definition 6.1) of the type $\phi(r) = \alpha + r^\beta$.

Originally the database has a table with three features: $T(h, x, y, z)$. The feature x has dimension 3, the feature y has dimension 5, and the feature z has dimension 15. The relations given between the three by the feature designer are

$$\begin{aligned} &x\ [0.01|1\rangle\ y \\ &y\ [0.01|5\rangle\ z \end{aligned} \qquad (48)$$

which, by transitivity, leads to $x[0.02|1\rangle z$. The original table is too big to allow efficient indexing, and a search will require $10^7/360 \approx 28\,000$ page accesses.

There are two ways in which the table can be divided in two, using either x alone as a key, or y and y as a key. Consider the first alternative: we have the tables $K(h, x)$ and $T(h, y, z)$. The relation between the two tables must take into consideration the relation between x and y as well as that between x and z. For, say, a 30 nearest neighbors query, the relation between x and y gives $\mathcal{N} \approx 234$ page accesses, while the relation between x and z gives $\mathcal{N} \approx 8800$ page accesses, which represents the worst case and, therefore, the cost of a 30 nearest neighbor search in the decomposed database.

The second decomposition has $K(h, x, y)$ and $T(h, z)$. The best relation between the two table consists in searching the key using y and then extending the search to z. The relation between y and z leads to a cost $\mathcal{N} \approx 913$ pages. The latter is therefore the most convenient decomposition.

VII. Conclusions

In this paper we have presented some basic principles for the design of database schemas in multimedia databases. The main concept upon which our design model is based is that of *dependence* of features. We identified four kinds of dependencies between substructures of the same feature structure and showed how these dependencies can be exploited for the design of efficient search algorithms.

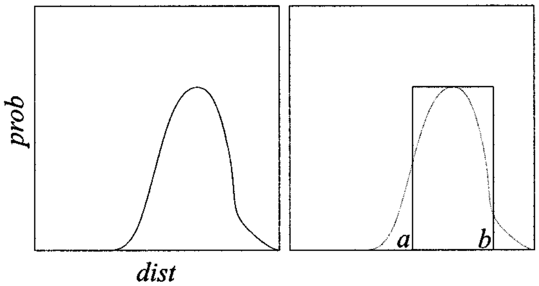

Fig. 2. *A priori* distribution of the distance between the query an arbitrary image in the database.

Based on these algorithm, we have introduced the notion of cost-bounded transitive closure and the related notion of cost-bounded normal form, as well as some design principles useful to reduce a database table into cost-bounded normal form.

The relations described in this paper, and the design techniques that were derived from them, are but a scratch in the surface of the general problem of database design. Issues that are still waiting for an answer include completeness and consistency of groups of transitive closure axioms, as well as relations based on criteria other than the functional relation of distance measures.

Appendix
Distance Distribution in Databases

Consider a database containing n images and in which a distance function d is defined. Assume that the images $[u_1, \ldots, u_n]$ are ordered by distance with the query q, so that $s(u_i) \leq s(u_{i+1})$, where $s(u) = d(q, u)$. The question we are trying to answer in this appendix is the following: what is the probability distribution of the distance between the query q and the image u_k? In other terms, what is the probability distribution of the kth statistics of $[u_1, \ldots, u_n]$?

In order to derive such a distribution it is first necessary to define an *a priori* probability distribution for $s(u)$: the distance between the query and an arbitrary image in the database. Stricker and Orengo [23] determined experimentally that in many cases the distribution has the shape of Fig. 2. We will make a very rough approximation of this distribution: we will assume that

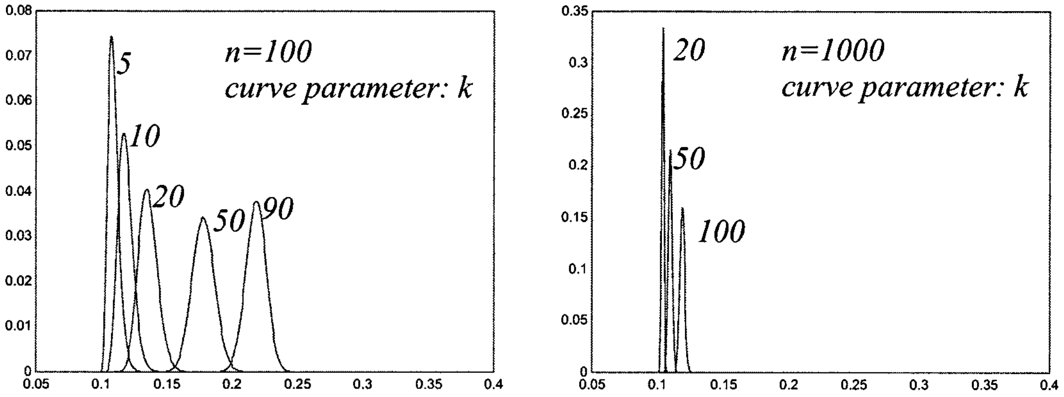

Fig. 3. Average distance between the query and the kth image in order of distance for various values of k and for a database of size (a) 100 and (b) 1000.

the probability density of the distance is uniform between two distance values a and b and zero everywhere else [Fig. 2(b)].

We start by determining the probability density of the minimum of two variables uniformly distributed in (a, b), which is given by

$$p_{\min(x,\,y)}(\alpha) = p_x(\alpha)\mathbb{P}[y \geq \alpha] + p_y(\alpha)\mathbb{P}[x \geq \alpha]. \qquad (49)$$

Because of the uniform distribution, it is $p_x(\alpha) = p_y(\alpha) = 1/(b-a)$ and $\mathbb{P}[x \geq \alpha] = \mathbb{P}[y \geq \alpha] = (b-\alpha)/(b-a)$, therefore

$$p_{\min(x,\,y)}(\alpha) = \frac{2}{b-a}\,\frac{b-\alpha}{b-a}. \qquad (50)$$

The relation can be generalized to the minimum of n values as

$$\begin{aligned} p_{\min(x_1,\,\ldots,\,x_n)}(\alpha) &= \sum_{i=1}^{n} p_{x_i}(\alpha) \prod_{j\neq i} \mathbb{P}[x_j > \alpha] \\ &= \frac{n}{b-a}\left(\frac{b-\alpha}{b-a}\right)^{n-1}. \end{aligned} \qquad (51)$$

Similar equations can be written for the probability density of the minimum of $(x_1, \ldots, x_n)$ conditioned to the minimum being greater than α, that is,

$$\begin{aligned} &p_{\min(x_1,\,\ldots,\,x_n)}(\beta|\min > \alpha) \\ &\quad= \sum_{i=1}^{n} p_{x_i}(\beta|\min > \alpha) \prod_{j\neq i} \mathbb{P}[x_j > \beta|\min > \alpha]. \end{aligned} \qquad (52)$$

Where, this time

$$\mathbb{P}[x_j > \beta|\min > \alpha] = \frac{b-\beta}{b-\alpha} \qquad (53)$$

for $a \leq \alpha$, $\beta \leq b$, and $b \geq \alpha$. Therefore

$$p_{\min(x_1,\,\ldots,\,x_n)}(\beta|\min > \alpha) = \frac{n}{b-\alpha}\left(\frac{b-\beta}{b-\alpha}\right)^{n-1}. \qquad (54)$$

Consider now the different statistics of $(x_1, \ldots, x_n)$. The first statistics p_1^n is the minimum, and has the distribution (51). Consider then the second statistics p_2^n. An event that leads to the second statistics having the value β is 1) the minimum of $(x_1, x_2, \ldots, x_n)$ has value α and 2) the minimum of $(x_2, \ldots, x_n)$ has value β conditioned to its value being greater than α, that is

$$p_2^n(\beta) = \int_{\alpha=a}^{b} p_1^n(\alpha)p_1^{n-1}(\beta|\min > \alpha)\,d\alpha. \qquad (55)$$

We make an approximation here: the second factor within the integral is p_1^{n-1}, since one is interested in the minimum of the remaining elements after the minimum has been removed. If, however, we are looking for the kth statistics with $k \ll n$ (as is usually the case when n is the number of images in the database and k is the number of images retrieved), one can think that adding a new image to the database will make little difference and therefore that $p_1^{n-1} \approx p_1^n$. The integral then becomes

$$\begin{aligned} p_2^n(\beta) &= \int_{\alpha=a}^{b} p_1^n(\alpha)p_1^n(\beta|\min > \alpha)\,d\alpha \\ &= \frac{n^2}{b-a}\int_a^b \frac{(b-\alpha)^{n-1}}{(b-a)^{n-1}}\,\frac{(b-\beta)^{n-1}}{(b-\alpha)^n}\,d\alpha \\ &= \frac{n^2}{(b-a)^n}(b-\beta)^{n-1}\int_a^b \frac{d\alpha}{b-\alpha} \\ &= \frac{n^2}{(b-a)^n}(b-\beta)^{n-1}\log\frac{b-a}{b-\beta}. \end{aligned} \qquad (56)$$

The distribution of the kth statistics is defined in a similar way as a function of the distribution of the kth statistics:

$$p_k^n(\beta) = \int p_{k-1}^n(\alpha)p_1^n(\beta|\min > \alpha)\,d\alpha. \qquad (57)$$

It is possible to verify that the solution to this iteration is given by distributions of the type

$$p_k^n(\beta) = C_k^n\left(\frac{b-\beta}{b-a}\right)^{n-1}\log^{k-1}\frac{b-\beta}{b-a} \qquad (58)$$

where C_k^n is a normalization constant equal to $C_k^n = n^k/(k-1)!1/(b-a)$.

The behavior of some of these functions is shown in Fig. 3 for several values of k and n. It is evident that, as n grows, the probability densities for low values of k are more and more concentrated. In some cases, it is possible to consider p_k^n as a δ-distribution of the form $p_k^n(\beta) = \delta(\beta - m_k^n)$ for some value

m_k^n. The value m_k^n can be chosen as the value in which p_k^n attains the maximum. Setting $u = (b - \beta)/(b - a)$ one obtains

$$p_k^n(u) = C_k^n u^{n-1} \log^{k-1} u \tag{59}$$

and

$$\begin{aligned} &\frac{d}{du}\, p_k^n(u) \\ &= C_k^n \left[(n-1)u^{n-2} \log^{k-1} u + \frac{1}{u}\,(k-1)u^{n-1} \log^{k-2} u \right] \\ &= C_k^n u^{n-2} \log^{k-2} u[(n-1)\log u + (k-1)] \end{aligned} \tag{60}$$

which is zero for $u = \exp(-(k-1)/(n-1))$. This gives

$$m_k^n = b - (b-a)\exp - \frac{k-1}{n-1}. \tag{61}$$

Consider now two images of order h and k, with $h > k$. The probability density of the distance between them is

$$p_{h,\,k}^n(\alpha) = \int p_k^n(\beta) p_h^n(\beta + \alpha)\, d\beta \tag{62}$$

that is, in the δ-distribution approximation

$$p_{h,\,k}^n(\alpha) = \delta(\alpha - (m_h^n - m_k^n)). \tag{63}$$

This is, of course, an expected result: in the case in which the variance of the distances is extremely small, the images can be considered to be placed deterministically at a distance corresponding to the difference of their average distance from the query.

REFERENCES

[1] G. G. A. Albano and R. Orsini, "Fibonacci: A programming language for object databases," *VLDB J.*, vol. 4, pp. 402–444, 1995.

[2] S. Berchtold, C. Böhm, D. A. Keim, and H.-P. Kriegel, "A cost model for nearest neighbor search in high dimensional data space," in *Proc. ACM Int. Conf. Principles of Database Systems, PODS '97*, Tucson, AZ, 1997.

[3] C. Carson, M. Thomas, S. Belongie, J. M. Hellerstein, and J. Malik, "Blobworld: A system for region-based image indexing and retrieval," in *Proc. Visual '99, Int. Conf. Visual Information Systems*, 1999.

[4] K. Chakrabarti, K. Porkaew, and S. Mehrotra, "Efficient query refinement in multimedia databases," in *Proc. Int. Conf. Data Engineering*, 2000.

[5] P. Ciaccia, M. Patella, and P. Zezula, "M-tree: An efficient access method for similarity search in metric spaces," in *Proc. 23th VLDB Conf.*, Athens, Greece, 1997.

[6] ——, "A cost model for similarity queries in metric spaces," in *Proc. 17th Symp. Principles of Database Systems*, 1998, pp. 59–68.

[7] D. Dubois and H. Prade, "A review of fuzzy set aggregation connectives," *Inform. Sci.*, vol. 36, pp. 85–121, 1985.

[8] J. P. Eakins, J. M. Boardman, and M. E. Graham, "Similarity retrieval of trademark images," *IEEE Multimedia*, vol. 5, no. 2, 1998.

[9] J. P. Eakins and M. E. Graham, "Content-based image retrieval," Report to the JISC Technology Application Programme, Inst. Image Data Res., Univ. Northumbria at Newcastle, U.K., 1999.

[10] P. G. Enser and C. G. McGregor, "Analysis of visual information retrieval queries," British Library, British Library Res. Develop. Rep. 6104, 1992.

[11] R. Fagin, "Combining fuzzy information from multiple systems," in *Proce. 15th ACM Symp. Principles of Database Systems*, Montreal, QC, Canada, 1996, pp. 216–226.

[12] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker, "Query by image and video content: The QBIC system," *IEEE Computer*, 1995.

[13] A. Gupta and S. Santini, "Toward feature algebras in visual databases: The case for a histogram algebra," in *Proc. IFIP Working Conf. Visual Databases (VDB5)*, Fukuoka, Japan, May 2000.

[14] S. K. Hastings, "Query categories in a study of intellectual access to digitized art images," in *ASIS '95: Proc. 58th ASIS Annu. Meeting*, vol. 32, 1995, pp. 3–8.

[15] J. H. Hellerstein, E. Koutsoupias, and C. Papadimitriou, "On the analysis of indexing schemes," in *Proc. PODS '97, ACM Conf. Principles of Database Systems*, Tucson, AZ, 1997, pp. 249–256.

[16] T. Kato, "Database architecture for content-based image retrieval," *Proc. SPIE*, vol. 1662, pp. 112–123, 1992.

[17] B. S. Manjunath and W. Y. Ma, "Texture features for browsing and retrieval of image data," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 18, pp. 837–842, Aug. 1996.

[18] D. Papadias, N. Karacapilidis, and D. Arkoumanis. (1998) Processing fuzzy spatial queries: A configuration similarity approach. [Online]. Available: http://www.cs.ust.hk/faculty/dimitris/

[19] V. Pestov, "On the geometry of similarity search: dimensionality curse and the concentration of measure," School Math. Comput. Sci., Victoria Univ. , Wellington, New Zealand, Tech. Rep. RP-99-01, 1999.

[20] K. Porkaew, S. Mehrotra, M. Ortega, and K. Chakrabarti, "Similarity search using multiple examples in MARS," in *Proc. Int. Conf. Visual Information Systems*, 1999.

[21] N. Roussopoulos, F. Kelley, and F. Vincent, "Nearest neighborhood queries," in *Proc. ACM SIGMOD Conf.*, 1995.

[22] S. Santini, *Exploratory Image Databases: Content Based Retrieval*. New York: Academic , 2001.

[23] M. Stricker, "Bounds for the discrimination power of color indexing techniques," *Proc. SPIE*, 1994.

[24] M. Jeffrey Ullman, *Principles of Database Systems*, 2nd ed. Rockville, MD: Computer Science , 1982.

[25] D. White and R. Jain, "Similarity indexing: Algorithms and performance," *Proc. SPIE*, vol. 2670, pp. 62–73, 1996.

Simone Santini (S'96–M'98) received the Laurea degree from the University of Florence, Italy, in 1990, and the M.Sc. and the Ph.D. degrees from the University of California at San Diego (UCSD), La Jolla, in 1996 and 1998, respectively.

In 1990, he was Visiting Scientist at the Artificial Intelligence Laboratory at the University of Michigan, Ann Arbor, and in 1993, Visiting Scientist at the IBM Almaden Research Center, San Jose, CA. He is currently a Researcher at the National Center for Microscopy and Imaging Research at UCSD. His current research interests are interactive image and video databases, data models and query models for structured and numerical data (typically image and video features) and for temporal data, and multi-modal information management. In particular, he is interested in the characterization of the process by which different media, user characteristics, and the circumstances of a query contribute to the emergence of meaning.

Amarnath Gupta received the B.Tech. degree in mechanical engineering from the Indian Institute of Technology, Kharagpur, in 1984, and the M.S. degree in biomedical engineering from University of Texas at Arlington in 1987 and Ph.D. degree (engineering) in computer science from Jadavpur University, India, in 1994.

He is an Assistant Research Scientist at the Center for Computational Science and Engineering, University of California at San Diego (UCSD), La Jolla. Before joining UCSD, he was a Scientist at Virage, Inc., where he worked on multimedia information systems. His current research interests are in multimedia and spatiotemporal information systems, heterogeneous information integration and scientific databases.