SIMILARITY-BASED OPERATORS AND QUERY OPTIMIZATION
FOR MULTIMEDIA DATABASE SYSTEMS

Solomon Atnafu, Lionel Brunie
Lab. d'Ingénierie des Systèmes d'Information (LISI), INSA de Lyon, France
20 Av. A. Einstein; 69621 Villeurbanne
{satnafu, Lionel.Brunie}@lisi.insa-lyon.fr

Harald Kosch
Institute of Information Technology,
University Klagenfurt, Austria,
Universitätsstr. 65-67; 9020 Klagenfurt
harald@itec.uni-klu.ac.at

ABSTRACT
The many successful research results in the domain of computer vision have made similarity-based data retrieval techniques a promising approach. As a result, the integration of similarity-based retrieval techniques of multimedia data into DBMSs is currently an active research issue. In this paper, we first illustrate the importance of similarity-based operations. Then, we present our image data repository model that supports similarity-based operations conveniently under an object-relational database paradigm. Furthermore, we present novel similarity-based operators on image tables and study their properties. Finally, based on the properties of the operators identified, we derive algebraic rules that are useful for similarity-based query optimization and will introduce a cost model for an implementation of one of the major similarity-based operators.

1. MOTIVATION
The use of low-level contents of multimedia data for its identification, storage and operation purpose has been one of the major issues of research in the last decade. As a result, a number of research prototypes, applications, and commercial systems that support low-level content manipulation have been developed [1, 2, 3, 4, 5]. These works practically demonstrated that, the need for an automatic extraction, classification, and manipulation of the content of multimedia data is of critical significance for an efficient multimedia data management. The problem of extracting the content of an image and identifying the proper techniques for comparing whether two images are alike or not is more of a work in the field of visual recognition.

In this paper, we focus on the management of content-based image databases. We considered images, because image data is the most common and the widely used media data. Moreover, the volume of image data actually collected and stored in image databases for different applications is so huge, that the need for its systematic and efficient management has become very crucial. In the Database Management domain, a number of works are initiated to integrate image, video, text, and sound data with content-based data retrieval methods using different techniques [6]. What the number of content-based image retrieval systems currently available do in common is that, for a given single query image or a feature vector representation of an image, they search for the most similar images from a set of images. Then, the user browses on the returned images to choose those that looks like best. This can be associated to a content-based selection operation in a multimedia database systems. Other more complex operations such as "similarity-based join operation" (i.e., a Similarity-based matching of two or more image tables) are not supported in the works done in this area. However, we believe that there is the need to develop a formal framework for the similarity-based operations on image databases identical to that of the traditional database operations in such a way that this formalism serves image data management and query optimization techniques. Hence, the purpose of this paper is to introduce the most needed similarity-based operations, to study their properties, to formalize the use of these operations, and to use these as a basis for a similarity-based query optimization for image database systems.

We demonstrate the importance of our work with examples. Let an image table consist of the attribute components: image, the feature vector representation of the image, a component that contains alphanumeric information about the image.

Let EMP and SI be two image tables. EMP is a

---

1In this paper, the terms "content-based" and "similarity-based" are used interchangeably.

2That is, a vector representation of an image in terms of its features such as color, texture, shape, size, etc.
table of employee of a company and contains the attribute components: the photo of the employee, the feature vector representation of the photo, the name of the employee, his/her occupation and address. Let SI contain images of individuals who appeared in front of a gate of the company where a surveillance camera is mounted. Let the components of SI be: the images taken by the camera, their corresponding feature vector representations, the date and time at which each of the images were scanned or taken.

Suppose now that there is an investigation scenario of an event that is associated to the images in SI. SI alone can not give a complete information about a person. It is therefore necessary, to perform some operations on the tables in order to get a complete information of a person whose image is in SI. It is important to note that operations based on similarity measures do not perform accurate matchings. Thus, a common practice is to search for a range of the most similar images and then browse among the selected images. The following statement illustrates a possible query.

1) For pictures of individuals in SI that were scanned yesterday from 4 to 6PM, find their most similar images from EMP, with their corresponding name and address.

Processing this query requires a relational selection on the SI table and then a "similarity-based join" on SI and EMP. So we need to be able to combine and process relational and similarity-based operations on image tables.

Suppose now that the surveillance system is organized in such a way that there is one camera mounted at the external gate of the company’s compound and another camera at an internal gate of its particular department. Let SIE be the image table at the external gate of the company's compound and SII be the image table at the internal gate. The following is a possible query for a particular investigation purpose.

2) Get the names of employees who entered the gate of the department within two hours they entered the main gate of the compound on a particular date.

This query requires among others a similarity-based join on the three image tables SIE, SII, and EMP.

The above two examples demonstrate the need: to introduce novel operators such as the similarity-based join, to create a mechanism to be able to use the traditional operators in combination with the similarity-based operators, and to introduce a data repository model that enables us to perform these operations.

The remaining part of the paper is organized as follows. In Section 2 the related work is summarized. Section 3 presents our model for an image data repository. Section 4 introduces the novel similarity-based operators and studies their properties. Section 5 presents some important concepts for a similarity-based query optimization. Finally, conclusions are given.

2. STATE OF THE ART

In the domain of visual pattern recognition, a lot of work has been done to represent visual data such as image and video by its low-level features of color, texture, shape, etc. These low-level features can, therefore, be used for the purpose of recognition and content-based retrieval of multimedia data. In this regard, a number of content-based image retrieval research prototypes and commercial systems such as Photobook, Netra, Surfimage, VisualSeek, CAFIIR, STAR etc. [4, 7, 8, 9] are currently in practice. Commercial database systems have also started to integrate content-based retrieval modules into their systems to support multimedia data. QBIC is a content-based image query system available commercially either in standalone form, or as part of other IBM products such as the DB2 Digital Library. It offers retrieval by any combination of color, texture or shape - as well as by text keyword [7]. However, it doesn’t support operations such as the "similarity-based join". The VIR Image Engine from Virage is a well-known commercial system. It is available as a series of independent modules, that system developers can integrate into their own programs. The engine provides the fundamental capability for analyzing and comparing images. But, it has no concept of persistent storage, user interfaces, query processing nor optimization [9]. It is a pure set of algorithms that is designed to be embedded in software and hardware systems where its capabilities are needed. This makes it easy to extend it by building new types of query interface, or additional customized modules to process specialized collections of images. As a result, the VIR Image Engine is available as an add-on to the existing database management systems such as Oracle8i Enterprise Edition [6]. The Excalibur Image DataBlade Module from Excalibur Technologies is a product that offers a variety of image indexing and matching techniques based on the company’s own proprietary pattern recognition technology. This data-blade module is incorporated in the Informix database system to support a content-based image storage and retrieval [10].

A common feature of the add-on modules integrated in these DBMSs is that, given a query image, they search its most similar images from a database of images using their respective content-based image retrieval engines.

That is, the attempts so far didn’t exceed from these one-to-many content-based image retrieval operations. A positive result of these works is that, one can use SQL based statements to store, and retrieve images using content-based feature representations. However, as said before, these systems are strongly limited in terms of supporting complex similarity-based operations and
mixed queries involving both relational and similarity-based operations.
Query optimization is the process of choosing the optimal processing strategy to answer a query. Relational query optimization is still a hot research issue after more than twenty years of research and experience in the field [11, 12]. When it comes to similarity-based multimedia database systems, query optimization is a fresh research area. Indeed, most of the currently existing similarity-based retrieval systems focus mainly on their content-based retrieval engine input/output capabilities and gave less emphasis to content-based query optimization. Actually, one of the key issues, as stated by S. Adali et al. [13] is that, there has been no work on developing an algebra for similarity-based database query operations. However, the "multi-similarity algebra" that Adali et al. propose remains at a higher abstraction level. That is, it does not address the definition of an "operational" multimedia algebra usable for modeling, optimizing and processing multimedia queries combining similarity-based and relational operators at the implementation level. In this paper, we will show that the system of similarity-based operators we introduced, will create an important similarity-based algebraic space with which we can formulate basic rules and introduce a system of similarity-based query optimization.

3. IMAGE DATA REPOSITORY MODEL

M. Stonbraker et al. have broadly elaborated the power of an Object Relational Database systems in saying that the next wave of database management systems is under the Object Relational (OR) paradigm [12]. The OR system can as well convenient support multimedia data management. So, we keep our discussion within this framework.

The introduction of images in to a database management system has required different techniques to store, describe, and manipulate image data. W.I. Grosky et al. have given a model that describes the information that can be captured by an image data to facilitate its storage and content-based retrieval [14]. According to this model, the information that an image data can posses may be seen as physical view (image matrix and image header) and logical view (global and content-based view). Thus, an image repository or an image table should manage to capture all these information regarding an image.

In this paper we introduce our novel image data repository model (which we also refer it as "image table"), that is defined under an OR paradigm.

Definition 3.1. (Image Table)
An object-relational image table is defined as a table of five components \(M(id, o, f, a, p)\). Where:

- \(id\) is a unique identifier of an instance of \(M\),
- \(o\) is a reference to the image object itself which can be stored as a BLOB internally in the table or which can be referenced as an external BFILE,
- \(f\) is a feature vector representation of object \(o\).
- \(a\) is an attribute component that is used to describe the object using key-word like annotations and may be declared as a set of object types,
- \(p\) is a data structure that is used to capture pointer links to instances of other tables associated by a binary operation.

Note that, in an image table \(M\), the item of primary importance for a content-based retrieval is the image itself. The image is described by its feature vector and all the remaining attributes are associated to the image. The three components \(o, f, a\) can be used to capture sufficient information on an image data. \(p\) can be considered as a column whose content is a data structure that can store links to instances of other tables during binary operations such as similarity-based join. \(p\) has a value null in the base tables, but a non-null value in the resulting table of a similarity-based binary operation. More formally, when it contains a value, \(p\) is expressed as a set of tuples of the form \((table, set.of.jids)\), where the component \(table\) denotes the associated table by a binary operation and \(set.of.jids\) is the set of its referenced \(id\) elements. After a similarity-based binary operation like a similarity-based join of \(M_1\) and \(M_2\), the \(p\) component of the resulting image table, \(M'\), holds a link to a table \(M_2\). Then, for each instance of \(M'\), \(p\) will contain elements of the form \((M_2, \{id_2^1, ..., id_2^h\})\), where \(h\) is the number of instances of \(M_2\) associated by the operation.

With the help of this image table model, we can manage the requirements to integrate a content-based image data in an OR data management system more conveniently. Furthermore, this model can be extended to support the storage of segmented objects of an image to enable us perform salient-based operations on image tables. Our model for an object-relational image table: \(\text{M}(id, o, f, a, p)\), is designed in such a way that it can as well support the storage of segmented image data objects. Thus, operations on salient objects could be made possible [15].

4. SIMILARITY-BASED DATABASE OPERATIONS

For the definition of the novel similarity-based algebraic operators we use the method of content-based range query [16]. A content-based range query on a set of images \(S\) returns those image objects that are
within distance\(^3\) \(\varepsilon\) from the query image \(q\). However, our methodology applies to a Nearest Neighbor Search too, i.e. return the \(k\)-nearest objects to the query image \(q\).

In order to define similarity-based selection and join operations, we have first to define formally a range query.

**Definition 4.1. (Range Query)**

Given a set of images \(S\), a query image \(q\), and a positive real number \(\varepsilon\); the range query is defined as:

\[
NN^\varepsilon(S,q) = X \iff X \subseteq S \land \forall x \in X \bullet \|x - q\| \leq \varepsilon.
\]

**The Similarity-Based Selection Operator:**

The similarity-based selection operator is a unary operator on an image table performed on the component \(f\) based on the following definition.

**Definition 4.2. (Similarity-Based Selection)**

Given an image query object \(o\) with its feature vector representation, an image table \(M\), and a positive real number \(\varepsilon\). A similarity-based selection operation, denoted by \(\sigma_\varepsilon(M)\), is defined formally as:

\[
\sigma_\varepsilon(M) = X \iff X \subseteq M \land \forall (id_x, x, fv_x, a_x, p_x) \in X \bullet x \in NN^\varepsilon(M, o).
\]

The similarity-based selection operation first uses the range query search method to select the image objects that are most similar to \(o\) from the objects in \(M\). Then, it identifies the instances of \(M\) whose image object components are selected to be similar to \(o\).

**The Similarity-Based Join Operator on Image Tables:**

A similarity-based join is a binary operator on image tables performed on the feature vector components as defined formally below.

**Definition 4.3. (Similarity-Based Join)**

Let \(M_1\) and \(M_2\) be two image tables and let \(\varepsilon\) be a positive real number. The similarity-based join operation on \(M_1\) and \(M_2\), denoted by \(M_1 \otimes^\varepsilon M_2\), is an image table and defined as:

\[
M_1 \otimes^\varepsilon M_2 = X \iff \prod_{(id, o, f, a)}(X) = \prod_{(id, o, f, a)}(M_1) \land \forall (id_x, x, fv_x, a_x, p_x) \in X \bullet (id_y, y, fv_y, a_y, p_y) \in M_1 \bullet p_x = p_y \cup (M_2, s_{id}^f(M_2, y)),
\]

where \(s_{id}^f(M_2, y) = \prod_{M_2, id}(\sigma_\varepsilon(M_2))\) (i.e. the projection on the \(id\) component of the associated instances of \(M_2\)).

The similarity-based operators defined above depend on "relative" measures. Consequently, the similarity-based operators possess different algebraic properties than that of the relational ones. For example, we observe that the similarity-based join operator \(\otimes^\varepsilon\) is not commutative. This shows that, in similarity-based join operation, the order of the image tables is meaningful. If an image table appears at the left of a similarity-based join, then its objects are taken as reference for the similarity operation. In Query 1 of

Section 1, the image table SI should appear at the left since we are interested to know about the individuals who entered the gate. Contrarily, the relational join operator is commutative. However, an operator without this property is difficult to be exploited for query optimization. We thus need to see possibilities of extending this operator so that it satisfies the symmetry properties.

**4.1. A Symmetric Similarity-Based Join**

In view of the current practices of content-based query systems (where for a given image query object, we search for its most similar objects from a database of image objects), the similarity-based join defined above is what may be needed for many applications. However, its properties makes it lose the good properties that the relational join operator has. To make the similarity-based join operator suitable for similarity-based query optimization, we extend the multimedia join operator to a Symmetric Similarity-Based Join. To facilitate this, let us first define a basic operator called, the Additive Union.

**Definition 4.4. (Additive Union)**

Let \(M_1\) and \(M_2\) be two image tables, the Additive Union of \(M_1\) and \(M_2\) denoted by \(M_1 \oplus M_2\) is an image table that contains all the instances that are either in \(M_1\) or in \(M_2\), without excluding none of the instances of \(M_1\) or \(M_2\).

To perform a similarity-based binary operation on two image tables, we assume that their feature vector component \(f\) have identical structures that permits the computation of range query. Moreover, when we perform an additive union, if the \(a\) components do not have the same structure we take the union of the the \(a\) components of the two operand image tables. Here it is important to note that the additive union is commutative. Below we define a symmetric similarity-based join that makes use of the similarity-based join operation and the additive union operator.

**Definition 4.5. (Symmetric Similarity-Based Join)**

Let \(M_1\) and \(M_2\) be two image tables, the symmetric similarity-based join of \(M_1\) and \(M_2\) denoted by \(M_1 \otimes^\varepsilon M_2\) is formally defined as:

\[
M_1 \otimes^\varepsilon M_2 = (M_1 \oplus M_2) \cup (M_2 \oplus M_1).
\]

Hence, a symmetric similarity-based join possesses the following property.

**Property 1:** The symmetric similarity-based join operator \(\otimes^\varepsilon\) is commutative, i.e. \(M_1 \otimes^\varepsilon M_2 = M_2 \otimes^\varepsilon M_1\).

This follows directly from the commutativity of the additive union \(M_1 \cup M_2\). Figure 1 illustrates the way the symmetric similarity-based join of two image tables is computed.

We can now generalize the symmetric similarity-based join on more than two image tables and define a Sym-
The symmetric similarity-based join denoted by $M_1 \otimes^ε M_2$, based query optimization. The symmetric similarity-based join can be interpreted as the additive union of the two one-sided joins (non-symmetric similarity-based joins). That is, $M_1 \otimes^ε M_2 = (M_1 \otimes^ε M_2) \uplus (M_2 \otimes^ε M_1)$. After having the resulting table for one of the one-sided joins, we present here the method to get the other without applying similarity-based operations. If for example we have the resulting table of $M_1 \otimes^ε M_2$ with its new $p_1$ component, we can make use of the contents of $p_1$ to get the instances of $M_2 \otimes^ε M_1$. This follows from the symmetric property of distance. That is, given the same value of $ε$, if an object $o_2$ of $M_2$ is within a distance of $ε$ to the object $o_1$ of $M_1$, then the converse is also true. Hence, we can compute the symmetric multimedia join, $M_1 \otimes^ε M_2$, by performing only one of the one-sided joins and then using an operator called the Mine operator for the other (See below for the definition). The major advantage of this approach is that, the Mine operator could be much less expensive than similarity matching.

Let us first define and demonstrate the use of the Mine operator on a simple multimedia join, $M_1 \otimes^ε M_2$. Then, we will show how it can be generalized for any complex multimedia join expression.

**Definition 4.8.(The Mine Operator)**
Consider the multimedia join $M_1 \otimes^ε M_2$, then $\text{Mine}(M_1 \otimes^ε M_2) = M_2 \otimes^ε M_1$.

The Mine operator on $M_1 \otimes^ε M_2$ may be efficiently realized by using the component $p_1$ of $M_1$ in the resulting table of $M_1 \otimes^ε M_2$ for building the table $M_2 \otimes^ε M_1$. Conversely, $\text{Mine}(M_2 \otimes^ε M_1)$ uses the component $p_2$ of $M_2$ for building the table $M_1 \otimes^ε M_2$.

We stated above that the symmetric similarity-based join can be interpreted as the additive union of the two one sided joins (non-symmetric similarity-based joins). That is, $M_1 \otimes^ε M_2 = (M_1 \otimes^ε M_2) \uplus (M_2 \otimes^ε M_1)$. After having the resulting table for one of the one-sided joins, we present here the method to get the other without applying similarity-based operations. If for example we have the resulting table of $M_1 \otimes^ε M_2$ with its new $p_1$ component, we can make use of the contents of $p_1$ to get the instances of $M_2 \otimes^ε M_1$. This follows from the symmetric property of distance. That is, given the same value of $ε$, if an object $o_2$ of $M_2$ is within a distance of $ε$ to the object $o_1$ of $M_1$, then the converse is also true. Hence, we can compute the symmetric multimedia join, $M_1 \otimes^ε M_2$, by performing only one of the one-sided joins and then using an operator called the Mine operator for the other (See below for the definition). The major advantage of this approach is that, the Mine operator could be much less expensive than similarity matching.

Let us first define and demonstrate the use of the Mine operator on a simple multimedia join, $M_1 \otimes^ε M_2$. Then, we will show how it can be generalized for any complex multimedia join expression.

**Definition 4.8.(The Mine Operator)**
Consider the multimedia join $M_1 \otimes^ε M_2$, then $\text{Mine}(M_1 \otimes^ε M_2) = M_2 \otimes^ε M_1$.

The Mine operator on $M_1 \otimes^ε M_2$ may be efficiently realized by using the component $p_1$ of $M_1$ in the resulting table of $M_1 \otimes^ε M_2$ for building the table $M_2 \otimes^ε M_1$. Conversely, $\text{Mine}(M_2 \otimes^ε M_1)$ uses the component $p_2$ of $M_2$ for building the table $M_1 \otimes^ε M_2$.

We stated above that the symmetric similarity-based join can be interpreted as the additive union of the two one sided joins (non-symmetric similarity-based joins). That is, $M_1 \otimes^ε M_2 = (M_1 \otimes^ε M_2) \uplus (M_2 \otimes^ε M_1)$. After having the resulting table for one of the one-sided joins, we present here the method to get the other without applying similarity-based operations. If for example we have the resulting table of $M_1 \otimes^ε M_2$ with its new $p_1$ component, we can make use of the contents of $p_1$ to get the instances of $M_2 \otimes^ε M_1$. This follows from the symmetric property of distance. That is, given the same value of $ε$, if an object $o_2$ of $M_2$ is within a distance of $ε$ to the object $o_1$ of $M_1$, then the converse is also true. Hence, we can compute the symmetric multimedia join, $M_1 \otimes^ε M_2$, by performing only one of the one-sided joins and then using an operator called the Mine operator for the other (See below for the definition). The major advantage of this approach is that, the Mine operator could be much less expensive than similarity matching.

Let us first define and demonstrate the use of the Mine operator on a simple multimedia join, $M_1 \otimes^ε M_2$. Then, we will show how it can be generalized for any complex multimedia join expression.

**Definition 4.8.(The Mine Operator)**
Consider the multimedia join $M_1 \otimes^ε M_2$, then $\text{Mine}(M_1 \otimes^ε M_2) = M_2 \otimes^ε M_1$.

The Mine operator on $M_1 \otimes^ε M_2$ may be efficiently realized by using the component $p_1$ of $M_1$ in the resulting table of $M_1 \otimes^ε M_2$ for building the table $M_2 \otimes^ε M_1$. Conversely, $\text{Mine}(M_2 \otimes^ε M_1)$ uses the component $p_2$ of $M_2$ for building the table $M_1 \otimes^ε M_2$.
of \( M_2 \) in the resulting table of \( M_2 \otimes^s M_1 \) and builds the table \( M_1 \otimes^s M_2 \).

Let \( M_1(id_1, o_1, f_1, a_1, p_1) \) and \( M_2(id_2, o_2, f_2, a_2, p_2) \) be two image tables. Suppose we want to compute \( M_1 \otimes^s M_2 \). Let us first consider that we are computing \( M_1 \otimes^s M_2 \).

Figure 3 shows the projection of the image data points of \( M_1 \) and \( M_2 \) on a 2D plane for illustration purpose. As shown on the figure, for the image object \( o_1' \) of \( M_1 \) and a given value of \( \varepsilon \), the content-based range query looks for all objects \( o_2 \) of \( M_2 \) that are within a distance of \( \varepsilon \). Thus, it selects the image data points: \( o_2^1, o_2^2, o_2^3, \) and \( o_2^4 \) of \( M_2 \). Then, the multimedia join \( M_1 \otimes^s M_2 \) identifies the id’s of the instances of \( M_2 \) containing these objects are stored in the corresponding \( p \) component of \( M_1 \) in the resulting table. From this example we clearly see that, when we latter on want to compute \( M_2 \otimes^s M_1 \), \( o_1' \) of \( M_1 \) will be returned as an image object within a range of \( \varepsilon \) for all objects of \( o_2^1, o_2^2, o_2^3, \) and \( o_2^4 \) of \( M_2 \), by the symmetric property of distance. Since this information is already contained in the \( p \) component of the instance of \( M_1 \otimes^s M_2 \) that contains the object \( o_1' \), we can get this information from \( M_1 \otimes^s M_2 \). This can be repeated on all \( p \) components of \( M_1 \otimes^s M_2 \) to get all the instances of \( M_2 \otimes^s M_1 \).

\[ \text{Create table } T = T(id_i, o_i, f_i, a_i, p_i) \]

\[ \text{Foreach instance } \text{inst} \text{ of } M' \text{ Do} \]

\[ \quad \text{Foreach element } id_2 \text{ of } \text{inst}.p_1' \text{ Do} \]

\[ \quad \quad \text{If } id_2 \text{ is not in } T \]

\[ \quad \quad \quad \text{Append } \text{get\_instance}(M_2, id_2) \text{ in } T \]

\[ \quad \quad \text{End If} \]

\[ \quad \text{Update } p_i \text{ of } T.id_2 \text{ with } M'.id_1 \]

\[ \text{End Do} \]

\[ \text{End Do} \]

\[ \text{Return(T). } /* T = M_2 \otimes^s M_1 */ \]

Figure 4: Algorithm for the \( \text{Mine} \) operator.

A function called \( \text{get\_instance}(M, query\_id) = (query\_id, o, f, a, p) \), where \( (query\_id, o, f, a, p) \in M \) allows us to retrieve the instances of an image table \( M \) by their unique identifier components. Actually, the \( \text{get\_instance} \) function is a relational selection operation on the \( id \) component of \( M \).

It is up to the query optimizer to choose which multimedia join to do first and then use the \( \text{Mine} \) operator for the other. Here, we clearly see that the similarity-based join can be more effectively processed with query optimization techniques. Even for the non-symmetric join, if we actually need is to compute \( M_1 \otimes^s M_2 \), it may be less costly to first perform \( M_2 \otimes^s M_1 \) and then use the \( \text{Mine} \) operator to get \( M_1 \otimes^s M_2 \).

The \( \text{Mine} \) operator given above for a simple similarity-based join (i.e. a multimedia join between two image tables) can be generalized to apply on complex multimedia join expressions. The general principle is that, for each computed one-sided similarity-based join, we can compute the other sided join with the use of the \( \text{Mine} \) operator.

Consider the symmetric multimedia join of three image tables \( M_1 \otimes^s M_2 \otimes^s M_3 \). To compute this, the system needs to process six similarity-based joins (non-symmetric) based on Definition 4.5. However, we can only compute three of the similarity joins and the rest three can be generated using the \( \text{Mine} \) operator. In
like manner, to compute a symmetric multi-similarity-based join with \( n \) operand tables, we need to compute \( n(n-1) \) non-symmetric multimedia joins out of which \( n(n-1)/2 \) of them can be generated using the \textit{Mine} operator. And then, the results can be merged to form the symmetric multi-similarity-based join based on Definition 4.5.

Inversely to the \textit{Mine} operator, we propose the \texttt{Extract}_{M_1}(M_1 \oplus^e M_2)\) operator to extract the instances of \( M_1 \) from \( M_1 \oplus^e M_2 \) keeping the modified \( p \) component of \( M_1 \). It is a useful operator to obtain a one-sided multimedia join from a symmetric one. Hence, \texttt{Extract}_{M_1}(M_1 \oplus^e M_2) = M_1 \oplus^e M_2 \) and \texttt{Extract}_{M_1}(M_1 \oplus^e M_2) = M_2 \oplus^e M_1 \).

Similarly, the \textit{Extract} operator may be applied on symmetric multi-similarity-based joins. For example: \texttt{Extract}_{M_2}(M_1 \oplus^e M_2 \oplus^e M_3) = (M_2 \oplus^e M_1) \uplus (M_2 \oplus^e M_3)\).

### 5. SIMILARITY-BASED QUERY OPTIMIZATION

A query in a multimedia database system can be quite different from a query in the relational database systems. Besides the fact that, queries can contain multimedia object(s) as an input given by the user, the results of multimedia queries are not based on perfect matches but on degrees of similarity. Moreover, considering the fact that most feature extraction and content-based retrieval algorithms are very expensive, the need for query optimization of multimedia databases is very critical.

#### 5.1. Algebraic Rules on the Similarity-Based Operations

To develop query optimization strategies for content-based multimedia database systems, a well defined algebra for the similarity-based operations is necessary. Our work to introduce the necessary novel similarity-based operators and to study their properties for a possible similarity-based query operation can be found in [15] and in this paper targets to formalize the similarity-based algebraic space. Thus, the properties of these similarity-based algebraic operators with the properties of the already existing relevant relational operators can be utilized to formulate query optimization rules for content-based multimedia databases. Such algebraic rules play an important role to query optimization in addition to indexing schemes. Below are the major rules that can be applied for query optimization purpose.

**The Properties of Pushing Selection:** We separately consider two types of selection operators. A relational selection, for example on the \( a \) component, and a similarity-based selection on the \( f \) component of an image table. Let us denote the relational selection by \( \sigma_{M,a} \) and the similarity-based selection by \( \sigma_{M,f} \).

The following properties hold true:

1. \( \sigma_{M,f}(M_1 \uplus^e M_2) = \sigma_{M,f}(M_1) \uplus^e M_2 \): pushing a similarity-based selection into an additive union,

2. \( \sigma_{M,f}(M_1 \oplus^e M_2) = \sigma_{M,f}(M_1) \oplus^e M_2 \): pushing a similarity-based selection into a similarity-based join,

3. \( \sigma_{M,f}(M_1 \oplus^e M_2) = \sigma_{M,f}(M_1) \oplus^e M_2 \): pushing a similarity-based selection into an additive union.

These properties hold true also for the relational selection operator, \( \sigma_{M,a} \).

**Transformation Rules on Similarity-Based Join:**

The properties of the similarity-based binary operators defined and discussed in Section 4 can be used to formulate the following rules for query optimization.

1. The commutative property of \( \oplus^e \).

2. The exchange property of \( \oplus^e \).

3. The transformation property to exchange order using \textit{Mine}. (i.e. \( M_1 \oplus^e M_2 \rightarrow \textit{Mine}(M_2 \oplus^e M_1) \)).

4. Join method choice:

    \( M_1 \oplus^e_{\text{method}_1} M_2 \rightarrow M_1 \oplus^e_{\text{method}_2} M_2 \); An optimum join algorithm that results to the same output table can be chosen.

The factors for the exchange of the order of the operand tables could be: the number of instances in the tables, the density of the objects in the feature space, the value of \( \varepsilon \), the amount of I/O and Memory requirement, etc. A query optimizer must therefore carefully compare the benefits of transforming an expression against the computing costs.

#### 5.2. Local Optimization

We refer a local optimization as the relative inner table choice in a non-symmetric similarity-based join. For instance, it could be less costly to compute \( M_2 \oplus^e M_1 \)
than $M_1 \otimes^e M_2$ or vice-versa. Based on Rule 3 above, the supplemental price for the exchange of order is the cost of the Mine operator. That is, instead of computing $M_1 \otimes^e M_2$, we compute first the $M_2 \otimes^e M_1$ and then apply the Mine operator on the result. It is the task of the query optimizer to decide which one-sided similarity-based join to perform first and then use the Mine operator for the other. The decision depends on the cost model for the similarity-based join. A cost model for a simple nested-loop similarity-based join implementation is presented in the Annex. This cost model bases on the previously developed cost models [17, 18] for content-based image retrieval. Based on this model, given that the feature vectors of both input tables are indexed by a data-partitioning multimedia index structure (like R, X, SS-trees), the cost of the join computes as:

$$cost(M_1 \otimes^e M_2) = \frac{N_1}{C_{eff}} (1 + N_2 \sum_{j=1}^{d'} \left( \frac{1}{2} \right)^j \cdot V_{d'-j})$$

with $N_1, N_2$ being the number of data points in $M_1$ and $M_2$ respectively, $C_{eff}$ being the effective capacity of a leaf partition, $d'$ be computed as $d' = \lceil \log_2(N_2/C_{eff}) \rceil$, and $V_j = \frac{\sqrt[2]{\pi}}{\Gamma(j+1)} \cdot \epsilon^j$. Finally, an uniform object distribution is considered in this model.

Obviously, the cost of similarity-based join depends linearly on the number of data points in $M_1$. However, the dependency in the number of data points in $M_2$ is far less obvious. This means, that compared to relational query optimization, no simple decision rule can be established which decides on the relative inner table choice to one similarity-based join. Therefore, we require to compute the cost of each alternative, separately and then decide which strategy to follow.

Consider for example the transformation rule: $M_1 \otimes^e M_2 \rightarrow \text{Mine}(M_2 \otimes^e M_1)$. This transformation rule is worthwhile only if $cost(M_1 \otimes^e M_2) > cost(M_2 \otimes^e M_1)$. Compared to the respective cost of computing the non-symmetric join operation, the cost of Mine could be negligible. Therefore, a test of the form: if $cost(M_1 \otimes^e M_2) > cost(M_2 \otimes^e M_1)$ then perform Mine($M_2 \otimes^e M_1$) instead of $M_1 \otimes^e M_2$; can mostly be applied.

6. CONCLUSION AND FUTURE WORK

The many successful research results in the domain of computer vision have made content-based image retrieval a promising approach. The integration of these facilities into DBMSs in order to efficiently support multimedia data management is what should follow next. In this view, we presented a schematic model for an image table based on an OR paradigm. We then have introduced several useful similarity-based and other relevant operators. The similarity-based algebraic operators we introduced, have rich semantics, and possess very useful properties. We further discussed useful properties of our operators where content-based query optimization can effectively be applied in a multimedia database. Necessary query optimization strategies are discussed and a cost model for a simple nested-loop implementation of a similarity-based join is developed.

Future work includes, the implementation of a similarity-based query optimizer, the extension of the similarity-based algebra to support operations on salient objects of images, and the implementation of these in to an OR data management system.

7. REFERENCES


The metric used is the number of required disk accesses for performing the similarity-based join. This is a common metric in use in related works [17, 18].

We assume that the objects of the two operand image tables $M_1$ and $M_2$ follow the same object distribution function over a normalized object space of $[0, 1)^d$ (d is the dimension of the feature vector) and that a DP-based index structure is used. The impact of a concrete distribution function is discussed after the general cost considerations. The notations used here are shown in the table below.

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Notations</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_1, M_2$</td>
<td>Operand image tables</td>
</tr>
<tr>
<td>$d$</td>
<td>Dimension of the feature vectors</td>
</tr>
<tr>
<td>$N_1, N_2$</td>
<td>Number of data points in $M_1$ and</td>
</tr>
<tr>
<td></td>
<td>$M_2$ respectively</td>
</tr>
<tr>
<td>$m_1, m_2$</td>
<td>Number of leaf nodes in the index</td>
</tr>
<tr>
<td></td>
<td>structure of $M_1$ and $M_2$ respectively</td>
</tr>
<tr>
<td>$o^k_1$</td>
<td>k-th object in the i-th leaf node Li of</td>
</tr>
<tr>
<td></td>
<td>$M_2$ i.e. $x^k_1 = (x^1_1, \ldots, x^d_1)(1 \leq k \leq d)$</td>
</tr>
<tr>
<td>$C_{eff}$</td>
<td>Average number of data points in one</td>
</tr>
<tr>
<td></td>
<td>leaf page of $M_2$</td>
</tr>
<tr>
<td>$V^k_j$</td>
<td>Hyper-volume of the hypersphere $S^k_j$</td>
</tr>
<tr>
<td></td>
<td>with radius $\varepsilon$ for a dimension $0 \leq j \leq d$</td>
</tr>
<tr>
<td>$DA(o^k_1)$</td>
<td>Expected disk accesses for the range-</td>
</tr>
<tr>
<td></td>
<td>search of an object $o^k_1$ in $M_2$</td>
</tr>
<tr>
<td>$DA_{total}$</td>
<td>Expected total number of disk accesses</td>
</tr>
</tbody>
</table>

The expected number of disk accesses for a similarity-based join is the sum of expected disk accesses for the range search of all objects $o^k_1$ of $M_1$ ($1 \leq k \leq N_1$) in $M_2$. The sum of disk accesses of one object $o^k_1$ of $M_1$ is the number of leaf partition nodes of $M_2$ that intersects the hypersphere $S^k_j$ containing the neighbors for $o^k_1$ within the distance of $\varepsilon$.

In order to compute the number of leaf partition nodes of $M_2$ that intersects $S^k_j$, the Minkowski Sum of $S^k_j$ and the leaf nodes is determined. The Minkowski Sum corresponds graphically to the volume of an area which results from moving the center of $S^k_j$ over the surface of the bounding box of one leaf node. Summing up the Minkowski Sum for each leaf node in $M_2$ results in the expected number of disk accesses $DA(o^k_1)$.

Annex: Cost Model

The common content-based image retrieval operations to date is of type: given an image, find its all similar images from a set of or a database of images. The complexity of content-based image representations has initiated the use of multidimensional indexing schemes [19]. A multidimensional index structure makes it possible to access only fewer index pages in order to increase the efficiency of content-based image retrieval. Existing index structures for high dimensional feature spaces can be classified into Data Partitioning (DP)-based and Space Partitioning (SP)-based structures [20].

Bases on a simple and straightforward nested-loop implementation of similarity-based join using the range-search, we develop a cost model that works with DP-based index structures and that uses a hierarchical index structure. The straightforward method for performing a similarity-based join is to directly apply the algorithm of a range query for each object of the left input table $M_1$ as a query object looking for its similar objects from the right input table $M_2$. 

![Figure 5: Minkowski Sum in a 2-dimensional space.](image-url)
For instance, consider figure 5 for the Minkowski Sum in a 2-dimensional space. One leaf node $L_i$ of $M_2$ is a rectangle with length vector $x_i^2 = (x_i^1, x_i^2)$. The query circle $S_{Li}^2$ with radius $\varepsilon$ is the region which includes the neighbors of the query object. The number of disk accesses for the query object is the sum of the probabilities for intersections with all leaf partitions. The probability that $S_{Li}^2$ intersects one leaf partition $L_i$ corresponds graphically (see figure 5) to an area which results from moving the circle $S_{Li}^2$ around the leaf partition $L_i$. The volume of this area, i.e. the probability that $S_{Li}^2$ intersects $L_i$, is:

$$\text{volume}(L_i) + \text{perimeter}(L_i) \cdot \varepsilon + V^2 \varepsilon^2.$$ 

Summing it up to the index structure with $m_2$ index nodes, we obtain the expected number of disk accesses $DA(o_i^2)$ for the range-search of an object $o_i^2$ which expresses in the 2-dimensional case as:

$$\sum_{i=1}^{m_2} \text{volume}(L_i) + \text{perimeter}(L_i) \cdot \varepsilon + V^2 \varepsilon^2 = \sum_{i=1}^{m_2} x_i^2 \cdot \pi \cdot \varepsilon + x_i^2 \cdot \varepsilon + \pi \cdot \varepsilon^2.$$ 

Expressing the last line in terms of hypersphere volumes $V_j^d$ of dimensions lower than $d$, we obtain a slightly modified formula that corresponds exactly to the Minkowski Sum in $d$ dimensions. We can now express the expected total disk accesses, $DA_{total}^1$, in terms of hypersphere volumes $V_j^d$ as:

$$DA_{total}^1 = \sum_{i=1}^{m_2} \sum_{j=1}^{d} \sum_{(y^1, \ldots, y^j) \in \text{PowerSet}(x_1^1, \ldots, x_d^d)} \text{volume}((y^1, \ldots, y^j)) \cdot V_j^{d-j}.$$ 

where,

(a) $V_j^d = \frac{\pi^{\frac{d-j}{2}}}{\Gamma(\frac{d-j+1}{2})} \cdot \varepsilon^j$, 
(b) $\text{volume}((y^1, \ldots, y^j)) = \prod_{k=1}^{j} y^k$.

In order to obtain a cost model for high-dimensional spaces, we have to consider the so-called boundary effects (i.e. the perimeter of the bounding box of one leaf node of $M_2$ is partially outside the data space $[0, 1]^d$ and does consequently not contribute to the probability that this leaf node intersects the query object [17]). The main idea to deal with this problem is to attribute significance to the dimensions. i.e. a dimension is significant if its part in the Minkowski sum contributes to the overall probability. It has been shown by Berchtold et al. in [17] that only the first $d'$ dimension are significant with $d' = \lceil \log_2(N_2/C_{eff}) \rceil$. Consequently, equation (1) may be rewritten as follows in order to suit to boundary effects.

$$DA(o_i^1) = \sum_{i=1}^{m_2} \sum_{j=1}^{d'} \sum_{(y^1, \ldots, y^j) \in \text{PowerSet}(x_1^1, \ldots, x_d^d)} \text{volume}((y^1, \ldots, y^j)) \cdot V_j^{d-j}.$$ 

We can now express the expected range accesses, $DA_{total}$. For each object $o_i^1$ of $M_1$ we compute the neighbors within distance $\varepsilon$ in $M_2$. We can obviously assume that for two objects located in the same leaf node of $M_1$, the leaf node is still in the cache. Therefore $DA_{total}$ computes as:

$$DA_{total}^1 = m_1 + \sum_{i=1}^{N_1} DA(o_i^1)$$ 

It is a common use in relational databases to estimate uniform object distribution, for instance, for attribute values in order to compute the join selectivity. We will adopt this principle here. However, we are aware that this gives only a rough estimation, as many applications do not follow an uniform distribution [17, 18]. Uniform object distribution means that the object points follow a random distribu-