Foundation of the DISIMA Image Query Languages

Vincent Oria
Department of Computer Science
New Jersey Institute of Technology
Vincent.Oria@njit.edu

M. Tamer Özsu
School of Computer Science
University of Waterloo
tozsu@db.uwaterloo.ca

Paul J. Iglinski
Department of Electrical & Computer Engineering
University of Alberta
iglinski@ee.ualberta.ca

Abstract. Because digital images are not meaningful by themselves, images are often coupled with some descriptive or qualitative data in an image database. These data also divided into syntactic (color, shape, and texture) and semantic (meaningful real word object or concept) features, necessitate novel querying techniques. Most image systems and prototypes have focussed on similarity searches based upon the syntactic features. In the DISIMA system, we proposed an object-oriented image data model that introduces two main types: *image* and *salient object*. We further defined operations on the images and the salient objects as new joins. This approach is necessary in order to envision a declarative query language for images. This paper summarizes the querying facilities implemented for the DISIMA system and gives their theoretical foundation: the data model and the complementary algebraic operations, the textual query language (MOQL) and its visual counterpart (Visual-MOQL) based on an image calculus. Both languages are declarative and allow the combination of semantic and similarity queries.

Keywords: multimedia databases, image databases, image content modelling, image content-based querying

1. Introduction

Multimedia data, especially images, have become ubiquitous. Computer users around the world are sharing that data on a daily basis. As is the case for all multimedia data, the digital image data does not convey any meaningful information about the image. Any interpretation of the image data needs to be modelled and stored separately. The term meta-data is commonly used to refer to the data, other than the "raw" multimedia data, that help describe or understand the multimedia data. For images, these data can be divided into syntactic features, which can denote the visual properties (e.g. color, texture and shape) and semantic features, which describe the content of the images in terms of real world objects or the semantic concepts they convey. Because of

their size and lack of inherent semantics, raw images are not directly queried. Instead, queries are based on the syntactic and semantic metadata. In the DISIMA System (a research prototype developed at the University of Alberta, Canada from 1995 to 2000) the content of an image is described by means of salient objects (regions of interest) organized in an hierarchical way within the object-oriented paradigm. Semantic queries in DISIMA are based on the salient objects and their properties.

Visual features provide an alternative for image database querying and have been extensively studied for similarity searches (Del Bimbo, 1999). The visual features are extracted and represented as points in a multi-dimensional vector space. Multi-dimensional access methods are used to support efficient image searches (Stehling et al., 2000), (Korn et al., 1996), (Faloutsos et al., 1994). As low-level features (color, shape and texture) do not intrinsically carry any semantics, heuristics are often used to try to infer semantics to make the search more accurate. Shapes are combined with spatial relationships in (Del Bimbo and Vicario, 1998); color, shape and texture are combined in (Bartolini et al., 2000); whereas (Leung and Ng, 1998) superimposes a fixed grid on the image and combines the color properties of image blocks with the relative positions of the blocks. Another approach in capturing more semantics is image classification based on the same low-level features prior to similarity searches. For example, (Szummer and Picard, 1998) classify images into indoor and outdoor scenes. In (Bartolini et al., 2001) relevance feedback is used to learn image similarities and semantics in order to improve the similarity searches.

In general, low-level feature approaches to image retrieval are restricted, as it is difficult to capture high level semantics with low-level features. For better results, image DBMSs should include a combination of syntactic feature-based searching and searches based on semantics. This means that the system has to support a rich model that captures both the semantic and the syntactic features of images, a powerful declarative query language that can express both exact match and similarity queries and specific indexes to speed up the query processing.

This paper presents the querying functionality of the DISIMA image database system and is an extended version of (Oria et al., 2001). Section 2 discusses the modelling of semantic and syntactic features of images in DISIMA. Section 3 describes predicates defined on images and salient objects, which support the new algebraic operators used in querying. Section 4 presents the algebraic operators and their use in the MOQL query language. It also defines the calculus based on the image and salient object predicates. The calculus is the foundation of

the visual query language VisualMOQL. Finally, Section 5 concludes the paper.

2. Modelling Images and Salient Objects

The DISIMA project involved an extensive study of the requirements of a next generation image DBMS (Oria et al., 1997). This Section starts with an overview of the model, then presents in detail the modelling of the salient objects and the images.

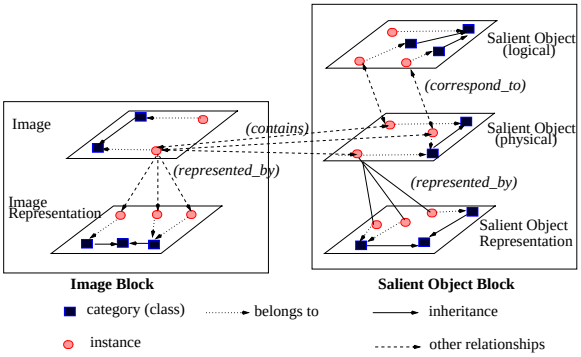

Figure 1. An Overview of the DISIMA Model

2.1. The DISIMA Model Overview

A data model is defined as a collection of mathematically well-defined concepts that express both static and dynamic properties of data intensive applications. The DISIMA model introduces two main concepts: *Image* and *Salient Objects* and operators to manipulate them.

A layered architecture was developed to define the image database model. The first layer is an object-oriented model that hosts the DISIMA model. The object-oriented model takes care of the classification of images and salient objects since the DISIMA model itself is composed of the image layer and the salient object layer (Figure 1). All the entities types in the DISIMA model are implemented as classes. The DISIMA type system provides a root type (or class) for each layer of the model: *Image, Image_Representation, LSO, PSO* and *PSO_ Representation*. By means of schema specifications, *Image* and *LSO* are subtyped by the application developer to define application-specific types. *PSO_ Representation* has two subclasses: *Raster_Representation*, which is similar to *Image _Representation* and *Vector_Representation*, which represents the geometry of physical salient objects since the properties of an object are implemented as behaviors, we use the two words interchangeably.

As shown in Figure 2, the two central entities in the DISIMA model are the images and the physical salient objects (PSOs). All the others

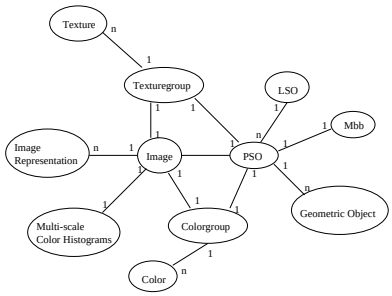

Figure 2. The DISIMA type system overview.

can be seen as properties of either images or physical salient objects. The multi-scale color histogram describes the color distribution in the image. In addition, dominant colors and textures in the image are selected and stored as features of physical salient objects. The MBB (Minimum Bounding Box) defines the minimum bounding box of the geometric object representing the physical salient object; the Geometric Object class defines its actual shape; the Texturegroup represents the set of textures found in the PSO; the Colorgroup gives a set of dominant colors in the PSO. Details on these objects are given below.

2.2. Modelling Salient Objects

A salient object is an object of interest in an image. Modelling image content by means of salient objects involves two main issues: (i) how to detect and recognize the salient objects and (ii) how to represent, index and query images through salient objects. The recognition of the physical salient objects is part of an active research area of Computer Vision, which is out of the scope of this paper. The results and challenges of this research are well summarized in (Jain et al., 2000): "In spite of almost fifty years of research, design of a general-purpose machine pattern recognizer remains an elusive goal....The best pattern recognizers in most instances are humans, yet we do not understand how humans recognize patterns". These are certainly the main reasons behind the fact that the process of image understanding is manual, or semi-automatic in the best cases. The work presented in this paper is more related to the second issue of image representation and querying. In the DISIMA project we first use a well-known edge detection technique, combined with other detection techniques, to draw contours of the isolated objects. The semantics of the detected objects are provided by manually linking them to a semantic object.

2.2.1. *Salient Objects: Definition*

A *salient object* is an object of interest in an image. In addition to generic semantic information, an object found in an image has some

specific properties linked to that image. To take into account this distinction, DISIMA introduces two notions of salient objects: *Logical Salient Object (LSO)*, which refers to a semantic object in an abstract sense and *Physical Salient Object*, which represents a semantic object as it appears in a particular image.

DEFINITION 1.

- A *physical salient object (PSO)* is a region of an image, that is, a geometric object (without any semantics) in a space (defined by an image) with properties like shape, color, and texture.
- A *logical salient object (LSO)* is the interpretation of a region. It is a meaningful object that is used to give semantics to a physical salient object.

2.2.2. *Modelling the Geometric Shapes of Salient Objects*
The object-oriented modelling of geometric objects potentially conflicts with the their mathematical definitions. The problem has already been addressed, but without a general solution that integrates code reuse at the data structure and the method level. We provided a more general solution to the shape hierarchy design issue (Oria et al., 1999a), following the model presented in (Leontiev et al., 1998).

Mathematically, a triangle and a rectangle are polygons, and a square is a special kind of rectangle. Accordingly, the class *Triangle* should be a subclass of the class *Polygon*. In the same way, the class *Square* should be a subclass of *Rectangle* which, in turn, should be defined as a subclass of *Polygon*. But from the point of view of data representation, a shape subclass might actually require fewer data members than its superclass, which leads to a conflict. For example, a polygon minimally requires a list of n consecutive points for its description, whereas a rectangle (n=4) can be defined by just three points and a square by just two points, if we take advantage of their symmetry.

The solution we proposed is based on the object model in (Leontiev et al., 1998) with a total separation between interface, implementation, and representation. We use the term *interface type* to refer to real-world entities and their programmatic interface. The term *implementation type* refers to the internal data representation. Interface and implementation types form two separate type hierarchies. *Class* or *concrete type* refers to the creation of instances and extent maintenance. Each of these types is then implemented by a C++ class. Each concrete type inherits from an interface type and an implementation type. We use three different kinds of C++ classes to simulate the notions of *interface type*, *concrete type* and *implementation type* defined in our

model. The interface types, whose names are not prefixed, declare the interface visible to the user; these types are usually abstract with pure virtual functions. An exception is the concrete interface type *Composite*. Some of the classes in this interface type hierarchy, e.g., *Atomic*, *1D*, and *2D*, are termed *abstract type*, since only their subtypes have a create method. The *I_*prefixed classes are the implementation types; they contain the actual data members. The *C_*prefixed classes represent concrete types; each of these concrete classes is publicly derived from its interface type and privately derived from its implementation type.

Figure 3 shows the final design of the geometric object class hierarchy. All the concrete types are associated with an implementation type and an interface type. Abstract interface types do not have any shallow-extent; they are not associated with any concrete type.

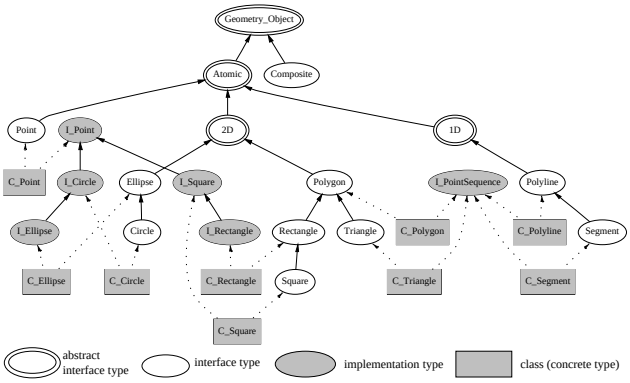

Figure 3. The Final Geometric Object Hierarchy

2.3. Modelling Images

The image object encapsulates all the properties of an image. This consists of a unique identifier, a set of representations, a set of general descriptors (including color, texture, and other syntactic features), and image content (physical salient objects contained in the image together with their associated logical salient objects). The distinction between logical and physical salient objects (referred to as semantic independence in (Oria et al., 1997))offers more flexibility and is introduced to handle both general and specific properties of salient objects.

2.3.1. *Image: Definition*

DEFINITION 2. Image Content: Let P be the domain of physical salient objects and L be the set of all logical salient objects. The *content of an image i* is defined by a pair $Cont(i) =< P_i, s >$ where:

- P_i is a set of physical salient objects of an image i ($P_i \subseteq P$)

- $s : P_i \longrightarrow L$ maps each physical salient object to a logical salient object.

Alternatively, the content of an image is defined as a set of ordered pairs $< pso, lso >$ where $pso \in P$ is a region of the image and $lso \in L$ is a logical salient object.

An image is then defined as follows:

DEFINITION 3. Image: An image i is defined by a quadruple $< id, Rep, Cont, Desc >$ where:
- id is the unique identifier of the image;
- Rep is a set of representations of the raw image in a format such as GIF, JPEG, etc;
- $Cont$ is the content of the image i as defined earlier;
- $Desc$ is a set of descriptive alpha-numeric data associated with i.

Color, texture and other syntactic features characterizing the whole image are part of the $Desc$.

2.3.2. *Image Color Representation*
Colors are one of the visual features that people immediately perceive when looking at an image. In addition to describing the main colors in each physical salient object, the colors in the image can be represented as a color histogram. An image color histogram captures the color distribution of the image pixels and can be defined as a discrete function $h(c_k) = n_k$, where c_k is the k-th color value and n_k is the number of pixels in the image with that color. In order to compare color histograms of images with different sizes, color histograms are often normalized as $H(c_k) = \frac{n_k}{n}$ where n is the total number of pixels in the image. For more accuracy in color similarity searches, a 64-color histogram is computed for the entire image, then the image is recursively decomposed into four quadrants with a color histogram computed for each quadrant. Hence, the image color properties are represented by a quadtree with a predetermined child order (Figure 4). The root stores the color histogram of the entire image while the nodes store the color histograms of their respective quadrants. The image color structure is referred to as a *multi-scale color histogram*. The *multi-scale color histogram* has 3 levels (1 to 3), as experimental results did not show any improvement beyond level 3.

3. Predicates on Images and Salient Objects

Predicates are used to define conditions in queries. They can be directly used in calculus-based queries to define formulas or in the definition of

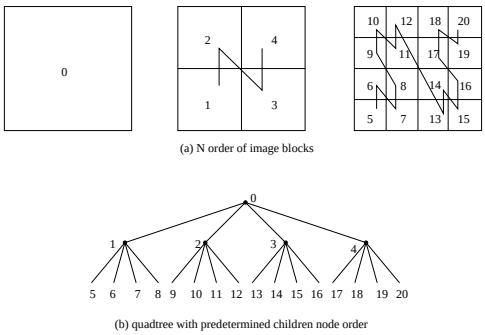

Figure 4. A quadtree stores color histograms of image blocks

algebraic operators. Since the classical predicates $\{=, <, \leq, >, \geq\}$ are not sufficient for images, we defined a new set of predicates to be used on images and salient objects.

3.1. Contains Predicate

The objects found in images are physical salient objects. So we defined a predicate (*contains*) to check if an image contains a given physical salient object. But since it is easier for users to define queries on semantics and other visual descriptors than on physical salient objects, we generalized the contains predicate to work on logical salient objects, colors, textures and shapes.

DEFINITION 4. The Contains Predicate:

- Contains predicate and physical salient objects: Let i be an image, p a physical salient object and *pso* a behavior that returns the set of physical salient objects contained in an image $contains(i, p) \Longleftrightarrow p \in i.pso$.
- Generalized Contains predicate: Let i be an image, o an object with a behavior *pso* that returns the set of physical salient objects associated with o: $contains(i, o) \Longleftrightarrow \exists p \in o.pso \ \wedge p \in i.pso$.

The object o can be a logical salient object, a color or a shape. Both $o.pso$ and $i.pso$ are sets of PSOs (an image can be composed of more than one physical salient object, and a logical salient object can be found in more than one image). An image contains a logical salient object if it is associated with a physical salient object that is contained in the image. The same scheme can be applied to objects of *MBB, Colorgroup, Texturegroup* and *Geometric_object* (Figure 2) as they are all

associated with physical salient objects through a *pso* behavior. In this case, queries like "find images with a red object" or "find images with an object of a rectangular shape" can easily be answered by combining a selection condition in the respective classes and a contains condition.

3.2. Shape Similarity Predicates

The geometric model stores any given physical salient object shape in the extent of its most specific class. Shapes can be compared by their types or their appearances, their boundaries or their pixel distributions.

DEFINITION 5. The Shape Type Similarity Predicate:
Let *type* be the behavior that returns the type of a shape. Two shapes s and t are type similar if $s.type = t.type$. In other words:
$shape\_type\_similar(s,t) \Longleftrightarrow s.type = t.type$.

The $shape\_type\_similar(s,t)$ predicate can also be used to check wether a given shape is of a certain type: $shape\_type\_similar(s, rectangle)$ is true if s is a rectangle.

When comparing shapes, there are two main approaches: region-based similarity and contour-based similarity (Bober, 2001). *Region-based similarity* checks the similarity in terms of the spatial distribution of pixels in the regions, whereas *contour-based similarity* concerns the contours of the objects. We chose the contour-based turning angle algorithm (Arkin et al., 1991) because of its orientation invariance. Basically the turning angle algorithm describes a shape from a starting point on the perimeter and, following a counterclockwise direction, it moves around the perimeter recording the angles it encounters. By plotting the cumulative angles on the horizontal axis of a 2-dimensional space and the distance traversed on the vertical axis, a step-like graph is obtained. In order to compare shapes of different sizes, the perimeter is normalized to 1. A distance d_{shape} is then defined on the plots. The turning angle algorithm, however, cannot be applied to circles and ellipses, for which we have developed some formulas(circle: $(x-h)^2+(y-k)^2=r^2$ where (h,k) is the center and r is the radius; ellipse: $\frac{(x-h)^2}{a^2}+\frac{(y-k)^2}{b^2}=1$, where (h,k) is the center, a is the major and b is the minor) that can be used for similarity comparisons.

DEFINITION 6. The Shape Similarity Predicate:
Given a shape similarity metric d_{shape} and a similarity threshold $epsilon_{shape}$, two shapes s and t are similar with respect to d_{shape} if $d_{shape}(s,t) \leq \epsilon_{shape}$. In other words:
$shape\_similar(s,t,\epsilon_{shape}) \Longleftrightarrow d_{shape}(s,t) \leq \epsilon_{shape}$.

Note that the *shape_similar* can be generalized to include *shape_type_similar*. For example, *shape_ similar*(s, t, ϵ_{shape}) can have the semantics of *shape_type_similar* if the parameter ϵ_{shape} is missing.

3.3. Color Similarity Metric and Predicates

Similarity searches are often based on some metrics. For color similarity in DISIMA, we defined two distance metrics: one on the multi-scale color histograms and another one on the image average colors. The second metric is used mainly for the 3DH index structure (Lin et al., 2001) to filter the images before applying the distance metric on multi-scale color histograms.

- Distance metric on multi-scale color histograms:
  In this work, we used the weighted Euclidean distance function defined as (Hafner et al., 1995):

  $$d^2_{hist}(X, Y) = Z^T A Z$$

  where $Z = (X - Y)$, Z^T denotes the transpose of Z, and $A = [a_{ij}]$ is a *similarity matrix* whose elements a_{ij} denote similarity between colors i and j. The distance between two multi-scale color histograms at a level l $(1 \leq l \leq 3)$ is defined as the average of the distances on respective quadrants: $d_l = \frac{1}{4^{l-1}} \sum_{i=1}^{4^{l-1}} (d_i)$ where d_i is the distance on the i^{th} color histogram.

- Similarity metric on average colors: The average color does not need to be explicitly stored, as it can be computed from the color histogram at the first level. Let $C = [c_1 c_2 \ldots c_{64}]$ be a 3×64 matrix representing the 64 colors used to define the color histograms with $c_i = [\alpha_i, \beta_i, \gamma_i]$ where α, β and γ are the magnitudes along the 3 color dimensions (R,G,B). Given two normalized n-dimensional color histograms at level 1, X and Y, the 3×1 average color vector for each is: $X_{ave} = CX$, $Y_{ave} = CY$. The squared average color distance is defined by $d^2_{ave} = (X_{ave} - Y_{ave})^T (X_{ave} - Y_{ave}) = (X - Y)^T C^T C (X - Y)$.

While the average color comparisons are not as accurate as one between full n-dimensional histograms, they are much faster. Moreover, the images retrieved by average color comparisons are guaranteed to include all images that should be retrieved by color histogram comparisons, as proven in (Hafner et al., 1995). Accordingly, for any range query of the form $d_{l1} \leq \epsilon$, $d_{ave} \leq f(\epsilon)$ (f depends on d_{l1} which is the distance at level 1) can be used to retrieve images quickly and without misses. The expensive measure d_{hist} will then have to be applied only to the filtered set of images (Lin et al., 2001).

DEFINITION 7. The Color Similarity Predicate:
Given 2 color representations (c_1, c_2) and a color distance metric d_{color}. The color representations c_1 and c_2 are similar with respect to d_{color} if $d_{color}(c_1, c_2) \leq \epsilon_{color}$. In other words: $color\_similar(c_1, c_2, \epsilon_{color}) \Longleftrightarrow d_{color}(c_1, c_2) \leq \epsilon_{color}$.

3.4. Spatial Predicates

The spatial model for DISIMA is based on Allen's temporal logic (Allen, 1983) that gives a temporal interval algebra for representing and reasoning about temporal relations between events represented as intervals. The elements of the algebra are sets of seven basic relations (before, meets, overlaps, during, starts, finishes and equal), which can hold between two intervals, and their inverse relations. The temporal interval algebra can be seen as topological or directional relations in one dimensional space. This Algebra can be enhanced by extension into multi-dimensional space. Based on that observation and by combining the interval relationships on the two axes defined on an image, we defined 12 directional relations classified into the following three categories: *strict directional relations* (north, south, west, and east), *mixed directional relations* (north-east, south-east, north-west, and south-west), and *positional relations* (above, below, left, and right). In addition, we defined 6 topological relationships (equal, inside, cover, inside touch and disjoint). The definitions of these relations in terms of Allen's temporal algebra can be found in (Li et al., 1996). Defining the spatial relations on the intervals obtained from projections of the objects onto the axes is equivalent to defining them on the minimum bounding boxes of the objects. Since some topological relations can be true on the minimum bounding boxes and false for the objects, a refinement phase is necessary to get rid of the false positives.

4. Querying Images

Two query languages MOQL (Li et al., 1997) and VisualMOQL (Oria et al., 1999b) were defined for DISIMA. MOQL is based on an algebra defined using the predicates presented above. VisualMOQL is based on an object calculus defined with the same predicates.

4.1. DISIMA Algebra and MOQL

The new operators are defined as joins: semantic join, similarity join and spatial join.

4.1.1. *Semantic Join*
DEFINITION 8. Semantic Join (physical salient objects):
Given a set of Images I and a set of physical salient objects P the *semantic join* between I and P, $I \bowtie_{contains} P$, defines the elements of $I \times P$ where $contains(i, p)$.

The semantics of the *contains* predicate was extended to work on objects with a *pso* behavior that returns the physical salient objects with which they are associated. These objects are of type *logical salient object (LSO), MBB, Colorgroup, Texturegroup* and *Geometric_object*. All these objects express properties of physical salient objects. They are more accessible to users than the PSO and can be used to find desired physical salient objects. Because of that, we refer to them as semantic objects.

DEFINITION 9. Generalized Semantic Join:
Let S be a set of semantic objects of the same type with a behavior *pso* that returns, for a semantic object, the physical salient objects it describes. The *semantic join* between an image class extent I and the semantic object class extent S, denoted by $I \bowtie_{contains} S$, defines the elements of $I \times S$ where for $i \in I$, and $s \in S$, $contains(i, s)$.

As defined, the semantic join can be useful in finding the images that contain some salient object as well as the salient objects contained in an image. It is used not only to query images that contain some logical salient objects but also to query images that contain a physical salient object of a specific shape (rectangle, polygon, etc.), color or texture. Examples of queries that can be expressed using the semantic join are "list the salient objects of a given image", "list the images that contain a person", "for each image, list the salient objects it contains", "list the images that contain an object of a rectangular shape".

Images and physical salient objects have properties on which some distance metrics are commonly defined. These properties are color, texture and geometric shape. The image properties can directly be used to find desired images while the metric properties on physical salient objects are used to select some physical salient objects before a semantic join.

4.1.2. *Similarity Joins*
Similarity joins are join operations that involve metric data types and operators. Examples include queries like "find all scenery images that are similar (with regard to color) to images with animals", "find all images of scenery that look like a given image", "find images that contain an object with a shape similar to a given shape", "find images that contain an object with a color similar to a given color".

DEFINITION 10. Similarity Join:
Given a similarity predicate *similar* and a threshold ϵ, the *similarity join* between two sets R and S of images or physical salient objects, denoted by $R \bowtie_{similar(r.i,s.j,\epsilon)} S$ for $r \in R$ and $s \in S$, is the set of elements from $R \times S$ where the behaviors i defined on the elements of R and j on the elements of S return some compatible metric data type T and $similar(r.i, s.j)$ (the behaviors i and j can be the behaviors that return color, texture or shape).

A shape in one class can be similar to shapes from another class. In the example given in Figure 5, the the polygonal shape (c) is close to the shape (a). From the geometric shape model (Figure 2), we defined three groups of shape: *Ellipse*, *Polyline* and *Polygon*. The *Geometric_Object* class supports three types of similarity metrics: *full-group, class*, and *sub-group* The *ellipse* group includes the *Ellipse* and *Circle* classes. The *polyline* group includes the *Polyline* and *Segment* classes. The *Polygon*, *Rectangle*, *Square*, and *Triangle* classes belong to the *polygon* group.

Given a shape similarity query, we must decide which extent to use (shallow or deep extent). Instead of asking for another parameter, the decision is made on the basis of the similarity threshold in the query and the presence or absence of a shape. With a shape as parameter, if the similarity threshold is set to 1, a class match is performed; otherwise, a full-group match is performed. When the parameter of a shape similarity query is a shape type rather than an instance of a shape type, a subclass match is performed.

4.1.3. *Spatial Join*

The shapes are defined on physical salient objects and a spatial join can be use to select them.

DEFINITION 11. Spatial Join:
The spatial join of the extent of two sets R and S, denoted by $R \bowtie_{r.i\theta s.j} S$, is the set of elements from $R \times S$ where the behaviors i defined on the elements of R and j on the elements of S return some spatial data type, θ is a binary spatial predicate, and $R.i$ stands in relation θ to $S.j$ (θ is a spatial operator like north, west north-east, intersect, etc.).

The above spatial join definition is adapted from the one given in (Günther, 1993) in the relational context. In DISIMA, the spatial join is used for spatial relationships between salient objects, PSOs of a given image, considered within a 2-dimensional space.

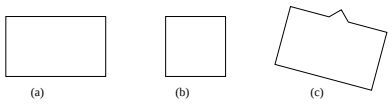

Figure 5. Shape similarity.

4.1.4. *DISIMA Algebra and MOQL*
As in relational and object-oriented database systems, the join operators introduced are implemented in the **where** clause of the query language. The DISIMA query language is MOQL (Multimedia Object Query Language), a declarative text-based multimedia query language (Li et al., 1997), which is an extension of the standard OQL language (Cattell et al., 1997). MOQL is a general multimedia query language but this work uses only the clauses related to images. The extensions introduced to OQL by MOQL are in the **where** clause, in the form of four new predicate expressions: *spatial_expression*, *temporal_expression*, *contains_predicate*, and *similarity_expression*. The *spatial_expression* is a spatial extension which includes spatial objects, spatial functions, and spatial predicates. The *contains_predicate* is defined as: *contains_predicate* ::= *image_object* **contains** *salientObject*. The **contains** predicate is used to express semantic join conditions and the **similarity** predicate expresses similarity

THEOREM 1. *The DISIMA Algebra is equivalent to MOQL.*

Proof (sketch): To prove this we need to prove that the DISIMA Algebra is complete with respect to MOQL and vice-versa. Let us denote by M_D, the DISIMA model, L_A the set of queries that can be expressed with the DISIMA algebraic language and by L_M the set of queries expressed in MOQL using the predicates introduced. Given *db*, a database in M_D and q a query expressed in MOQL, L_A is complete with respect to L_M is formally expressed as $\forall db \in M_D\ \forall q' \in L_A\ \exists q \in L_M\ q(db) = q'(db)$ and L_A is complete with respect L_M, means $\forall db \in M_D\ \forall q \in L_M\ \exists q' \in L_A\ q'(db) = q(db)$. This is always the case as the new clauses introduced in the definition of MOQL express conditions based on the predicates used in the definition of the DISIMA algebra.

4.2. Image Calculus and VisualMOQL

As indicated earlier, the DISIMA model is an object-oriented one. Consequently, in the image calculus that we have defined, atoms refer to objects or class extents. The new classes we introduced for the DISIMA model are Image, LSO (Logical Salient Object), PSO (physical Salient Object), Geometry_Object (shape), Color and Texture. In addition to the classic predicates (= [value equality], == [object identity],

$<, \leq, >, \geq$) the predicates we defined can be used. An image query can be expressed as: $\{m|m \in Image \Phi(m)\}$ where Φ is a formula without a free variable. A query like "Give me all the images containing a shape similar to a given shape r with similarity threshold (ϵ)" can be expressed as follows:
$\{m|m \in Image\ \exists o \in PSO, \exists s \in Geometry\_Object\ s == shape(o) \wedge shape\_similar(s, r, \epsilon) \wedge contains(m, s)\}$

This calculus represents the foundation of VisualMOQL (Oria et al., 1999b), the visual counterpart of MOQL. VisualMOQL provides an easier way to express queries by means of visual objects. It allows users to define complex queries by composing several subqueries. For example, a query Q "Find images with 2 people next to each other without any building, or images with buildings without people" can be decomposed into 2 sub-queries Q_1: "Find images with 2 people next to each other without any building" and Q_2: " Find images with buildings without people"[1]. The query $Q = Q_1 \vee Q_2$

In general the user defines subqueries that are combined in the query canvas to form a compound query. A subquery can be a negative formula of the form $\neg\Phi$ or a positive formula of the form Φ. The semantics of VisualMOQL is based on the object calculus as follows:

- Subquery $\Phi(x) = \{\exists i \in I\ \exists s_1 \in S_1 ... \exists s_k \in S_k | cond(i) \wedge cond(s_1 ... s_k) \wedge contains(i, s_1)\ \wedge ... \wedge\ contains(i, s_k) \wedge i == x\}$ where:
  - I is an image class (i.e., Image or a subclass of Image) called *the range of* Φ, $range(\Phi) = I$.
  - $cond(i)$ expresses some selective conditions on the image properties.
  - $cond(s1...sk)$ expresses some selective conditions on salient object properties (exact or similarity matches and and spatial relationships).
- Composite subquery: If Φ_1 and Φ_2 are subqueries then:
  - $\neg\Phi_1$ is a subquery; $range(\Phi_1) = range(\neg\Phi_1)$
  - $\Phi_1 \wedge \Phi_2$ is a subquery;
    $range(\Phi_1 \wedge \Phi_2) = common\_ancestor(range(\Phi_1), range(\Phi_2))$
  - $\Phi_1 \vee \Phi_2$ is a subquery;
    $range(\Phi_1 \vee \Phi_2) = common\_ancestor(range(\Phi_1), range(\Phi_2))$
  - $\Phi_1 - \Phi_2$ is a subquery;
    $range(\Phi_1 - \Phi_2) = common\_ancestor(range(\Phi_1), range(\Phi_2))$.

[1] Each of these sub-queries can further be decomposed into simpler sub-queries.

– A query is of the form: $Q = \{m|\Phi(m)\}$ where: Φ is a subquery.

The function *common_ancestor* returns the common ancestor in the type system. The DISIMA type system is rooted, so given two image classes, the common ancestor always exists. The query is normalized (negations are pushed in) in the translation. The subqueries are safe even with negations because they are range restricted.

THEOREM 2. *A query expressed in VisualMOQL can always be expressed in MOQL.*

Proof (sketch): The theorem expresses the completeness of MOQL with respect to VisualMOQL. Since we have already established the completeness between MOQL and the Algebra and VisualMOQL is based on the calculus, we need to establish that the DISIMA algebra is complete with respect to VisualMOQL. Here again the prove is based on the definitions of the common predicates.

The theorem is necessary to keep one query processor for both MOQL and VisualMOQL. A query expressed in VisualMOQL is translated into MOQL before being submitted to the processor.

5. Conclusion

In this paper, we have presented the foundation of the query languages developed for the DISIMA image database management system. The query languages are built on an object-oriented model that defines the types and the algebraic operators. The two main types introduced are the image type and the salient object types (logical and physical). The image type defines the content of an image as a set of physical salient objects that are regions of the image and the semantics of a physical salient object are given by a logical salient object. Each of the types introduced can be subtyped by an application developer to define the application schema. We have introduced an algebra and have shown that MOQL, the declarative textual query language, is based on the algebra. We have also shown that VisualMOQL is based on an image calculus and that a query expressed in VisualMOQL can always be expressed in MOQL. The whole system is implemented on top of the ObjectStore object-oriented database management system.

Although ObjectStore provides some querying facilities over collections, it does not have a built-in declarative query language. Therefore, we have fully implemented a MOQL query processor, based on an image algebra. The result of the MOQL parser is an internal query tree structure which is later transformed into an execution plan. The query engine uses the query tree directly to generate a non-necessarily optimized execution plan. This is just a proof of concept. A complete

query processor should integrate an algebraic query optimizer that will use some equivalence formulas based on the operators to rewrite the user query in a more optimal way and a physical query optimizer to select the optimal access methods for each operation. A challenge in multimedia query optimization is the integration of partially ordered subquery results as multimedia uses similarity searches.

Acknowledgements

The authors thank the following people who worked on different parts of the DISIMA system at the University of Alberta: John Zhong Li (Ph.D. 1998), Lin Irene Cheng (M.Sc. 1999), Shu Lin (M.Sc. 2000), Bing Xu (M.Sc.2000).

This research is supported by a strategic grant from the Natural Science and Engineering Research Council (NSERC) and the Institute for Robotics and Intelligent Systems (IRIS) of Canada. The work was conducted while the authors were affiliated with the University of Alberta.

References

Allen, J. F.: 1983, 'Maintaining Knowledge about Temporal Intervals'. *Communications of ACM* **26**(11), 832—843.

Arkin, E. M., L. P. Chew, D. P. Huttenlocher, K. Kedem, and J. S. B. Mitchell: 1991, 'An Efficiently Computable Metric for Comparing Polygonal Shapes'. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **13**(3).

Bartolini, I., P. Ciaccia, and M. Patella: 2000, 'A Sound Algorithm for Region-Based Image Retrieval Using an Index'. In: *Proceedings of the 4th International Workshop on Query Processing and Multimedia Issue in Distributed Systems (QPMIDS'00)*. Greenwich, London, UK, pp. 930–934.

Bartolini, I., P. Ciaccia, and F. Waas: 2001, 'FeedbackBypass: A New Approach to Interactive Similarity Query Processing'. In: *Proceedings of the 27th International Conference on Very Large Data Bases (VLDB 2001)*. Rome, Italy, pp. 201–210.

Bober, M.: 2001, 'Galileo: A strongly-typed, interactive conceptual language'. *IEEE Transactions on Circuits and Systems for Video Technology* **11**(6), 716—719.

Cattell, R. G. G., D. Barry, D. Bartels, M. Berler, J. Eastman, S. Gamerman, D. Jordan, A. Springer, H. Strickland, and D. Wade (eds.): 1997, *The Object Database Standard: ODMG 2.0*. Morgan Kaufmann.

Del Bimbo, A.: 1999, *Visual Information Retrieval*. Morgan Kaufmann Publishers.

Del Bimbo, A. and E. Vicario: 1998, 'Weighting spatial relationships in retrieval by visual contents'. In: *Proceedings of 4th IFIP 2.6 Working Conference on Visual Database Systems - VDB 4*. L'Aquila, Italy, pp. 277—292.

Faloutsos, C., W. Equitz, M. Flickner, W. Niblack, D. Petkovic, and R. Barber: 1994, 'Efficient and effective querying by image content'. *Journal of Intelligent Information Systems* **3**(3/4), 231—262.

Günther, O.: 1993, 'Efficient Computation of Spatial Joins'. In: *Proc. IEEE 9th Int. Conference on Data Engineering*.

Hafner, J., H. S. Sawhney, W. Equitz, M. Flickner, and W. Niblack: 1995, 'Efficient Color Histogram Indexing for Quadratic Form Distance Functions'. *IEEE Trans. on Pattern Analysis and Machine Intelligence* **17**(7), 729–736.

Jain, A. K., R. P. W. Duin, and J. Mao: 2000, 'Statistical Pattern Recognition: A Review'. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **22**(1), 4–37.

Korn, F., N. Sidiropoulos, C. Faloutsos, E. Siegel, and Z. Propopapas: 1996, 'Fast Nearest Neighbor Search in Medical Image'. In: *Proceedings of the 22nd International Conference on Very Large Databases, VLDB*. Bombay, India.

Leontiev, Y., M. T. Özsu, and D. Szafron: 1998, 'On Separation between Interface, Implementation and Representation in Object DBMSs'. In: *Proceedings of Technology of Object-Oriented Languages and Systems 26th International Conference (TOOLS USA98)*. Santa Barbara, pp. 155—167.

Leung, K. and R. T. Ng: 1998, 'Multiscale Similarity Matching for Subimage Queries of Arbitrary Size'. In: *Proc. 4th Working Conf. on Visual Database Systems*. L'Aquila, Italy, pp. 243–264.

Li, J. Z., M. T. Özsu, and D. Szafron: 1996, 'Spatial Reasoning Rules in Multimedia Management Systems'. In: *Proceedings of the International Conference on Multimedia Modeling MMM'96*. Toulouse, France.

Li, J. Z., M. T. Özsu, D. Szafron, and V. Oria: 1997, 'MOQL: A Multimedia Object Query Language'. In: *Proceedings of the 3rd International Workshop on Multimedia Information Systems*. Como, Italy, pp. 19—28.

Lin, S., M. T. Özsu, V. Oria, and R. Ng: 2001, 'An extendible hash for multi-precision similarity querying of image databases'. In: *Proceedings of the 27th VLDB conference*. Rome, Italy.

Oria, V., M. T. Özsu, and P. Iglinski: 2001, 'Querying Images in the DISIMA DBMS'. In: *7th International Workshop on Multimedia Information Systems (MIS)*. Capri, Italy, pp. 89—98.

Oria, V., M. T. Özsu, P. Iglinski, and Y. Leontiev: 1999a, 'Modeling Shapes in an Image Database System'. In: *Proceedings of the 5th International Workshop on Multimedia Information System*. Indian Wells, California, pp. 34—40.

Oria, V., M. T. Özsu, X. Li, L. Liu, J. Li, Y. Niu, and P. J. Iglinski: 1997, 'Modeling Images for Content-Based Queries: The DISIMA Approach'. In: *Proceedings of 2nd International Conference of Visual Information Systems*. San Diego, California, pp. 339—346.

Oria, V., M. T. Özsu, B. Xu, L. I. Cheng, and P. Iglinski: 1999b, 'VisualMOQL: The DISIMA Visual Query Language'. In: *Proceedings of the 6th IEEE International Conference on Multimedia Computing and Systems*, Vol. 1. Florence, Italy, pp. 536—542.

Stehling, R. O., M. A. Nascimento, and A. X. Falcao: 2000, 'On "Shapes" of Colors for Content-Based Image Retrieval'. In: *Proceedings of ACM Multimedia 2000 Workshops*. Los Angeles, California, pp. 171—174.

Szummer, M. and R. W. Picard: 1998, 'lndoor-outdoor image classification'. In: *Proceeding of the IEEE Conference on Computer Vision and Pattern Recognition - Whorkshop on Content-based Access of Image and Video Libraries*. Santa Barbara, California.