Querying Images in the DISIMA DBMS*

Vincent Oria†, M. Tamer Özsu‡and Paul J. Iglinski§

Abstract

Because digital images are not meaningful by themselves, images are often coupled with some descriptive or qualitative data in an image database. Moreover the division of these data into syntactic (color, shape, texture) and semantic (meaningful real word object or concept) features necessitates novel querying techniques. Most image systems and prototypes have focussed on similarity searches based only on the syntactic features. In the DISIMA system we also propose a solution for similarity searches that combines color histograms, spatial relationships of image blocks and a hash structure to better discriminate among images. Additionally we query images on the basis of salient objects (regions of interest in images) and their properties. This paper presents the querying facilities implemented for the DISIMA system. Both the textual query language (MOQL) and its visual counterpart (VisualMOQL) allow the combination of semantic queries with different types of image queries for better results.

1 Introduction

Multimedia data, especially images, are becoming ubiquitous and are manipulated on a daily basis by computer users. As is the case for all multimedia data, the digital image data does not convey any meaningful information about the image. That is why most image database systems and prototypes focus on content-based image retrieval (CBIR) based visual features such as color, texture and shape [Del99]. The visual features are extracted and represented as points in a multi-dimensional vector space and multidimensional access methods are used to support efficient image searches [SNF00, BBB+97, KSF+96]. The query results returned by these systems can fairly be accurate for some well-defined domains but fail in the general case.

To overcome this problem, an image database has to model and store image semantics that can be used in the querying process. Obtaining these semantics is another issue resolved in most cases using a semi-automated approach. In the DISIMA System, the content of an image is described by means of salient objects (regions of interest) organized hierarchically, following the object-oriented paradigm. The aim of this paper is to show how MOQL [LÖSO97], a declarative query language is used to integrate different image querying approaches: image and salient object semantics, salient object syntactic properties [OÖIL99], and image color distribution [LÖON01].

Section 2 discusses the modelling of semantics in DISIMA. Section 3 presents the querying facilities in DISIMA. Section 4 describes the type system and the query processor. Finally, Section 5 concludes the paper.

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

†College of Computing Sciences, New Jersey Institute of Technology, oria@homer.njit.edu.

‡Department of Computer Science, University of Waterloo, tozsu@db.uwaterloo.ca.

§Department of Computer Science, University of Alberta, iglinski@cs.ualberta.ca.

2 Modelling Semantics in DISIMA

Image semantics in the DISIMA system are captured through salient objects, their shapes, and their spatial relationships within images. The DISIMA model [OÖL+97] provides an efficient representation of images and related data to support a wide range of queries. The DISIMA model is composed of two main blocks: the image block and the salient object block. The image block is made up of two layers: the *image* layer and the *image representation* layer. An image is distinguished from its representations to maintain an independence between them.

At the *image* layer, the user defines an image type classification. Figure 1(b) depicts a partial type hierarchy for an application that involves medical images, electronic commerce catalogs, and news images. These first level image types are derived from the type *Image*, the root image type provided by DISIMA. The type *NewsImage* is specialized by three types: *EnvironmentalImage*, *PersonImage*, and *MiscImage*. An image is an object of an image class with some user-defined properties in addition to low-level feature properties such as textures and color distributions (color histograms).

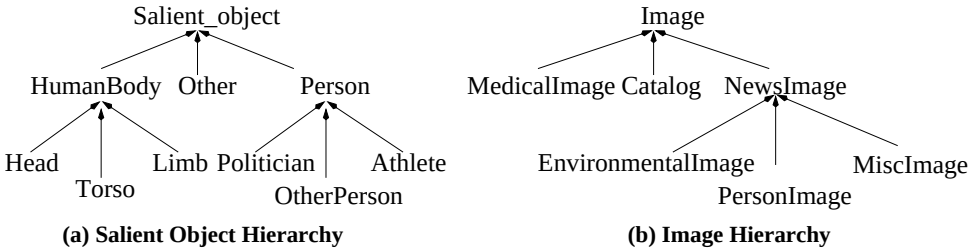


Figure 1: An Example of an Image Hierarchy.

The *salient object* block is designed to handle salient object organization. A simple example of a salient object hierarchy, corresponding to the image hierarchy defined in Figure 1(b), is given in Figure 1(a). DISIMA distinguishes two kinds of salient objects: logical and physical salient objects (LSO and PSO). An LSO is an abstraction of a salient object that is relevant to some application; it is a meaningful object. A PSO is a syntactic object in a particular image (region of an image) with its semantics given by an associated LSO. A PSO has a shape which is a geometric object stored in its most specific class [OÖIL99], a set of colors and textures.

The object-oriented modelling of geometric objects potentially conflicts with the their mathematical definitions. In [OÖIL99], we provided a more general solution to the shape hierarchy design issue based on the mathematical definitions that allows shape group similarity matches in addition to integrating code reuse at both the data structure and the method levels. The shapes of the salient objects are further used to define some spatial qualitative relations between the objects which are very important in multimedia databases because they implicitly support semantic queries which are captured by qualitative reasoning. The spatial model for DISIMA [OÖIL99] is based on Allen temporal algebra [All83]. The shape of the objects are projected onto the axes and the intervals are combined with the allen's temporal operators to define some topological relations in one-dimensional space. The one-dimensional relationships are further combined to define the spatial relationships.

DISIMA permits feature-based search in addition to search based on image content semantics. This allows more sophisticated queries such as the following: (Find images of flowers taken by John that look like this one). The fundamental elements of histogram-based image retrieval include the selection of the color space, the color space quantization, and the histogram distance metric. There is no general agreement as to the most suitable color space for color histogram-based image retrieval. This is a result

of the fact that color perception is highly subjective. Therefore, a variety of color spaces are used in practice such as RGB, HIS, or L*u*v*. In our work, we have chosen to represent the color space using the RGB model. Besides extracting color histograms from entire images, an image can also be segmented into several blocks, each of which has an associated color histogram. These color histograms together form *multi-scale color histograms* of an image. The multi-scale color histograms are used to define image multi-precision similarities [LÖON01].

3 Querying Images

Querying images on salient objects and their properties assumes the detection of these objects. The image annotation is semi-automatic. The image syntactic features are automatically extracted and a human-annotator adds the semantic information on the salient objects in the image. In addition, an image has some descriptive properties (i.e., meta-data), such as date and photographer, that have to be provided. Till now, multimedia data are produced without accompanying meta-data and a human-annotator is often solicited to get this information. This is changing. The emergence of MPEG-7 [Gro01] will lead to media production together with the description of the content descriptions that will provide information such as salient-objects. For the rest of the paper, we assume that the information on salient objects is provided.

3.1 Querying Semantics with MOQL

MOQL (Multimedia Object Query Language) is a text-based multimedia query language [LÖSO97], which is an extension of the standard OQL language [CBB+97].

Most 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 *temporal_expression* deals with temporal objects, functions, and predicates for videos. The *contains_predicate* is defined as: *contains_predicate* ::= *media_object* **contains** *salientObject* where, *media_object* represents an instance of a particular medium type, e.g., an image or video object, while *salientObject* is an object within the *media_object* that is deemed interesting (salient) to the application (e.g., a person, a car or a house in an image). The **contains** predicate checks whether or not a salient object is in a particular media object. The **similarity** predicate checks if two media objects are similar with respect to some metric.

3.1.1 Querying Images Through Salient Objects

The following are two examples of queries expressed in MOQL. The first query looks for images with people and the second finds all image in which a politician named "Chretien" appears.

Query 1 Find all images in which a person appears.

```
select   m
from     Images m, Persons p
where    m contains p
```

Query 2 Find all images in which a politician named Chretien appears.

```
select   m
from     Images m, politician p
where    m contains p
And      p.lastName = "Chretien"
```

Queries in MOQL can easily become non-trivial to express. A query Q: "Find images with 2 people next to each other without any building, or images with buildings without people" can be expressed in MOQL as follows:

Query 3 Find images with 2 people next to each other without any building, or images with buildings without people.

```
select   m
from     image m, building b1, person p1, person p2
where    ( m contains p1 and m contains p2
         And      p1.MBB west p2.MBB
         And      m not in
                  (Select m1
                  From image m1, building b2
                  Where m1 contains b2))
Or       ( m contains b1
         And      m not in
                  (Select m2
                  From image m2, person p3
                  Where m2 contains p3))
```

This example points to the need for a visual query interface. Although the user may have a clear idea of the kind of images he/she is interested in, the expression of the query is not straightforward. The logic in VisualMOQL is based on the observation that queries expressed in natural languages are often composite. VisualMOQL provides a way to construct complex queries by composing simple query blocs or sub-queries.

3.2 Querying Semantics with VisualMOQL

VisualMOQL provides an easier way to express queries, and then translates them into MOQL. Query Q 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". Each of these sub-queries can further be decomposed into 2 simpler sub-queries.

User can choose the image class they want to query and the salient objects they want to see in the images. They can also specify the maximum number of images they want, and the similarity threshold (for similarity queries). The working canvas is where users construct simple queries. They can insert the salient objects that they want to see in images, into the working canvas. The spatial relationships may also be specified between the salient objects. The color, texture, and shape properties of images and salient objects can be specified through a dialog box. After users finish constructing a simple query in the working canvas, it is moved into the query canvas. Several simple queries are combined in the query canvas to form a compound query. Finally, the user presses the query button to submit the query. The VisualMOQL query specified in the query canvas will then be translated into an MOQL query before being submitted to the query processor.

3.3 Feature-Based Searches in DISIMA

DISIMA permits feature-based search in addition to search based on image content semantics. This allows more sophisticated queries such as the following: (Find images of flowers that look like this one).

3.3.1 Color and Texture Similarity for Salient Objects

Since colors and texture are perceived differently by people, searching for an exact match, even at the salient object level, will yield poor results. Color and texture comparisons are done consequently through similarity searches. The user can give the RGB values for the color if he/she knows it or pick a color from an appropriate window to get the color value. The same thing applies to textures. For example:

Query 4 Find all images that contain a salient object with a color similar at 80% to the RGB value (255,0,255) and similar at 70% to the texture value (0.6).

```
select   m
from     Images m, LSO o
where    m contains o
And      o.color similar colorgroup(255,0,255) similarity 0.8
And      o.texture similar texturegroup(0.6) similarity 0.7
```

3.3.2 Shape Similarity

The *Geometric_Object* class supports three types of similarity match: *full-group, class,* and *sub-group,* depending on the similarity threshold specified in the query. 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. The shape similarity algorithm we used is the *turning angle algorithm* [ACH+91] because it is orientation invariant. A shape similarity query can be posed with or without a given shape as illustrated by the two following examples:

Query 5 Find all images containing salient objects with a rectangular shape.

```
select   m
from     Images m, LSO o
where    m contains o
And      o.shape similar rectangle similarity 0.5
```

When a shape is not given (i.e., only the shape class is given) in a shape similarity query, the query is processed without any similarity metric. For the example, the query processor will select images for which at least one salient object has a shape of interface type *Rectangle*. The question is which extent to use (shallow or deep extent)? This decision is made with regard to the similarity threshold in the query. If the similarity threshold is set to 1, the query is processed using the shallow extent (class match) otherwise the deep extent is used (sub-group match).

Query 6 find all images containing salient objects with a a shape 50% similar to the rectangle((1,2),(10,2),(10,7)).

```
select   m
from     Images m, LSO o
where    m contains o
And      o.shape similar rectangle((1,2),(10,2),(10,7)) similarity 0.5
```

If we let ϵ denote the threshold, r the rectangle ((1,2),(10,2),(10,7)), S a set of shapes, PSO a set of physical salient objects , and I a set of images, then the solution of the similarity query is $\{i \in I |\ \exists s \in S,\ \exists o \in PSO,\ d(s,r) \leq \epsilon \wedge s = shape(o) \wedge i\ contains\ o\}$ where d is a distance function using the shape similarity algorithm. The problem here is to find the minimal extent that contains all the shapes satisfying the conditions. For example, if we are trying to match a given rectangle within a

similarity threshold of 0.9, should we begin our search with the deep extent of all polygons or simply confine our search to the extent of all rectangles? Are we willing at higher thresholds, to miss matching some polygons when we match against a similar rectangle? Of course the lower ϵ is, the wider both the solution and the shape search space will be. In the current implementation, full-group match is applied when the similarity threshold in the search condition is less than 1. Class match is applied when the similarity threshold equals 1 (exact match).

3.4 Image Similarity Queries Using MOQL

The *similarity_expression* can also be used to check whether two images are similar with respect to the metric defined in the multi-precision similarity algorithm.

3.4.1 Image Similarity Queries

For querying images by color histogram matching, two kinds of *similarity_expression* are used to check if two images are similar. One is whole-image similarity queries. For example:

Query 7 Find images that are similar to the user-provided image e1, with respect to color histogram matching at precision level 1, with the similarity threshold 0.8.

```
select   m
from     Images m
where    m.color_histogram similar e1.color_histogram
         precision 1 similarity 0.8
```

The other kind of expression is querying sub-images. For example:

Query 8 Find images whose left halves are similar to the user-provided image i, with respect to color histogram matching with the similarity threshold 0.6.

```
select   m
from     Images m
where    m.color_histogram similar i.color_histogram
         quadrants (1, 2) similarity 0.6
```

3.4.2 Combining Similarity Queries with Semantics

A similarity query can be combined with some semantic properties for a more precise response. The result of such a query is all the images similar with respect to the similarity metric which also satisfy the semantic conditions. For example:

Query 9 Find images with people that are similar to the user-provided image e1, with respect to color histogram matching at precision level 2, with the similarity threshold 0.8.

```
select   m
from     Images m, person p
where    m contains p
And      m.color_histogram similar e1.color_histogram
         precision 2 similarity 0.8
```

The same applies to sub-image queries:

Query 10 Find images with people whose left halves are similar to the user-provided image i, with respect to color histogram matching with the similarity threshold 0.6.

```
select   m
from     Images m, person p
where    m contains p
And      m.color_histogram similar i.color_histogram
         quadrants (1, 2) similarity 0.6
```

Note that the precision levels are not defined for the sub-image queries; that is, all the sub-image queries are carried out at the first precision level.

3.5 Image Similarity Queries Using VisualMOQL

An image in the result of a query can be selected as an entry to a similarity query. A dialog box is brought up by pressing the image property button under the working canvas. The right part of the dialog box is for textual properties such as title, publisher, creation date, etc. The left part of this dialog box is for color histogram similarity matching. The similarity can be done on the whole image or a sub-image. For sub-image queries, users can select the region that they want to query on. However, the size and position of the region is limited by the grid partitions that the system provides [LÖON01].

4 Type System and Query Processor

The DISIMA prototype is implemented on top of a commercial object database. The type system provides the data structures to store and index the images that are used by the query processor we implemented to answer user queries.

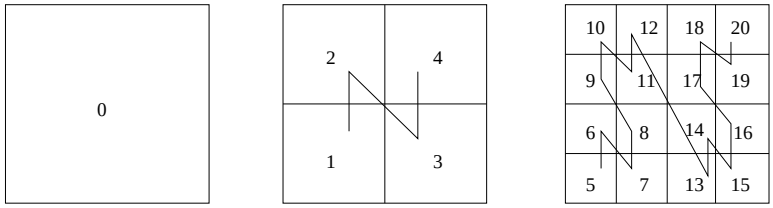
4.1 Type System

The DISIMA type system has been extended to include some structures needed by the multi-precision similarity algorithm. The DISIMA type system is the implementation of the DISIMA model. It 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. Figure 3 shows a high level view of the classes used in the DISIMA type system. The Image, Image Representation, PSO, and LSO classes have been introduced before. The MBB (Minimum Bounding Box) class defines the spatial feature; the Geometric Object class defines the shape feature; the Texturegroup class defines the texture feature; the Colorgroup class and the Multi-scale Color Histogram class define the color feature. The straight line connecting two classes represents the relationship between the classes. The numbers on the straight lines indicate the cardinality of the relationship. The Color Histogram class stores the multi-scale color histograms as quadtrees (Figure 2). An integrated indexing structure (3DEH) is used to index average colors of these color histograms to facilitate image/sub-image queries by color histogram matching [LÖON01].

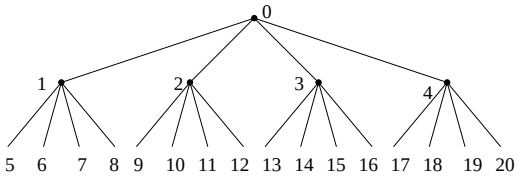
4.2 Implementation of the Index Structure for Image Similarity

While tree-based structures are good for supporting nearest neighbor search, there is considerable traversal that needs to be done. In the DISIMA system, we implemented a hash structure for multi-dimensional indexing called three-dimensional extendible hashing (3DEH). Like traditional hashing, this structure provides efficient, direct addressing of the targeted buckets.

The hash directory of the three-dimensional extendible hashing has three *initial depths* (d_1, d_2, d_3), one for each of the R, G, B color components; it also has a *growth depth* d_g, which is 0 at the



(a) N order of image blocks



(b) quadtree with predetermined children node order

Figure 2: A quadtree stores color histograms of image blocks

beginning and will increase as the address space increases. The number of bits of a hash address is $(d_1 + d_2 + d_3 + d_g)$, so the hash directory has $2^{(d_1+d_2+d_3+d_g)}$ entries. The bucket in three-dimensional hashing has three local depths (p_1, p_2, p_3), which means that all records in the bucket have common p_1, p_2, p_3 leading bits of the R, G, B values, respectively. At the beginning, the local depths of buckets are the same as the initial depths of the directory. The directory entry either points to a bucket or holds a null value if no data records are hashed to this entry.

When a bucket overflows in three-dimensional hashing, like traditional extendible hashing, the hash address space increases and the bucket splits. Unlike traditional extendible hashing, in which a bucket can be split along only one dimension, a bucket in three-dimensional extendible hashing can be split along any of the R, G, B dimensions. We split the bucket along the dimension with the highest variance so that the records can distribute as evenly as possible in the two resulting buckets.

Since the bucket can be split along any one of the three dimensions, we need to record in which dimension it is split. A data structure named *mask track* is maintained to keep track of the splitting history of the bucket. For example, if the bucket 010 split along the R dimension, we record the fact that no buckets have been split, except that the bucket with the initial hash address 010 is split along R dimension in the mask track (Figure 4). That is, every entry in the mask track is zero, except that the 010 entry is 100. We use 3 bits to record that information but the space used for this purpose can be optimized by using fewer bits (e.g., 2 bits for three dimensions). The experiments ran, reported in [LÖON01], show that the 3DEH significantly out performs the SR-tree [KS97]

The three-dimensional extendible hashing is designed to index average colors of the images and their quadrants. If the images are not categorized in the database, then all the images will be simply inserted into one index structure. However, the images in the DISIMA system are organized as hierarchical image classes, so the index structure has to correspond to the hierarchy of images. That is an index for each image class,

4.3 Query Processing

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 parser and query processor for MOQL queries. The result of the parser is an internal query tree structure which is later transformed

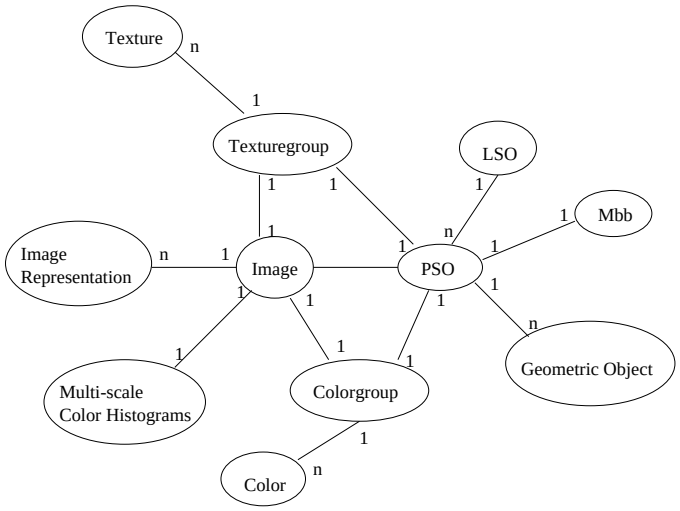


Figure 3: The DISIMA type system overview.

Figure 4: First expansion of the hash directory

into an execution plan.

As MOQL queries follow the same SELECT-FROM-WHERE structure as traditional queries, the design of the DISIMA parser is able to make use of basic rules defined in SQL parsers. The new rules defined on top of the basic rules deal with objects and the clauses introduced by MOQL. The objective of the DISIMA parser is to check the semantics and syntax of the external query, which is in the form of a character string. The parsed string is then converted into a query tree. A query object stores all the information given by the query string.

The query engine uses the query tree directly to generate a non-necessarily optimized execution plan. In the query tree, each node is associated with either one or two conditions. When there are two conditions, either intersection or union is performed upon the results of the conditions for *and* and *or* operators, respectively. Once the query tree is constructed, the query engine initiates post-order traversal. The left-condition is executed before the right-condition, and any sub-node before the higher-level node.

5 Conclusion

In this paper, we have presented the querying functionality of the DISIMA DBMS. We have shown how the the integration of different image query types was achieved in the DISIMA syatem through a declarative query language and a rich type system. We have extended both MOQL and VisualMOQL to support image similarity. Hence, an image similarity search can start with a semantic search to reduce the query domain before the similarity matches. In the current implementation, the semantic information is provided by a human-annotator. But the emergence of MPEG-7 will facilitate the production of media objects together with their content descriptions.

References

[ACH+91] E. M. Arkin, L. P. Chew, D. P. Huttenlocher, K. Kedem, and J. S. B. Mitchell. An efficiently computable metric for comparing polygonal shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(3), March 1991.

[All83] J. F. Allen. Maintaining knowledge about temporal intervals. *Communications of ACM*, 26(11):832—843, 1983.

[BBB+97] S. Berchtoll, C. Bohm, B. Braunmuller, D. Kein, and H. Kriegel. Fast parallel similarity search in multimedia databases. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, Tucson, Arizona, May 1997.

[CBB+97] R. G. G. Cattell, D. Barry, D. Bartels, M. Berler, J. Eastman, S. Gamerman, D. Jordan, A. Springer, H. Strickland, and D. Wade, editors. *The Object Database Standard: ODMG 2.0*. Morgan Kaufmann, 1997.

[Del99] A. Del Bimbo. *Visual Information Retrieval*. Morgan Kaufmann Publishers, 1999.

[Gro01] MPEG-7 Working Group. The MPEG-7 web page. http://www.cselt.it/mpeg/standards/mpeg-7/mpeg-7.htm, 2001.

[KS97] N. Katayama and S. Satoh. The SR-tree: An index structure for high dimensional nearest neighbor queries. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 369—380, Tucson, Arizona, May 1997.

[KSF+96] F. Korn, N. Sidiropoulos, C. Faloutsos, E. Siegel, and Z. Propopapas. Fast nearest neighbor search in medical image. In *Proceedings of the 22nd International Conference on Very Large Databases, VLDB*, Bombay, India, September 1996.

[LÖON01] S. Lin, M. T. Özsu, V. Oria, and R. Ng. An extendible hash for multi-precision similarity querying of image databases. In *Proceedings of the 27th VLDB conference*, Rome, Italy, September 2001.

[LÖSO97] J. Z. Li, M. T. Özsu, D. Szafron, and V. Oria. MOQL: A multimedia object query language. In *Proceedings of the 3rd International Workshop on Multimedia Information Systems*, pages 19—28, Como, Italy, September 1997.

[OÖIL99] V. Oria, M. T. Özsu, P.J. Iglinski, and Y. Leontiev. Modeling shapes in an image database system. In *Proceedings of the 5th International Workshop on Multimedia Information System*, pages 34—40, Indian Wells, California, October 1999.

[OÖL+97] V. Oria, M. T. Özsu, X. Li, L. Liu, J. Li, Y. Niu, and P. J. Iglinski. Modeling images for content-based queries: The DISIMA approach. In *Proceedings of 2nd International Conference of Visual Information Systems*, pages 339—346, San Diego, California, December 1997.

[SNF00] R. O. Stehling, M. A. Nascimento, and A. X Falcao. On "shapes" of colors for content-based image retrieval. In *Proceedings of ACM Multimedia 2000 Workshops*, pages 171—174, Los Angeles, California, November 2000.