

## INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

**The quality of this reproduction is dependent upon the quality of the copy submitted.** Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

ProQuest Information and Learning  
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA  
800-521-0600

UMI<sup>®</sup>



**University of Alberta**

**A VISUAL QUERY FACILITY FOR DISIMA IMAGE DATABASE  
MANAGEMENT SYSTEM**

by

 **Bing Xu**

**A thesis submitted to the Faculty of Graduate Studies and Research in partial  
fulfillment of the requirements for the degree of Master of Science.**

**Department of Computing Science**

**Edmonton, Alberta  
Spring 2000**



National Library  
of Canada

Acquisitions and  
Bibliographic Services

395 Wellington Street  
Ottawa ON K1A 0N4  
Canada

Bibliothèque nationale  
du Canada

Acquisitions et  
services bibliographiques

395, rue Wellington  
Ottawa ON K1A 0N4  
Canada

*Your file* *Votre référence*

*Our file* *Notre référence*

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-60196-X

**Canada**

**University of Alberta**

**Library Release Form**

**Name of Author:** Bing Xu

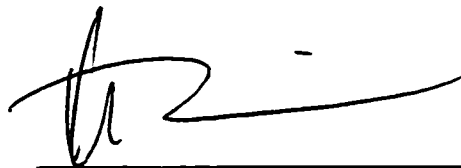
**Title of Thesis:** A Visual Query Facility for DISIMA Image Database Management System

**Degree:** Master of Science

**Year this Degree Granted:** 2000

Permission is hereby granted to the University of Alberta Library to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves all other publication and other rights in association with the copyright in the thesis, and except as herein before provided, neither the thesis nor any substantial portion thereof may be printed or otherwise reproduced in any material form whatever without the author's prior written permission.



---

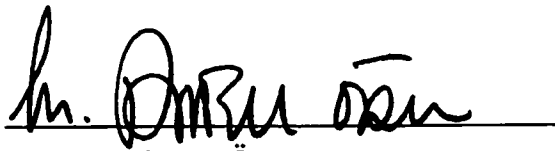
Bing Xu  
615 GSB  
Department of Computing Science  
University of Alberta  
Edmonton, Alberta  
Canada T6G 2H1

**Date:** April 10, 2000

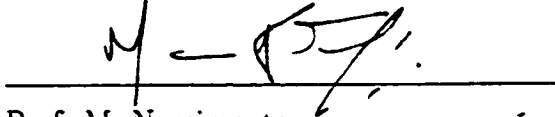
University of Alberta

Faculty of Graduate Studies and Research

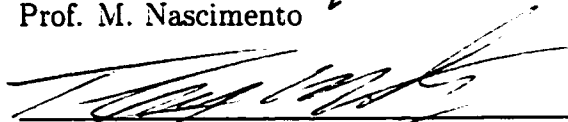
The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled **A Visual Query Facility for DISIMA Image Database Management System** submitted by Bing Xu in partial fulfillment of the requirements for the degree of **Master of Science**.



Prof. M. Tamer Özsu  
Supervisor



Prof. M. Nascimento



Prof. C. Montgomerie  
External Examiner

Date: April 10, 2000

# Abstract

Recent advances in image processing and networking technologies have made the development of an image database management system (DBMS) possible. Images require specific storage management, processing methods and manipulation languages which are application dependent. Managing images for efficient retrieval is a growing need and a challenging issue. One aspect of this challenge is the development of a high level interface to access these databases.

This thesis addresses the design and implementation of a visual query interface, called VisualMOQL, for the DISIMA image DBMS. DISIMA has a text-based query language, MOQL, which is an extension of MOQL with multimedia constructs. VisualMOQL is the visual interface to OQL and combines semantic-based (query image semantics) and textual-based (specify and compare attribute values) query approaches. This combination leads to a flexible, yet powerful visual query interface. A query specified using VisualMOQL is translated into MOQL to make use of the MOQL parser and query processor.

*To my parents*



# Acknowledgements

I would like to express my appreciation to the following people who have made the completion of this study possible.

Dr. Tamer Özsu for his guidance, advice, patience, and for being a great supervisor.

Dr. Vincent Oria for his valuable suggestions and help.

Paul Iglinski for his helpful support.

Irene Cheng, who implemented the query parser and search engine, for her suggestions.

Finally, I want to thank my fiancée Esther for her continuous love, encouragement, and moral support.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Motivation . . . . .	1
1.2	Thesis Scope and Organization . . . . .	3
<b>2</b>	<b>Related Work</b>	<b>5</b>
2.1	Content-based Image Retrieval . . . . .	5
2.1.1	General Approaches . . . . .	5
2.1.2	Visual Query Languages . . . . .	7
2.1.3	Other Image Retrieval Systems . . . . .	9
2.1.4	Conclusions . . . . .	16
2.2	Image Similarity . . . . .	18
2.2.1	Color Similarity . . . . .	19
2.2.2	Shape Similarity . . . . .	26
2.2.3	Texture Similarity . . . . .	31
2.2.4	Other Image Similarity Measures . . . . .	32
2.2.5	Conclusions . . . . .	34
<b>3</b>	<b>The DISIMA System</b>	<b>36</b>
3.1	Objective . . . . .	36
3.2	Functionality . . . . .	36
3.3	The DISIMA Model . . . . .	38
3.3.1	The Model Components . . . . .	38
3.3.2	The Image Block . . . . .	39
3.3.3	The Salient Object Block . . . . .	40
3.3.4	Defining a DISIMA schema . . . . .	44

3.4	The DISIMA Architecture . . . . .	44
3.5	MOQL: A Multimedia Extension of OQL . . . . .	47
<b>4</b>	<b>The User Interface</b>	<b>51</b>
4.1	Introduction . . . . .	51
4.2	VisualMOQL: The DISIMA Visual Query Interface . . . . .	53
4.2.1	Interface Overview . . . . .	54
4.2.2	Salient Object Class Browser . . . . .	56
4.2.3	Working Canvas . . . . .	57
4.2.4	Query Canvas . . . . .	62
4.3	VisualMOQL Query Semantics and Translation . . . . .	64
4.3.1	VisualMOQL Query Semantics . . . . .	64
4.3.2	VisualMOQL Query Translation and Processing . . . . .	65
4.4	VisualMOQL and Other Content-Based Image Retrieval Systems	67
<b>5</b>	<b>Implementation Issues</b>	<b>69</b>
5.1	VisualMOQL Translation . . . . .	69
5.2	Query Parser and Query Engine . . . . .	72
<b>6</b>	<b>VisualMOQL Query Examples</b>	<b>74</b>
<b>7</b>	<b>Conclusions and Future Work</b>	<b>92</b>
7.1	Conclusions . . . . .	92
7.2	Future Work . . . . .	93
<b>A</b>	<b>VisualMOQL Translation Algorithm</b>	<b>105</b>

# List of Figures

2.1	<i>Query interface for QBIC</i>	10
2.2	<i>Query interface for Virage</i>	13
2.3	<i>Query interface for VisualSeek</i>	14
2.4	<i>Query interface for Photobook</i>	15
2.5	<i>Query interface for IFQ</i>	17
2.6	<i>Three color histograms whose perceptual similarities do not correspond to their L-distance</i>	21
2.7	<i>The cumulative histogram of the histograms displayed in Figure 2.6</i>	22
2.8	<i>Turning angle representation</i>	28
2.9	<i>Turning angle representation using Euclidean distance is sensitive to noise</i>	30
3.1	<i>DISIMA model overview</i>	39
3.2	<i>An example of image and logical salient object hierarchies.</i>	40
3.3	<i>An example of salient objects hierarchy</i>	41
3.4	<i>Extended ER diagram for part of entities in the DISIMA model</i>	43
3.5	<i>The DISIMA architecture.</i>	45
3.6	<i>Distribution in DISIMA</i>	46
3.7	<i>Mediator-wrapper architecture</i>	47
4.1	<i>Main window of VisualMOQL</i>	54
4.2	<i>Object property input window for object Person.</i>	58
4.3	<i>Dialog window for defining directional spatial relationship</i>	60
4.4	<i>Definitions of directional relations</i>	61
4.5	<i>Definitions of topological relations</i>	61

4.6	<i>Dialog window for defining image property . . . . .</i>	63
4.7	<i>Warning box appears when users try to AND queries containing incompatible image classes . . . . .</i>	66
5.1	<i>The query tree of a compound query containing negator operator</i>	70
5.2	<i>The query tree of a compound query with negator in normalized form . . . . .</i>	70
5.3	<i>A VisualMOQL compound query with NOT operator . . . . .</i>	71
5.4	<i>The VisualMOQL compound query in Figure 5.3 after being converted to normalized form. . . . .</i>	71
5.5	<i>DISIMA query processing . . . . .</i>	73
6.1	<i>VisualMOQL for query 1 . . . . .</i>	75
6.2	<i>Translated MOQL for query 1 . . . . .</i>	75
6.3	<i>Query result of query 1 . . . . .</i>	76
6.4	<i>VisualMOQL for query 2 . . . . .</i>	77
6.5	<i>Object property window for query 2 . . . . .</i>	78
6.6	<i>Translated MOQL for query 2 . . . . .</i>	78
6.7	<i>Query result of query 2 . . . . .</i>	79
6.8	<i>VisualMOQL for query 3 . . . . .</i>	80
6.9	<i>Spatial relationship dialog window for query 3 . . . . .</i>	81
6.10	<i>Translated MOQL for query 3 . . . . .</i>	81
6.11	<i>Query result of query 3 . . . . .</i>	82
6.12	<i>VisualMOQL for query 4 . . . . .</i>	83
6.13	<i>Translated MOQL for query 4 . . . . .</i>	84
6.14	<i>Query result of query 4 . . . . .</i>	84
6.15	<i>VisualMOQL for query 5 . . . . .</i>	85
6.16	<i>Image property dialog window for query 5 . . . . .</i>	86
6.17	<i>Translated MOQL for query 5 . . . . .</i>	86
6.18	<i>Query result of query 5 . . . . .</i>	87
6.19	<i>VisualMOQL for query 6 . . . . .</i>	88
6.20	<i>Translated MOQL for query 6 . . . . .</i>	89
6.21	<i>Query result of query 6 . . . . .</i>	89

6.22	<i>VisualMOQL for query 7 . . . . .</i>	90
6.23	<i>Translated MOQL for query 7 . . . . .</i>	91
6.24	<i>Query result of query 7 . . . . .</i>	91

# Chapter 1

## Introduction

### 1.1 Motivation

Multimedia information has become a very important part of our everyday life. From textbooks to computer games, from geographical information systems to the latest web sites, the popularity of multimedia is growing at a rapid rate. While this provides people greater opportunity to explore more information, it also poses a new problem: how can users find the desired information easily and effectively? In the last ten years, a significant amount of research effort has been directed into this area, especially in the context of image databases.

An image database is a database that stores both image data and associated symbolic data. An image DBMS supports acquisition, storage, management, and access to image data, similar to what traditional database systems do with conventional data. The usage of image databases is increasing. For example, a fashion designer may need to find a fabric of a certain color, a visitor may want to search a painting by Picasso from a museum's collection, a doctor may need to find a X-ray image of a tumor with particular shape, and a newspaper editor may require an image database to retrieve pictures s/he wants.

Unlike the problem of data processing in traditional alphanumeric database systems, retrieving images efficiently and effectively in databases is difficult. Due to their visual nature, it is hard, if it is possible at all, to find a verbal description for image content. This makes efficient image retrieval a challenging issue.

In order to build an image database system, we need to find answers to several questions: Would an easy text search be enough? Should the database provide both image manipulation features and information retrieval functionality? How should we design the interface so that not only experts, but also naive users, can use the database? What kind of description is most suitable?

Most early image retrieval systems are text-based, in that relevant text annotations are attached to each image and stored in a traditional database. These annotations are used as the basis for retrieval after the user enters the keywords for the image that s/he wants to retrieve. This approach, known as “query-by-caption” [32], has the following disadvantages:

1. User queries are restricted by the particular vocabulary used, because keywords do not allow unanticipated search in subsequent applications.
2. It is hard to obtain consistent annotations.
3. It does not catch the semantic features of the images or support queries based directly on the visual features of the images. For example, it is difficult, if not impossible, to specify texture by means of words.
4. It can't provide queries based on similarities between images.

To overcome the problems of text-based image retrieval systems, the use of image content features as the basis for retrieval has been investigated-where searches are performed directly on image data as opposed to searches of associated textual information that has been “attached” to each database image by an interpreter or analyst. This approach allows the user to query over the color, texture, shape of objects, and even the spatial relationship among objects in the image. Content-based retrieval will greatly improve the value of massive image databases, especially for those applications where data are collected in such staggering quantities that human experts will only be able to examine a small portion of the entire data set.

The concept of content-based retrieval alleviates several technological problems. It gives a user the power to retrieve visual information by asking a query



like “Show me all pictures that look like this”. The system satisfies the query by comparing the content of the query picture with that of all target images in the database. This is called query-by-example (QBE) [18] and is a simple form of content-based retrieval.

Traditional database access is via well-defined textual query languages. Some image database systems have adopted this idea and developed interfaces based on keyword and SQL-like query languages. These are more precise, but not natural or friendly to the users, since they would require knowledge of the database schema and learn a query language. Due to the diversity of image database users, such requirements are often not realistic. Another disadvantage of this approach is that query languages do not visualize queries, so users are not able to match the query representation with their mental models. An image database needs a visual query interface to allow effective and easy access to the database.

Another approach is to develop more user friendly interfaces. Approaches include cognition-based interfaces that support query by image examples [64, 19], and a descriptive semantics-based approach [23] in which users pose queries by describing the semantics of the target image. These interfaces have the advantage of being more natural to users, but they have low precision, because the queries are ambiguous.

To fill the gap between these two approaches, an image database query interface should not only visualize queries, but also support precise manipulations of query construction. Moreover, image data are semantically richer than the traditional text data. The fact that segments in an image have associated semantics and that they constitute meaningful structures is ignored by most systems. Users should be able to query objects with finer granularity than a whole image.

## 1.2 Thesis Scope and Organization

This thesis identifies the query interface requirements of an image database. It addresses the design and implementation of a visual query interface, called

VisualMOQL, for the DISIMA image database management system [62]. DISIMA has a text-based query language, MOQL, which is an extension of OQL with multimedia constructs. VisualMOQL is the visual interface to MOQL and combines semantic-based (query image semantics) and textual-based (specify and compare attribute values) query approaches. This combination leads to a flexible, and yet powerful visual query interface which allows the user to access the DISIMA image DBMS easily and more effectively. A query specified using VisualMOQL is translated into MOQL to make use of the MOQL parser and query processor.

This thesis is organized as follows: Related issues in image database and visual interface design are discussed in Chapter 2. Chapter 3 explains the DISIMA DBMS. The visual query interface is detailed in Chapter 4. Chapter 5 discusses some implementation issues. Sample VisualMOQL queries are presented in Chapter 6. Chapter 7 concludes the thesis and proposes some future work.

# Chapter 2

## Related Work

Building visual interfaces to databases has become a fairly active research area. There are a number of systems that have been proposed, with varying capabilities [24, 35, 46, 58, 66, 74, 75, 76]. Building such interfaces for image databases is more involved, since the systems have to be able to deal with issues such as similarity searches and content-based querying. In this chapter related work in these areas is reviewed.

### 2.1 Content-based Image Retrieval

#### 2.1.1 General Approaches

With the advent of image technologies such as like pattern recognition, image compression, and edge detection, considerable research effort has been directed into image processing and image retrieval during the last several years. This work has made content-based image retrieval systems possible.

Research on ways to extend and improve content-based query methods for image databases is widespread. Currently, there are more than twenty research and three commercial image retrieval systems and these numbers keep growing.

Two main approaches have been investigated. The first one is to use traditional SQL-like query languages or keyword-based search techniques. As a result, the interfaces are built based on the system's view. They often require the user to learn the database schema and become a master of the query language. This causes serious problems since most image database users are not computer experts and do not have time to learn a new query language.

Recently developed image information systems lean more toward the query-by-example paradigm, where the user provides an example image by choosing it from the database or partly drawing what a desired image should look like. The most popular image features used in these content-based retrieval systems are color, texture, and shape. Besides extracting the global features of images, some systems also calculate the regional color and texture. These techniques have been applied to overall image content without taking into account the characteristics of individual objects. While the techniques work well for the retrieval of images with similar overall content (including backgrounds), their accuracy is limited because they are unable to take advantage of individual objects' visual characteristics, and to perform object-level retrieval.

Other information systems have focused on the representation of spatial structures and efficient access methods to these structures, such as 2D-string feature indexing [21]. In these approaches image contents are modeled as a set of user defined attributes, extracted automatically or manually, and managed within the framework of conventional database management systems. The values of the attributes are grouped together and translated into a signature of the image. The signature is then stored in the database with the corresponding image. When the user presents a query containing a set of desired values, it is first translated into a signature, and then compared with the image signatures stored in the database. The similarity between them is computed using some distance function, the most common one being Euclidean distance. Finally, a set of ranked images is returned to the user as the query result. Attribute-based representation of images entails a high level of image abstraction. The major disadvantage of these symbolic methods is that they cannot handle direct spatial queries. For example, given a GIS map, the query "Is there a river within 5km distance of Toronto" cannot be answered by the above approaches. Another limitation is the difficulty of modeling the temporal and evolutionary relations in the data model.

## 2.1.2 Visual Query Languages

The effectiveness of an image retrieval system ultimately depends on the types and correctness of the image content representations used, the types of image queries allowed, and the efficiency of search techniques implemented. These factors pose new challenges for the traditional query languages and user interfaces.

A query language is a set of formally defined operators which allow users to express requests to a database. While traditional query language (SQL) can handle alphanumeric data very well, it does not fulfill the requirements for new image retrieval systems. The main reasons are:

- It is not user friendly.
- It does not have operators to handle the complex spatial, temporal, and evolutionary relationships among objects in the images.
- More importantly, it deals only with queries based on exact match, while content-based image queries require similarity match.

Trying to overcome the disadvantages of traditional query languages, researchers have proposed several new query languages designed for multimedia databases in the last few years. These proposals can be classified into three categories:

- Entirely new and specialized languages: [5, 16, 33, 37, 47].
- Languages that are based on a logic or functional programming approach: [28, 55].
- Languages that are extensions of the standard database query language SQL: [2, 10, 60, 61, 69].

It has been pointed out that completely new multimedia query languages (at least those that have been proposed) present the problem of lack of theoretical framework to reason about the soundness and expressive power of the

languages [51]. It is also not easy to convince users to learn and use a new language for every new application domain. The majority of existing proposals of new multimedia query languages are based on extensions of SQL.

As analyzed before, formal query languages alone are not suitable for the user interface of image databases since they do not fulfill the requirement of user friendliness. Users should be able to construct queries in a more natural way than using the query language directly. Recently, many visual interfaces for multimedia database systems have been proposed in the literature. They adopt a range of different visual representations and interaction strategies.

Several authors singled out a number of usability features that can be obtained with direct manipulation visual techniques [81]:

- Shortening of the distance between the user's mental model of reality and the representation of such reality proposed by the computer;
- Reduction of the dependency on the native language of the user;
- Ease in learning of the basic functionality of the interaction;
- High efficiency rate obtained also by expert users, partly because of the possibility of defining new functions and features;
- Significant reduction in the error rate.

Visual languages coupled with internal query languages provide an abstract set of operators and a mechanism to express operators defined in the internal language. There are a few prototypes using this approach. The QBD\* visual language [4] is not directed at multimedia applications but is one of the first visual languages to separate the external model and querying from the internal query language. The QBD\* visual query language is based on an entity-relationship representation at the external level and uses SQL internally. A query is expressed by selecting an entity, then following a path by choosing a relationship. Conditions can be expressed on attribute values. *Delaunay<sup>MM</sup>* [25] is a framework for querying multimedia data stored in distributed data repositories, including the web. The *Delaunay<sup>MM</sup>* query interface provides a

format in which users can enter object SQL-like queries. The queries are then translated into the syntax accepted by the destinations.

### 2.1.3 Other Image Retrieval Systems

IBM's QBIC system [58] is one of the early image retrieval systems in which the user provides value-examples for one or more visual features. QBIC (Figure 2.1) is able to not only pick out the colors in an image, but also to gauge texture by measures—contrast, coarseness, and directionality. It uses simple keyword descriptors and allows text-based search. It also has a limited ability to search for shapes within an image. However, the user cannot search on objects and the spatial relationships among them in the images.

SaFe [76] is a general tool for spatial and feature image search. It provides a framework for searching and comparing images by utilizing the spatial arrangement of regions or objects. In a SaFe query, objects or regions are assigned by the user. These are given properties of spatial location, size, and visual features such as color. SaFe also resolves spatial relationships, which allows the user to position objects relative to each other in a query. But as with QBIC, it does not support queries based on salient objects and visual properties.

VISUAL [8] is a graphical icon-based query language. It is designed for scientific databases where the data has spatial properties. VISUAL is nonprocedural and uses the example element concept of query-by-example [82] to formulate query objects. The data model used is an object-oriented data model with complex objects, which can be built by set, bag, tuple, and sequence constructors. Utilizing graphical icons, VISUAL allows users to express visual queries in an incremental fashion. VISUAL has an object-oriented query specification model. A query object is composed of subquery objects, which can be saved and reused.

The UCLA Medical Image Model [38] consists of two layers. The lower layer represents the related image and object contours. The upper layer abstracts objects from images. Additional hierarchical, spatial, temporal, and evolutionary constructs help capture the relationships among objects. The prototype

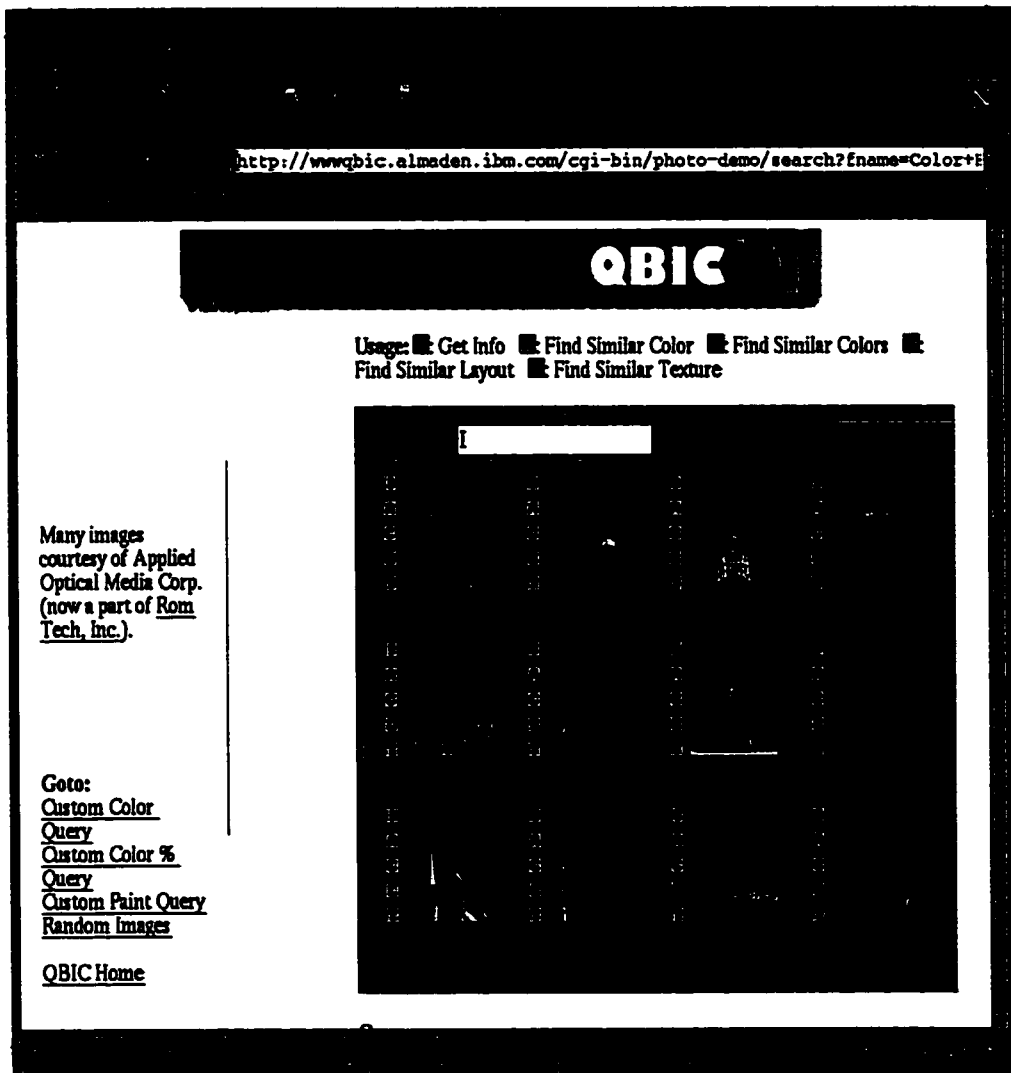


Figure 2.1: Query interface for QBIC



built on top of the commercial object-oriented DBMS Gemstone [15, 12] provides a framework for image preprocessing and feature extraction. Its query language, SEQL (Spatial Evolutionary Query Language), allows users to express queries with temporal, evolutionary and spatial predicates.

WebSEEK [74] is a web image search engine developed at Columbia University. It lets users begin a search by selecting a category from a menu, say “cat”. WebSEEK provides a sampling of icons for the “cat” category. To narrow the search, the user can click on any icon that shows black cats. Using its previously generated color analysis, the search engine looks for matches of images that have a similar color profile. The presentation of the next set of icons may show black cats — but also some marmalade cats sitting on black cushions. A visitor to WebSEEK can refine a search by adding or excluding certain colors from an image when initiating subsequent queries. WebSEEK has downloaded and indexed more than 650,000 pictures from tens of thousands of web sites. While providing search ability on objects and some visual properties, WebSEEK lacks the ability to process spatial queries. And since it is not based on a formal multimedia query language, the query is not very precise.

ImageRover [71] is an experimental content-based web image browser. At search time, users can iteratively guide the search through the selection of relevant examples. The query is formulated based on the composition of color, texture, and dominant orientations presented in the user-selected example images. Query-by-example interface has been developed and is accessible via a web interface.

Los Alamos National Laboratory has developed the CANDID (Comparison Algorithm for Navigating Digital Image Databases) system [45, 46] which employs the idea of comparing a global signature generated to describe an entire image, or a specific region of interest, to other global signatures. When a user queries the database to retrieve images that are similar to a given image, a global signature for that example image is first computed, and then compared to the signatures of all images in the database. This system has been used for both a medical application (retrieving pulmonary CT images to identify

patients suffering from similar lung diseases) and a remote-sensing application (searching a database of digital satellite imagery), and has obtained promising results. However, both ImageRover and CANDID have the same general problems that query-by-example approaches have — namely, they do not capture the image semantics represented by salient objects and their spatial relationships.

Virage (Figure 2.2) is an image retrieval system based on visual features as image primitives. These primitives can be general, such as color, shape, or texture, or domain specific. Virage transforms images from a data-rich representation of explicit image pixels to a more compact, semantically rich representation of visually salient characteristics. The Virage model [34] uses layered views of image data. The system is made up of four layers: image representations and relations (IR), domain objects and relations (DO), image objects and relations (IO) and domain events and relations (DE). The Virage system allows the user to incrementally express incomplete queries, starting from semantic pictorial objects with similarity predicates, and it can retrieve images or part of images.

VisualSeek [75] (Figure 2.3) is a image database system which provides color and spatial querying. Since the discrimination of images is only partly provided by global features such as color histograms, VisualSeek instead utilizes salient image regions and their colors, sizes, spatial locations, and relationships to in order to compare images. This allows the user to form queries by diagramming spatial arrangements of color regions. The system finds the images that contain the most similar arrangements of similar regions. Prior to the queries, the system automatically extracts and indexes salient color regions from the images. By utilizing efficient indexing techniques for color information, region sizes and absolute relative spatial locations, a wide variety of complex joint color/spatial queries may be computed.

Photobook [66] (Figure 2.4) is set of interactive tools for browsing and searching images, based on image content. It uses semantics-preserving image compression, which reduces images to a small set of coefficients, and compares features associated with images, rather than the images themselves. The fea-

<http://www.virage.com/virdemo.html>

PRODUCTS & TECHNOLOGIES

**VIR**  
TECHNOLOGY DEMO

CLICK AN IMAGE TO FIND SIMILAR IMAGES

**VIRAGE**

Color  10

Composition  5

Texture  5

Structure  5

Show  12  images

tab tab tab

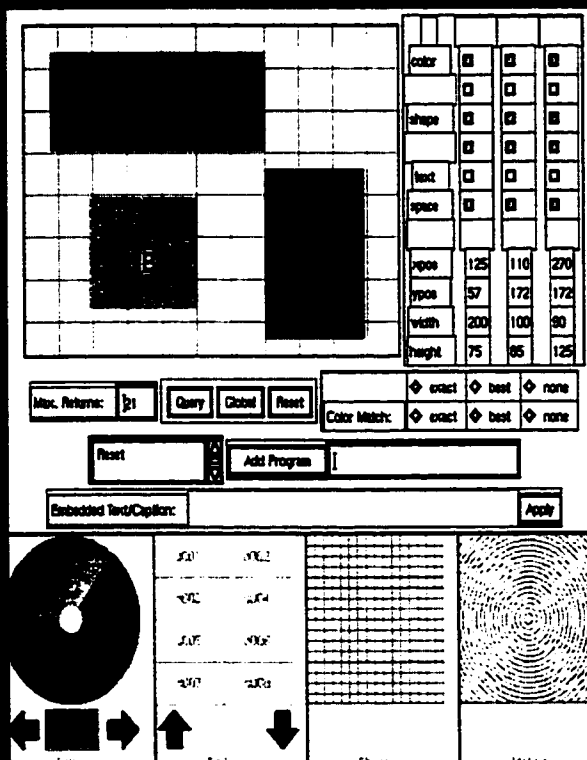
tab tab tab

tab tab tab

tab tab tab

Figure 2.2: Query interface for Virage

<http://www.ctr.columbia.edu/~jrsmith/VisualSEEK/VisualSEEK.html>



Since January 14, 1996

A

Figure 2.3: Query interface for VisualSeek

tures used are commonly color, texture, and shape. Features are compared using some matching algorithms provided by the Photobook library. In version 5, the algorithms include Euclidean, Mahalanobis, divergence, vector space angle, histogram, Fourier peak, and wavelet tree distances, as well as any linear combination of these. Version 6 allows user-defined matching algorithms via dynamic code loading.

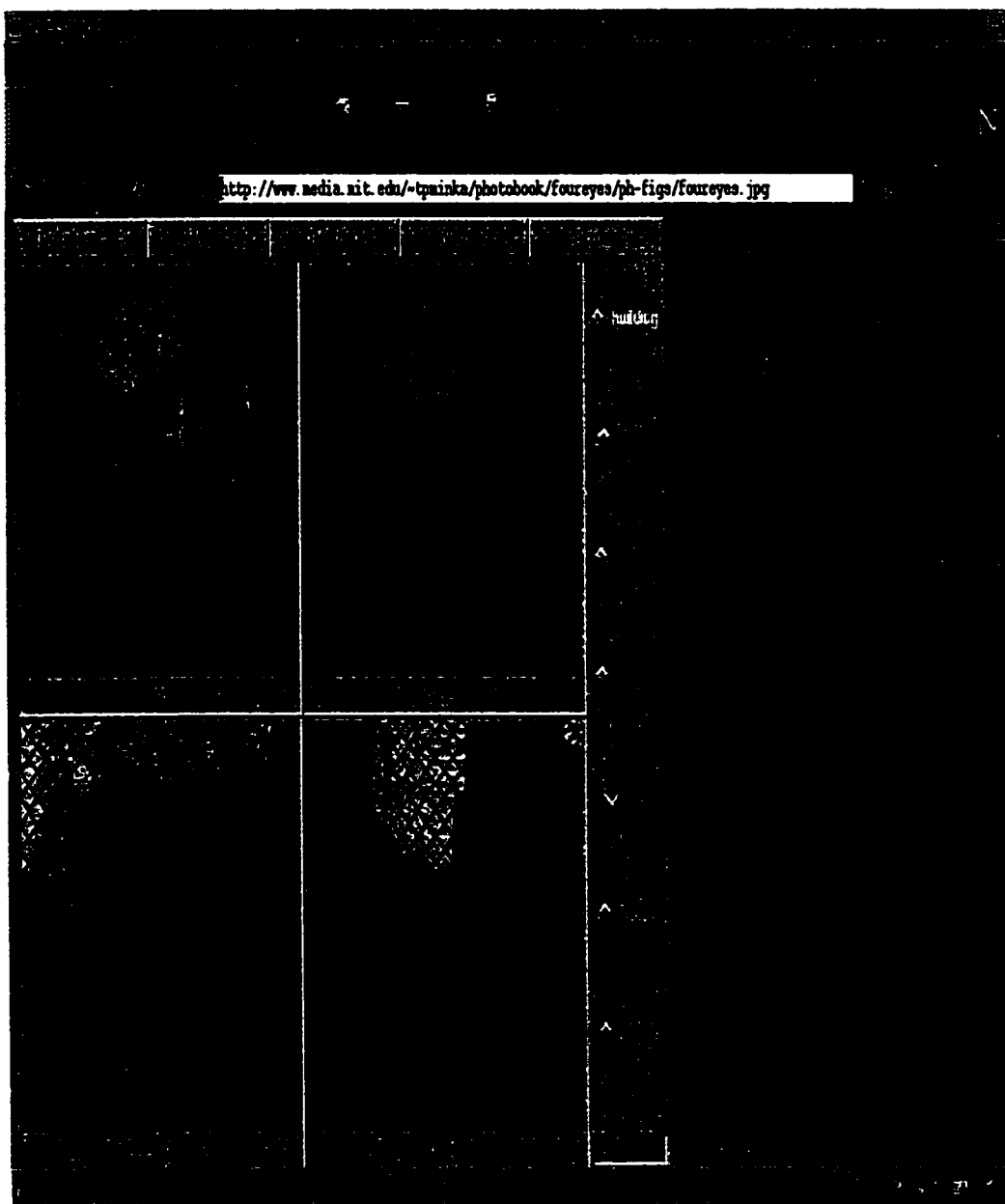


Figure 2.4: *Query interface for Photobook*

Photobook includes FourEyes [57], an interactive learning agent which selects and combines models based on examples from the user. This makes Photobook different from tools like QBIC and Virage, which support a search on various features, but offer little assistance in actually choosing one for a given task. FourEyes, by contrast, allows users to directly address their intent.

Although Virage, VisualSeek, and Photobook support queries on color, texture, and other visual properties, they do not provide a view of the internal data structure. And they do not support complicated queries involving logical operators (like OR, NOT) either.

SEMCOG (SEMantics and COGnition-based image retrieval) [53] is another visual query language developed at NEC and CCRL. It integrates semantic and cognition-based approaches with a visual approach. The query is posed by specifying image objects and their layout using the visual query interface IFQ (In Frame Query) (Figure 2.5). A query specification in IFQ is done in two steps. First, the user introduces image objects, describes them, and then specifies their spatial relationships. A query specified using IFQ is then translated into CSQL (Cognition and Semantics-based Query Language), which is a SQL-like language extended with semantic predicates *is\_a* (specialization), *s\_like* (semantic like), *i\_like* (image like), *contains*, and spatial relationships. SEMCOG is being implemented on top of a deductive and a commercial object-relational DBMS. A comparison between IFQ and Visual-MOQL is given in Section 4.4.

#### 2.1.4 Conclusions

In most of the interfaces developed for image databases, users can specify the color, texture, and shape of the objects. Query by example image is also available in most of the systems (QBIC [58], Virage [7]).

However, current image retrieval systems have many significant limitations. Many are specialized for a particular class of images and/or queries, supporting relatively weak querying by content (i.e., by color, texture or shape, but with no deeper understanding of the structure of the image). We can make significant improvement by providing a visual interface on top of the formal

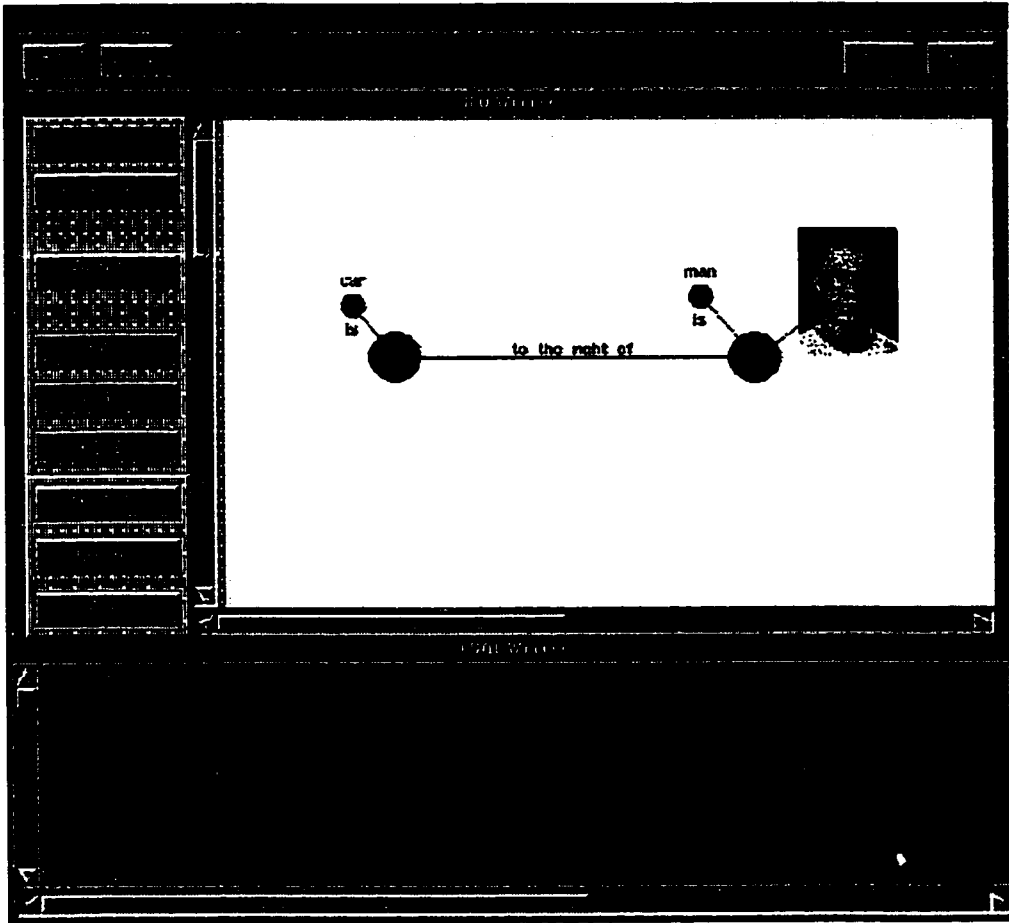


Figure 2.5: Query interface for IFQ

multimedia query language in order to bridge the gap between users' perceptive concepts and direct manipulation of query languages.

## 2.2 Image Similarity

In content-based image retrieval, similarity measures, rather than exact match, are used for the search. The similarity selection operation is the generalization of the  $k$ -nearest neighbor query and spherical range query, and approximate versions of those queries [78]. The parameters of the similarity selection query operation are the following:

1. A query vector  $q \in R^d$ . The query results are ordered in increasing distance from  $q$ , where distance is usually the Euclidean or the weighted Euclidean distance.
2. A positive integer  $k$  that specifies the maximum number of nearest neighbors returned in the query result.
3. A positive real number  $T$  that specifies the maximum distance from the query vector to any vector in the query results. Note that if  $T$  is larger than the distance from  $q$  to the  $k$ th nearest neighbor, then an exact query returns the  $k$ -nearest neighbors of  $q$ . If  $k$  is larger than the number of objects within distance  $T$  of  $q$ , then an exact query becomes a spherical range query bounded by  $T$  relative to  $q$ .
4. A non-negative real number  $\epsilon$  that specifies a bound on the approximation error. The approximation error bound is given by

$$\frac{D}{D^*} \leq 1 + \epsilon \quad (2.1)$$

where  $D$  is either equal to  $T$  if  $k$  result vectors have not been found, or is the distance from  $q$  to the current  $k$ th nearest neighbor vector in the approximate query result; and where  $D^*$  is the distance from the closest missed vector to the query point  $q$ .



The similarity between two images can be measured by comparing them in their entirety (whole-matching or global matching) or by comparing parts of them (sub-pattern matching or local matching). Global matching is relatively easy to do and needs less storage. Typically, a few numbers will be computed over the entire image and two images are then considered similar if these numbers are close to each other. The most common features used here are color and texture. Global matching is limited because it cannot take advantage of the local features in an image. Local matching needs more complicated algorithms to capture and compare more features of an image, such as the objects contained and their properties. It also needs more storage and longer search time, but it is usually more accurate and provides more information and query power to the user.

The interface for an image DBMS should provide tools to support the image similarity algorithms used in the system. And often, these algorithms affect the design of the interface. In the following sub-sections, we discuss some commonly used algorithms for calculating the image similarities.

### 2.2.1 Color Similarity

Color indexing is one process by which the images in the database are retrieved on the basis of their color content. A color indexing system requires that several important objectives are satisfied, namely: automated extraction of color, efficient indexing, and effective retrieval.

There are two techniques for color indexing: (1) global color distribution and (2) local or region color. An important distinction between these techniques is that indexing by global distribution enables only whole images to be compared, while regional indexing enables matching among localized regions within images. Both techniques are very useful for retrieval of images, but are best suited for different types of queries.

Color indexing by global distribution is most useful when the user provides a sample image for the query and is not concerned with the positions of colored objects or regions in the images. However, it does not provide the means for resolving the spatially localized color regions from the global distribution when

the user is interested in finding objects within images. On the other hand, color indexing by localized or regional color supports partial or sub-image matching between images. Localized or regional color indexing is generally more difficult because it requires the effective extraction and representation of local regions. In both cases a system is needed for the automated extraction and efficient representation of color to support effective image retrieval.

Color indexing is introduced in [79]. The coarsely quantized color histograms of the images are stored in the index. The color histogram  $H$  containing  $n$  colors of image  $M$  is a vector  $(h_{c_1}, h_{c_2}, \dots, h_{c_n})$ , where each element  $h_{c_i}$  represents the number of pixels of color  $c_i$  in the image  $M$ . It is assumed that all images contain  $N$  pixels and hence  $\sum_{i=1}^n h_{c_i} = N$ .

The  $L_1$ -distance and  $L_2$ -distance of two color histograms,  $H$  and  $I$ , are defined as

$$D_{L_1}(H, I) = \sum_{j=1}^n |h_{c_j} - i_{c_j}| \quad (2.2)$$

$$D_{L_2}(H, I) = \sqrt{\sum_{j=1}^n (h_{c_j} - i_{c_j})^2} \quad (2.3)$$

Some early experiments [78] show that this method does not retrieve all the images with perceptually similar color histograms. This happens because perceptually similar color histograms may be a large distance apart from each other. For example, in Figure 2.6, the distance between  $H_1$  and  $H_2$  should certainly be smaller than the one between  $H_1$  and  $H_3$  if we order the bins of color histogram in a way such that neighboring bins correspond to similar colors. But the distances are:

$$D_{L_1}(H_1, H_2) = 2N > D_{L_1}(H_1, H_3) = D_{L_1}(H_2, H_3) = 1.33N \quad (2.4)$$

$$D_{L_2}(H_1, H_2) = 0.82N > D_{L_2}(H_1, H_3) = D_{L_2}(H_2, H_3) = 0.66N \quad (2.5)$$

This clearly shows the undesired effect of using the  $L$ -distance as a similarity measure of color histograms.

In order to overcome the shortcomings of existing methods, a method called cumulative color histogram has been proposed [78]. The cumulative color histogram  $\tilde{H}(M) = (\tilde{h}_{c_1}, \tilde{h}_{c_2}, \dots, \tilde{h}_{c_n})$  of the image  $M$  is defined in terms of the color histogram  $H(M)$ :

$$\tilde{h}_{c_j} = \sum_{c_i \leq c_j} h_{c_i} \quad (2.6)$$

and the distance functions are defined as follows:

$$D_{L_1}(\tilde{H}, \tilde{I}) = \sum_{j=1}^n |\tilde{h}_{c_j} - \tilde{i}_{c_j}| \quad (2.7)$$

$$D_{L_2}(\tilde{H}, \tilde{I}) = \sqrt{\sum_{j=1}^n (\tilde{h}_{c_j} - \tilde{i}_{c_j})^2} \quad (2.8)$$

$$D_{L_\infty}(\tilde{H}, \tilde{I}) = \max_{1 \leq j \leq n} |\tilde{h}_{c_j} - \tilde{i}_{c_j}| \quad (2.9)$$

The cumulative histograms of  $H_1, H_2$ , and  $H_3$  in Figure 2.6 are shown in Figure 2.7. The distances between the cumulative histograms are

$$D_{L_1}(H_2, H_3) = 4N > D_{L_1}(H_1, H_3) = 3N > D_{L_1}(H_1, H_2) = N \quad (2.10)$$

$$D_{L_2}(H_2, H_3) = 1.41N > D_{L_2}(H_1, H_3) = 1.20N > D_{L_2}(H_1, H_2) = 0.58N \quad (2.11)$$

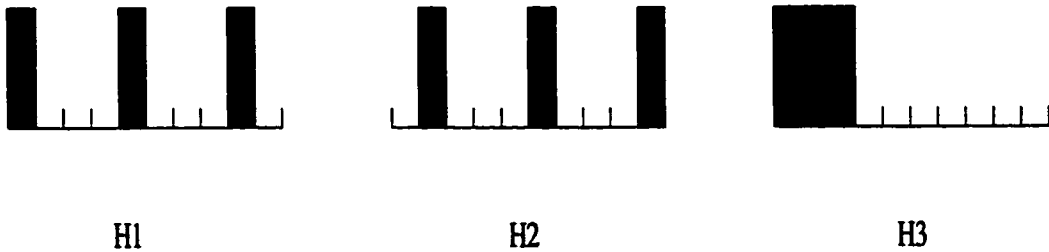


Figure 2.6: *Three color histograms whose perceptual similarities do not correspond to their L-distance*

$$D_{L_\infty}(H_2, H_3) = D_{L_\infty}(H_1, H_3) = 0.67N > D_{L_\infty}(H_1, H_2) = 0.33N \quad (2.12)$$

The order resulting from the cumulative color histogram distance corresponds to the intuitive order of the similarities of the color histograms. It has been shown that cumulative color histograms, together with the L-metrics, are more robust with respect to the quantization of the histograms than the L-distances applied to color histograms [78].

The cumulative color histograms are always completely dense vectors even if only a few colors of the discrete color space appear in each image. Usually, the non-empty bins in color histograms are sparse, and, thus, the price paid for the increased robustness of cumulative color histograms is an increase in the index size and a slower retrieval speed.

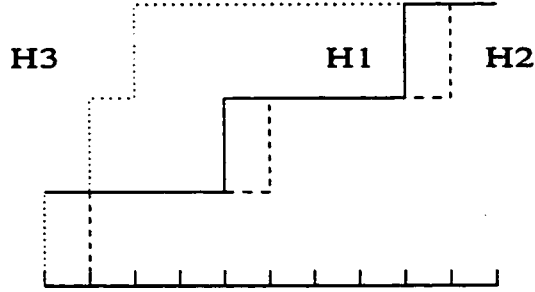


Figure 2.7: *The cumulative histogram of the histograms displayed in Figure 2.6*

Color histogram intersection for color image retrieval is proposed in [73]. It provides useful information such as “30% of the color distribution of image A is similar to the color distribution of image B”. The intersection of histogram  $H$  and  $I$  is given by:

$$D_T(H, I) = \frac{\sum_{j=1}^n \min(h_{c_j}, i_{c_j})}{\min(\sum_{j=1}^n h_{c_j}, \sum_{j=1}^n i_{c_j})} \quad (2.13)$$

Colors not presented in the user’s query image do not contribute to the intersection. This formulation differs from that originally proposed by [79] in order to make it a true distance metric. The intersection formula in [79] is not symmetric in  $h$  and  $i$  and therefore is not a distance metric.

In IBM's QBIC (Query By Image Content) system, both the global and local color information are represented by mean color and color histogram [58]. This system defines a metric making use of the color similarities of the bins in the color histograms. If the color similarity of the  $j$ -th and  $l$ -th bin is  $a_{jl}$  and the symmetric matrix  $A$  with entries  $a_{jl}$  is positively definite, then it defines a new distance

$$D_A(H, I) = \sqrt{\sum_{j=1}^n w_j \cdot (h_{c_j} - i_{c_j})^2} \quad (2.14)$$

where  $w_j$  is the eigenvalues of the matrix  $A$ . If we choose all weights as 1, *i.e.*,  $D_A = D_{L_2}$ . This method still suffers from the same problem as  $D_{L_1}$  as shown in Equation 2.11. Since the quadratic distance measure is very computationally intensive, the mean color distance is used as a pre-filter for color queries.

A new system for color indexing that provides automated extraction of local color regions and efficient indexing is presented in [73]. First, the system defines a quantized selection of  $n$  colors that will be indexed in the database. A binary color vector  $\hat{c}$  for a color region in the image is a vector  $(c_1, c_2, \dots, c_n)$ , where  $c_i$  is either 1 if color  $c_i$  is found in the region, or 0 if it is not. In order to be captured in the index by a color set, a region must meet two requirements:

1. There must be at least  $N$  pixels in the region, where  $N$  is a user defined integer.
2. Each color in the region must contribute at least  $t\%$  of the region area, where  $t$  is also user defined.

Each region is represented by the minimum bounding box and added to the database index. The information stored for each region includes the color set, region location, region size, and image ID. Together, they enable a rich variety of queries that specify not only color content but also the spatial relationships and composition of color regions. Color set technique provides an alternative to color histograms for representation of the color content of images and regions.

A similarity measure based on the color moments is proposed in [78]. The color distribution of an image is interpreted as a probability distribution, then the color distribution can be characterized by its moments. The first moment is the average, the second is the variance, and the third is the skewness. So the average color, the standard deviation, and the third root of the skewness of each color channel are stored in the index. Hence, all the values in the index have the same units, which makes them somewhat comparable. If the value of the  $j$ -th image pixel is  $p_j$ , then the index entries are:

$$E = \frac{1}{N} \sum_{j=1}^N p_j \quad (2.15)$$

$$\sigma = \left( \frac{1}{N} \sum_{j=1}^N (p_j - E)^2 \right)^{\frac{1}{2}} \quad (2.16)$$

$$s = \left( \frac{1}{N} \sum_{j=1}^N (p_j - E)^3 \right)^{\frac{1}{3}} \quad (2.17)$$

Let  $H$  and  $I$  be the color distribution of two images with  $r$  color channels. If the index entries of these images are  $E_i$  resp.  $F_i$ ,  $\sigma_i$  resp.  $\varsigma_i$ , and  $s_i$  resp.  $t_i$ , then the distance is defined as

$$D_{mom}(H, I) = \sum_{i=1}^r (w_{i1} | E_i - F_i | + w_{i2} | \sigma_i - \varsigma_i | + w_{i3} | s_i - t_i |) \quad (2.18)$$

where  $w_{ki} \geq 0 (1 \leq l, k \leq 3)$  are user defined weights and can be used to tune the similarity function for specific application. For example, if all the images in the database were taken under the same lighting conditions, then the weights can be set such that  $w_{i1} > w_{i2}$  and  $w_{i1} > w_{i3}$  in order to penalize shifts in the average color. For HSV color space, where usually the hue must be matched more strictly than the saturation and the value, all the weights for the moments of the hue channel can be set to a higher value than the other weights.

It should be noticed that  $D_{mom}$  is not a metric, *i.e.*, it is possible that two non-identical color distributions have a similarity value of 0. A retrieval based on  $D_{mom}$  may produce a false positive because the index contains no

information about the correlation between the color channels. Nevertheless, the experimental results reported show that this method is more robust and faster than the histogram-based methods.

The histogram-based color retrieval techniques described above suffer from a lack of important spatial knowledge. [39] discusses a technique of integrating color information with spatial knowledge to obtain an overall impression of the image. The technique involves three steps. In the first step, some heuristic rules are formulated to find the set of representative colors. Then this set of representative colors is used as the basis for obtaining relevant spatial information through a maximum entropy discretization process. Finally, the information obtained in step two is used to retrieve relevant images from an image database.

A color is selected if (1) it occupies a large percentage of the pixels in the image; or (2) it occupies a significant percentage of the pixels within a predefined “window” located at the center of the image. Two color histograms are computed. The first color histogram,  $H_g$ , represents the color composition of the entire image. The second color histogram,  $H_w$ , represents the color composition of the predefined central “window”. Both color histograms are then sorted according to the number of pixels per color. The color with the largest number of pixels in  $H_g$  is chosen as the background color of the image, and the color with the greatest number of pixels in  $H_w$  is chosen as the color of the object in the image. By doing this, the problem of having a relevant object color being hidden/distorted by the background color is avoided.

The experimental results show that this approach generally yields a better average precision for image retrieval and is more tolerant to noise in the image [39].

Various methods for comparing color similarity have been proposed. Each method has its own strengths and weaknesses. A color histogram is a high-dimensional feature vector—typically having greater than 100 dimensions—and the comparison of histograms is computationally intensive. They are best suited for representation of global color rather than local color regions, because of storage requirements and the large number of computations required

at query time. Other methods like color sets or color moments are reported better than color histograms on some applications, but further experiments must be conducted to test the robustness, accuracy, and efficiency of these methods[29].

It is clear that color plays a critical role in image similarities because of their perceptual importance and computational simpleness. It is also clear that color alone is not sufficient to characterize an image. Shortcomings in many of the existing color-based retrieval techniques include the inability to recognize similar objects with different color, the extreme sensitivity of the techniques to the scaling of an object, and the poor tolerance of noisy images. It is clear, therefore, that an object's color alone is not sufficient to determine its identity; in general, texture or geometric properties are needed to identify objects. Thus it is necessary to combine color indexing with texture and/or shape indexing methods.

### 2.2.2 Shape Similarity

Shape representation is an interesting problem that has attracted significant research, e.g., [6, 56, 41, 43]. There are two closely related problems:

- (a) How to measure the difference between two shapes, so that the difference corresponds to the visually perceived difference;
- (b) How to represent a single shape in a compact manner so it can be stored in the DBMS efficiently.

A shape representation must have several properties:

- It should be a metric. E.g., let  $A$  and  $B$  be two shape representations and  $d(A, B)$  be the measure of difference between them, then we should have

- $d(A, B) \geq 0$  for all  $A$  and  $B$ .
- $d(A, B) = 0$  if and only if  $A = B$ .
- $d(A, B) = d(B, A)$  for all  $A$  and  $B$ .
- $d(A, B) + d(B, C) \geq d(A, C)$  for all  $A, B$ , and  $C$ .



- It should be invariant under translation, rotation, and scale. This is to assure that shape similarity won't depend on the size of the images or the position of the object in the image. For example, the shape representation of a square should be the same after we rotate the object 45 degrees clockwise and/or make the side twice as long.
- It should be reasonably easy to compute. This must hold for the measure to be of practical use.
- Most important of all, it should match the human intuitive notions of shape resemblance. In other words, answers should be similar to those that a human might give.

Generally, there are three kinds of approaches used in shape representation:

1. Representation through 'landmarks'. For example, in order to match two faces, information about the eyes, nose, etc., is extracted manually or automatically. Thus, a shape is represented by a set of landmarks and their attributes (area, perimeter, relative position, etc). The distance between two images is the sum of the penalties for the differences of the landmarks [68].
2. Representation through numerical vectors, such as
  - Turning angle function [6].
  - Some coefficients of the 2-d Discrete Fourier Transform (DFT) [30].
  - The first few moments of inertia, as in [40] and QBIC.
  - Sign of curvature [70].
  - Chain codes [54].
3. Representation through a simpler shape, such as polygonalization and mathematical morphology. For example, in [43], the shape of an object is approximately represented by a group of rectangles. Similarity is based on the position and size of the most significant rectangles. The technique

can also be extended to apply to 3-D images. But it is not clear if the representation is invariant to rotation of the object.

An interesting comparison among various methods used to represent object shapes has been made on a large image database in [70]. According to the results presented, the most efficient method of shape representation for image retrieval seems to be the one based on turning angles.

Turning angle representation is proposed in [6]. Using this approach, the boundary of a simple polygon  $A$  is represented by the turning function  $\Theta_A(s)$ . The function  $\Theta_A(s)$  measures the angle of the counter-clock-wise tangent as a function of the arc-length  $s$ , measured from a reference point  $O$  on  $A$ 's boundary. Thus  $\Theta_A(0)$  is the angle  $v$  that the tangent at the reference point  $O$  makes with some reference orientation associated with the polygon (such as the  $x$ -axis).  $\Theta_A(s)$  keeps track of the turning that takes place, increasing with left-hand turns and decreasing with right-hand turns. Without loss of generality, it is assumed that the polygon is normalized so that the total perimeter length is 1; hence  $\Theta_A(s)$  is a function from  $[0, 1]$  to  $R$ . For a convex polygon  $A$ ,  $\Theta_A(s)$  is a monotone function, starting at some value  $v$  and increasing to  $v+2\pi$ . So  $\Theta_A(1) = \Theta_A(0) + 2\pi$ . An example can be found in Figure 2.8.

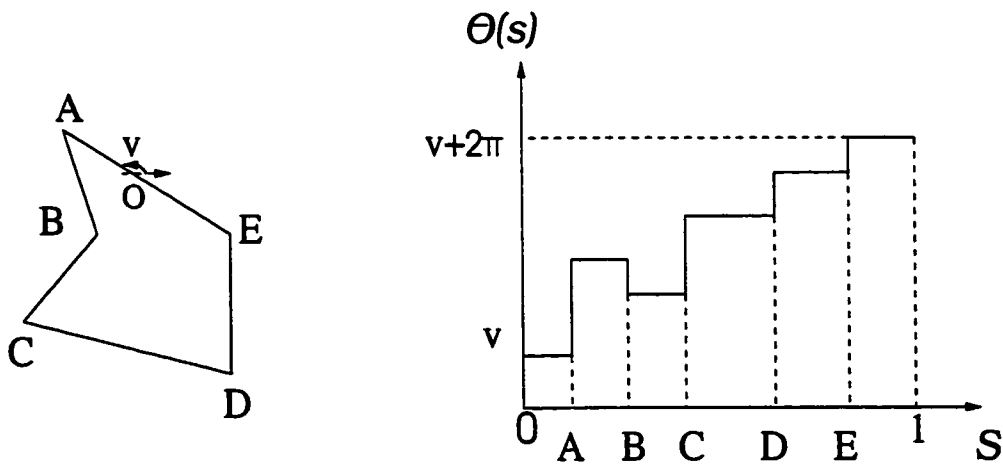


Figure 2.8: *Turning angle representation*

The distance function is defined as:

$$D(A, B) = \min_{r, \theta} \left( \sqrt{\sum_i (\Theta(A_i) - \Theta(B_i))^2} \right) \quad (2.19)$$

where  $\min_{r, \theta}$  represents the minimum of all the possible horizontal and vertical shifts of B. The complexity of this algorithm is in the order of  $O(n^2 * \log(n^2))$ .

Representation by turning angles has a number of advantages over other techniques:

- For polygons, the turning angle function is piecewise-constant, with jump points corresponding to the vertices.
- It is more or less independent of translation of objects and can be made invariant to rotations and scale factors. Rotation of a polygon A corresponds to a simple vertical shift of  $\Theta_A(s)$ . Changing the location of the origin  $O$  by an amount  $t \in [0, 1]$  along the perimeter of polygon A corresponds to a horizontal shift of  $\Theta_A(s)$ .
- It is also very natural, as it describes polygons in terms of their angles, a technique used by humans as well.
- Finally, it gives a compact description in the form of a unidimensional signal, which is much simpler to deal with than signals that have more dimensions.

Despite these advantages, the distance commonly used to measure the similarity between shapes represented by turning angles—the Euclidean distance—is generally too sensitive to small variations in the shapes. This is illustrated in Figure 2.9.

To overcome this difficulty, [41] proposes another distance function, called median distance function, on turning angle as follows:

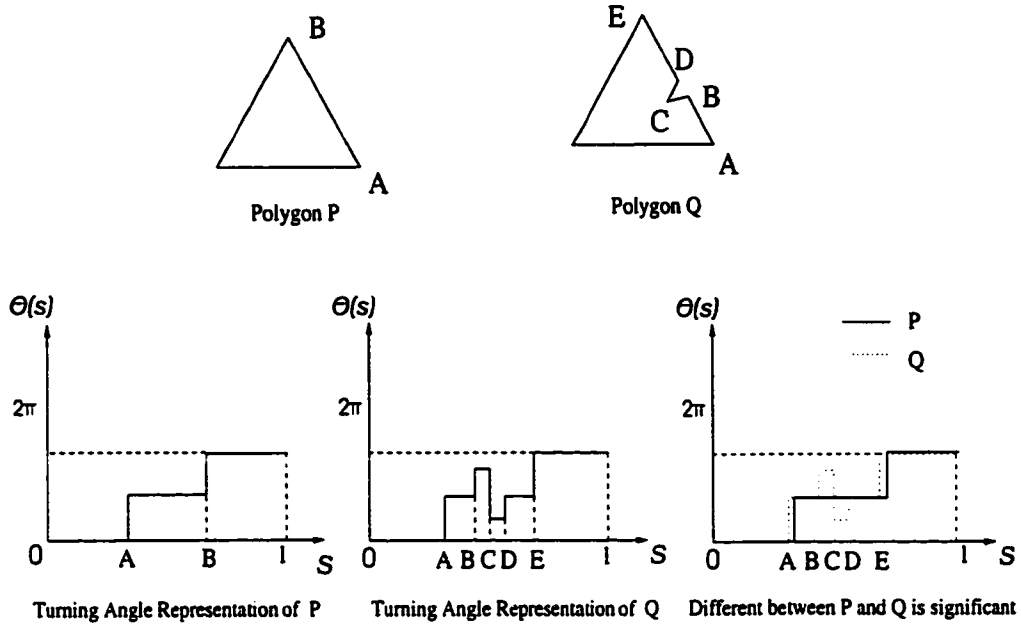


Figure 2.9: *Turning angle representation using Euclidean distance is sensitive to noise*

$$D_{median}(A, B) = \min_r ( | \Theta(A_i) - \Theta(B_i) | ) \quad (2.20)$$

where  $\min_r$  is the minimum of all the possible horizontal translations of B, and both A and B have been previously normalized:

$$\begin{cases} A = A - median(A); \\ B = B - median(B); \end{cases} \quad (2.21)$$

The purpose of this normalization is to eliminate the effect of any rotation A or B may have undergone. If polygon B is polygon A rotated by the angle  $\theta_1$ , then we have:

$$median(B) = median(A + \theta_1) = median(A) + \theta_1$$

and

$$\begin{cases} A = A - median(A); \\ B = B - median(A) - \theta_1; \end{cases} \quad (2.22)$$

So, the effect of rotations between the two polygons is eliminated after the relative median values are subtracted from both vectors A and B.

The experimental results [41] show that the median distance behaves better than Euclidean distance, giving values close to zero when the shapes are the

same and returning results very close to those which a human being would return, when the shapes are similar but not identical. The complexity of this algorithm is in the order of  $O(n^2 * \log(n))$ .

### 2.2.3 Texture Similarity

The precise definition of texture has been elusive. The notion of texture generally refers to the presence of a spatial pattern that has some properties of homogeneity [20]. In particular, the homogeneity cannot result from the presence of only a single color or intensity in the region but requires interaction of various intensities.

Since giving a verbal description of texture is not easy for naive users, queries over object texture is often done with the help of some sample textures. Towards the development of “query-by-texture”, it is necessary to find a meaningful measure of the similarity of textures (texture discrimination) and to develop a procedure for segmenting images based on textural content (texture segmentation). Furthermore, discriminant functions are needed to gauge the similarity between textured regions and the texture key used for searching.

In [20], in order to get texture classes as complete as possible in the process of determining the optimal texture discriminant function, the complete set of 112 Brodatz texture images were obtained. The Brodatz textures, scanned from the album published by photographer Brodatz [13], are well-known in the texture analysis field. The photographs in this album were not originally intended for scientific research, but rather for artistic use. They have, however, been widely used in the texture analysis literature.

A texture set approach is proposed in [72] to extract spatially localized texture information and to provide efficient indexing of texture regions. Texture regions are extracted by the following process:

1. conversion of color data to gray-level intensity;
2. orthogonal spatial-frequency decomposition of gray-scale image;
3. energy thresholding within each sub-band; and

4. operations to merge pixels of high spatial-frequency energy.

After filtering, the data in the sub-band are analyzed together to provide labels for the sufficiently large regions that produce characteristic patterns of energy distribution across the sub-bands. The pattern of energy distribution is represented using a binary texture set, whereby each element in the binary set corresponds to the presence of energy above the threshold within a particular sub-band. The binary texture sets, which label regions within the images, are used to retrieve images from the database based on texture content.

In QBIC [58], the texture features include coarseness, contrast, and directionality. The *coarseness* feature helps measure the scale of the texture, and is calculated using moving windows of different sizes. The *contrast* feature describes the vividness of the pattern, and is a function of the variance of the gray-level histogram. The *directionality* feature describes whether it is isotropic (like a smooth object). It is a measure of the “peakedness” of the distribution of gradient directions in the image.

It has been shown then that when texture is modeled sufficiently well it can be used to find certain items in the database. But not all objects and regions of interest can be characterized by texture processes.

## 2.2.4 Other Image Similarity Measures

A domain-independent framework for defining notions of similarity is presented in [44]. The framework contains three components: a pattern language  $P$ , a transformation rule language  $T$ , and a query language  $L$ . An expression in  $P$  specifies a set of data objects. To specify objects that match a pattern approximately, some transformation rules defined in the transformation language  $T$  are attached to patterns in  $P$ . An object  $A$  is considered to approximate an object  $B$  if  $B$  can be reduced to it by a sequence of transformations. A specific application of a transformation to an object has a cost, which is a measure of the distance between object  $A$  and object  $B$ . Finally, a query language  $L$  is obtained by wrapping logic or algebra around expressions built using  $P$  and  $T$ .

The framework can be tuned to the needs of a specific application domain by the choice of  $P$ ,  $T$ , and  $L$ . Consider shapes represented as a collection of black pixels, identified by location in a coordinate system. Assume that the pattern language  $P$  has a rectangular region of pixels as a basic unit, and union and difference as compositional operators. This allows defining expressions to describe shapes, and transformation rules to shift/scale objects along the  $x$  or  $y$  dimensions. The transformations, when embedded in the pattern language, permit portions of an object to be shifted/scaled in arbitrary ways, to create a variety of similar objects. With appropriate quantification on the permissible shift/scale parameters, one obtains a language for querying similar shapes.

But this approach has its limitations. For example, the transformation testing problem and membership testing is undecidable, *i.e.*, for a set of transformation rules  $T$ , a string  $s$ , and an expression  $e$  over  $T$ , the problem of testing whether  $s \in e$  is undecidable.

In [77], contextual similarity and spatial similarity are considered. Four levels of contextual similarity between two images  $M_1$  and  $M_2$  are defined:

- The images have exactly the same objects.
- Any object found in  $M_2$  exists in  $M_1$  ( $M_1$  includes  $M_2$ ).
- Any object found in  $M_1$  exists in  $M_2$  ( $M_2$  includes  $M_1$ ).
- There is at least one object common to both  $M_1$  and  $M_2$ .

Five levels of spatial similarity are also defined based on the spatial relation and distance among objects. As well, complex queries combining contextual and spatial constraints using logic ‘AND’ and ‘OR’ operators can be constructed.

A method for searching similar wavelet images is presented in [42]. The query image may be a hand-drawn sketch or a scan of the image to be retrieved. The coefficients of the Harr wavelet decompositions [11] are truncated, quantized and distilled into small signatures for each image, then an image querying metric that operates on these signatures is introduced. This metric essentially

compares how many significant wavelet coefficients the query has in common with potential targets. The resulting algorithm is simple, requiring very little storage overhead for the database of signatures, and is fast enough to be performed on a database of 20,000 images at interactive rates, as a query is sketched.

In [67], attributed relational graphs (ARG) are used to represent image content. Nodes with associated attributes represent objects in the image. The spatial relationship between two objects is recorded by the angle between the horizontal reference axis and the line connecting the centers of mass of the objects. Image similarity is decided based on both object properties and relationships among objects. The distance is measured by the cost of matching nodes and matching relationships among such nodes. This method relies on the assumption that a fixed number of labeled objects are common in all images of a given application domain in addition to a variable number of unlabeled objects.

## 2.2.5 Conclusions

With the increasing popularity of multimedia information systems, managing image data efficiently is becoming more important. It is obvious that image similarity measurement is fundamental to image databases. The measure must replicate as well as possible human similarity assessment. The major challenge is to find feature extraction functions that preserve the dissimilarity and distance among the objects as much as possible. At the same time, the interface should provide tools to support such similarity measurement. The user should be able to specify the criteria and other parameters for similarity selection query operation.

The meaning of similarity may vary depending on the application domain and even the purpose of the query. In many applications, we must expect to see several similarity criteria involved. Two images can be very similar in one context and very dissimilar in another. For example, two images picturing Bill Clinton would be considered similar by a newspaper editor but may not be by a fashion designer if s/he wants to find a image with the President dress in a



black suit. The availability of several similarity measures poses the problem of how to combine different similarity measures in a coherent and intuitive way. One simple way to do this is by successive refinements: create a first query with respect to one of the criteria, and order the whole database with respect to this. Then take the a portion of the result of the first query and order it with respect to the second similarity criteria, and so on.

# Chapter 3

## The DISIMA System

### 3.1 Objective

The retrieval of digital images is an active research subject, which promises to provide powerful new tools for database management in the near future. The DISIMA (DIStributed Image database MAnagement system) research project developed at the University of Alberta deals specifically with the development of technology to facilitate the management of, and access to, images using a database management system.

The major objectives of the DISIMA project are to model image data by using an object-oriented approach, and to support efficient content-based spatial query by using new indexing techniques. The model is sufficiently powerful to allow spatial and image data representations and indexing. Based on the model, a sophisticated query language is developed to allow complex queries involving image content and fuzziness.

This chapter introduces the data model, system architecture, and query language of the DISIMA system. A more detailed description can be found in [62] and [22].

### 3.2 Functionality

A distributed image database should provide the following functionality:

- A powerful data model which allows the representation of all types of information defined in the image, including raw data and symbolic data. Modeling of the structure of an image should be done in terms of abstract objects, independent of a particular presentation. The model must also be extendible, allowing the database to be extended and integrated with other multimedia information systems in the future.
- The image database should provide efficient and concurrent access to both an image and its associated symbolic data. A browsing and searching mechanism must be supplied to find information in the database. Meanwhile, must be able to manage and manipulate traditional alphanumeric data.
- The external components of an image database must provide several tools for modeling and manipulating itself, as the SQL language does for the definition and manipulation of classical data. These tools should include an image DDL and DML which offer several possibilities for image database querying and image operations: retrieval by content, query by the type of data or the links between data, conversion from and to different file formats, scaling, change of color depth, tiling, regions of interest, etc. The DDL and DML must be enhanced to allow the definition of more sophisticated data types and the execution of image processing functions upon a set of images retrieved by a query.
- A storage management system that stores different representations of the same image, including raw data, thumbnails, or symbolic data. These representations must be linked together so that the user can control the evolution of the system. It also needs to provide efficient storage, accessing, and sharing of image data among several sites and multiple users in a reliable and concurrent environment.

## 3.3 The DISIMA Model

A data model is defined as a collection of mathematically well-defined concepts expressing both static and dynamic properties of data intensive applications [14]. This section describes how DISIMA represents the content of an image and the model components. The model has similarities to VIMSYS [34] in its layered view of image data.

### 3.3.1 The Model Components

The DISIMA model aims at efficient representation of images to support a wide range of queries. Typical queries that DISIMA would support include the following:

- Find images that contain a given object  $o_1$  and satisfy  $P(o_1)$  where  $P$  presents query criteria specified on the object's properties (e.g., name, color, shape).
- Find images that contain objects  $o_1$  and  $o_2$ , and  $R(o_1, o_2)$  is true, where  $R$  expresses a spatial relationship (e.g., touch, contain, west, east).

These queries require information not only about images, but also the objects within images (called *salient objects*), and the spatial relationship between them. The DISIMA model addresses both image and spatial databases issues. It allows independence between image representations and applications, and distinguishes the identity of salient objects (*logical salient objects*) from their appearance in an image (*physical salient objects*). This allows the user to assign different semantics to an image component (semantic independence), and an image representation can be changed without any effect on applications using it (representation independence).

The DISIMA model, as depicted in Figure 3.1, is composed of two main blocks: the salient object block and the image block built on top of it. A block is defined as a functional first-level entity that can be broken down into several entities.

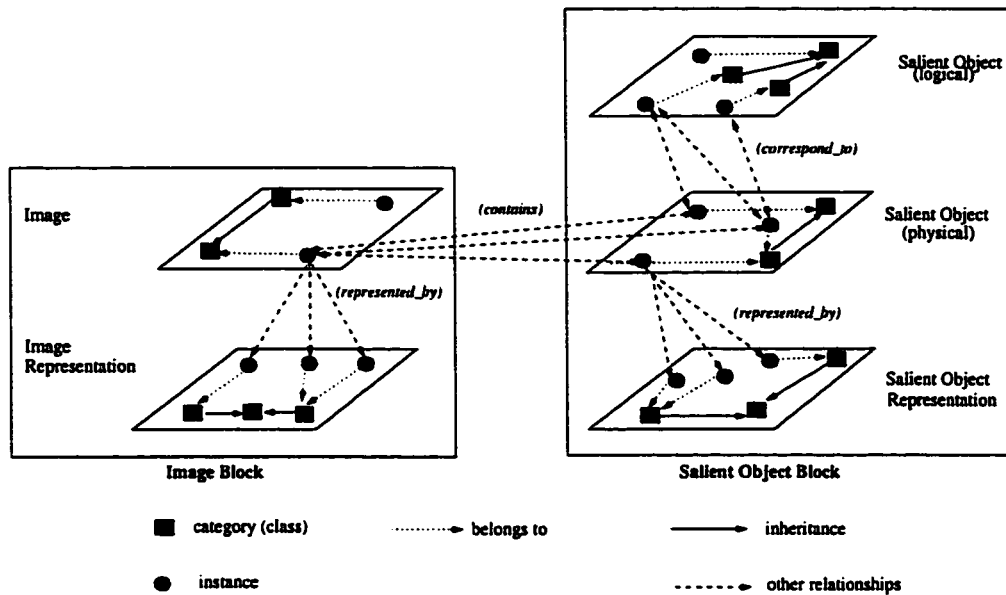


Figure 3.1: *DISIMA model overview*

### 3.3.2 The Image Block

#### The Image Block

An image is a basic unit in the DISIMA model.

**Definition 1** An image  $i$  is defined by a quadruple  $\langle i, R(i), C(i), D(i) \rangle$  where :

- $i$  is the unique (raw) image identifier;
- $R(i)$  represents the raw image;
- $C(i)$  is the content of  $i$ ;
- $D(i)$  is a set of descriptive alpha-numeric data associated with  $i$ .

The image block is made up of two layers: the *image* layer and the *image representation* layer. In order to accomplish *representation independence*, an image is distinguished from its representations. At the *image* layer, the user defines an image type classification similar to hierarchies in object type systems. A classification hierarchy for an application which manages news and medical images is illustrated in Figure 3.2. These images can be classified by the end users, according to specific criteria, during database population time. The *NewsImage* class is specialized by three classes: one for nature

images (*NatureImage*), another for images where a person has been identified (*PersonImage*), and the last one for the others (*Other Image*).

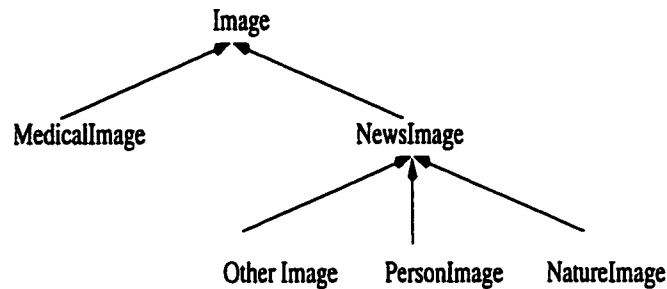


Figure 3.2: An example of image and logical salient object hierarchies.

Two main representation models: *raster* and *vector* are used in DISIMA. Raster representations are used mostly by image applications, while vector representation fits well with spatial applications. For raster representations, the JPEG format is used as default, and spaghetti modeling [49] is used for vector representations.

### 3.3.3 The Salient Object Block

In an image database system, images can be used in queries in the same way alpha-numeric data are used. However image data are too large to retrieve and manipulate in memory each time they are referenced; this can slow down the system performance. The common solution is to try to “understand” images and find a way to simply represent them, possibly by alpha-numeric data.

In DISIMA, the contents of an image are represented as a set of *salient objects* (i.e., interesting entities in the image) with certain spatial relationships to each other. The *salient object* block is designed to handle salient object organization. A simple example of a salient object hierarchy, corresponding to the image classes defined in Figure 3.2, is depicted in Figure 3.3. Several physical salient objects belonging to images may correspond to one logical salient object. Hence, a logical salient object does not have any representation, but a physical salient object may have one or more. Specific data members, such as posture and coordinates, can be added to a physical salient object. The logical salient object level models the semantics of the physical salient

object.

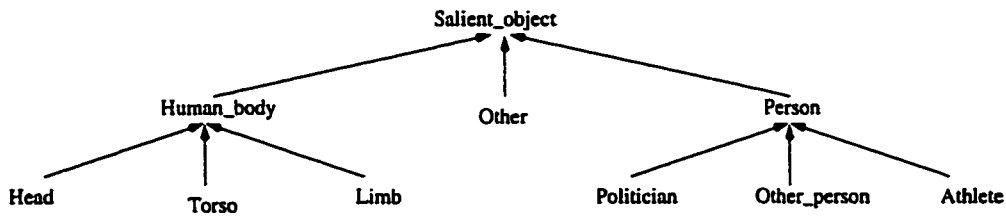


Figure 3.3: An example of salient objects hierarchy

For a given application, salient objects are known and can be defined. The definition of salient objects can lead to a type lattice as in Figure 3.3.

**Definition 2** Let  $SO$  be the set of all salient objects and  $\mathcal{SO}$  be the power set of  $SO$ . Furthermore, let  $RS$ ,  $CS$  and  $TS$  be the set of all representation, color, and texture objects, respectively. A representation object describes the spatial properties, such as shape and location of a logical salient object with respect to an image. A color representation describes color properties and texture representation describes texture properties of a logical salient object with respect to an image. The content of an image  $i$  is defined by a quadruple  $C(i) = \langle so, f, g, h \rangle$  where:

- $so \in \mathcal{SO}$ ;
- $f : so \rightarrow RS$ : maps each logical salient object to a representation object;
- $g : so \rightarrow CS$ : maps each logical salient object to a color object;
- $h : so \rightarrow TS$ : maps each logical salient object to a texture object.

A logical salient object is an abstraction of a salient object that is relevant to some application. For example Clinton may be created as instance of type *Politician* to represent President Clinton. Object Clinton is created and exists even if there is yet no image in the database in which President Clinton appears. This is called *logical salient object* and it maintains the generic information that might be stored about this object of interest (e.g., name, position, spouse)

Particular instances of this object may appear in specific images. There is a set of information (data and relationships) linked to the fact that “Clinton appears in image  $i_1$ ”. The data can be his posture, his location, and his shape in image  $i_1$ . Examples of relationships are spatial relationships with regards to other salient objects belonging to image  $i_1$ . A *physical salient object*, (P\_objectClinton\_1) linked to logical salient object objectClinton, is created to refer to image  $i_1$  and gives the additional information. Another physical salient object (P\_objectClinton\_2) will be created if President Clinton is found in another image  $i_2$ ,

Each physical salient object may have one or more representations (a minimum bounding box, a raster representation corresponding to a sub-image extracted from the image, or a vector representation extracted from the image after it has been segmented). Salient object relationships such as spatial relationships make sense only at the physical salient object level. A logical salient object has some functional relationships (is-a) with other logical salient objects. A logical salient object gives a meaning (a semantic) to a physical salient object. And several logical salient objects can correspond to a physical salient object. That is, depending on the application, the user can assign different semantics to the same physical salient object. Relationships between image, logical salient object (LogicalSalientObject), and physical salient object (PhysicalSalientObject) are summarized by an entity-relationship diagram in Figure 3.4.

**Definition 3** *A physical salient object belonging to an image  $\langle i, R(i), C(i), D(i) \rangle$  where  $C(i) = \langle so, f, g, h \rangle$  is defined by  $\langle x, f(x), g(x), h(x) \rangle$  and  $x \in so$ .*

The two levels of salient objects ensure the semantic independence and multi-representation of salient objects. Only the logical salient object level is used to answer the query “find all the images in which Clinton appears”. The query “find all the images in which Clinton is between Arafat and Netanyahu” requires the physical salient object level to check the spatial relationship among the three salient objects. The processing can indeed benefit from the existence of spatial indexes.



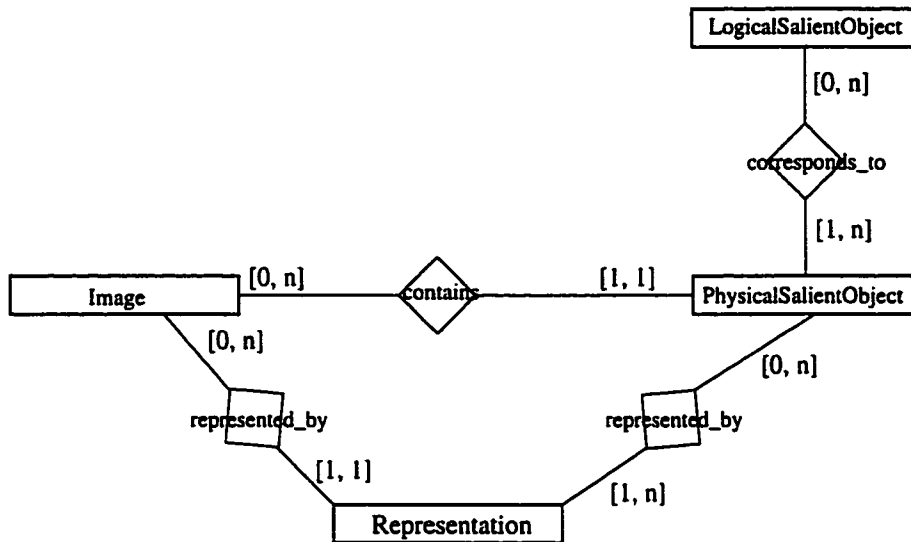


Figure 3.4: *Extended ER diagram for part of entities in the DISIMA model*

Similar to the images, the representation of salient objects is separated from their content information. The representation can be raster or vector. Raster representations of salient objects are useful to access parts of images, while vector representations can be used for spatial indexing and spatial relationships computation.

The distinction between the image block and the salient object block leads to more flexible and powerful modeling than having a fixed image hierarchy. Two salient objects belonging to the same image can have different data members. Let us take an image in which there is a well known politician shaking the hand of an athlete. For both of them we may want to know who and what they are. For the politician we may want to add the party s/he belongs to, while for the athlete we will be interested in the sports s/he practices. For this purpose, sub-classes (Athlete, Politician) were created in Figure 3.3.

The DISIMA model allows independence between image representations and applications (representation independence) and distinguishes the existence and identity of salient objects from their appearance in an Image (semantic independence). These independencies are achieved by introducing an abstraction between an image and its representation, and also between a salient object and its appearance in an image. For the management of different representa-

tions of both image and salient object, DISIMA provides a powerful version manager.

### 3.3.4 Defining a DISIMA schema

A DISIMA schema is made up of two sub-schemas: an image schema and a salient object schema.

**Definition 4** *Let ITG (Image Type Graph) be an image type hierarchy and STG (Salient Object Type graph) be a salient object type hierarchy. A DISIMA schema is defined by (ITG, STG).*

For some pure image applications in which there is no need for spatial searches, images and salient objects may not need to have vector representations. On the other hand, for some pure spatial applications there may not be any raster representations for either images or salient objects. In this case, an image object will, in fact, be a map which does not need any image processing function.

## 3.4 The DISIMA Architecture

This section first concentrates on the basic components of a DISIMA single site architecture (Figure 3.5) and then the distributed one.

The single site architecture is composed of the interface, the meta-data manager, the image and salient object manager, the image and spatial index manager, and the object index manager. The interface provides a visual query facility to define and search image data. The data definition language (DDL) used for the DISIMA project is C++ODL [17] and the query language is an extension of OQL. DISIMA API is a library of low-level functions that allows applications to access system services. DISIMA is built on top of object repositories (ObjectStore [48] for the current prototype). These object repositories may not have image and spatial indexes. And the object-oriented indexes they provide (if any) may not match DISIMA requirements. Moreover, the image and spatial index manager and the object index manager have

to dynamically integrate new indexes. The meta-data manager implements object-based meta-data that give information about images and salient objects. The salient object manager implements the DISIMA model and the index managers allow dynamic index management. Indexes include object, image, and spatial indexes.

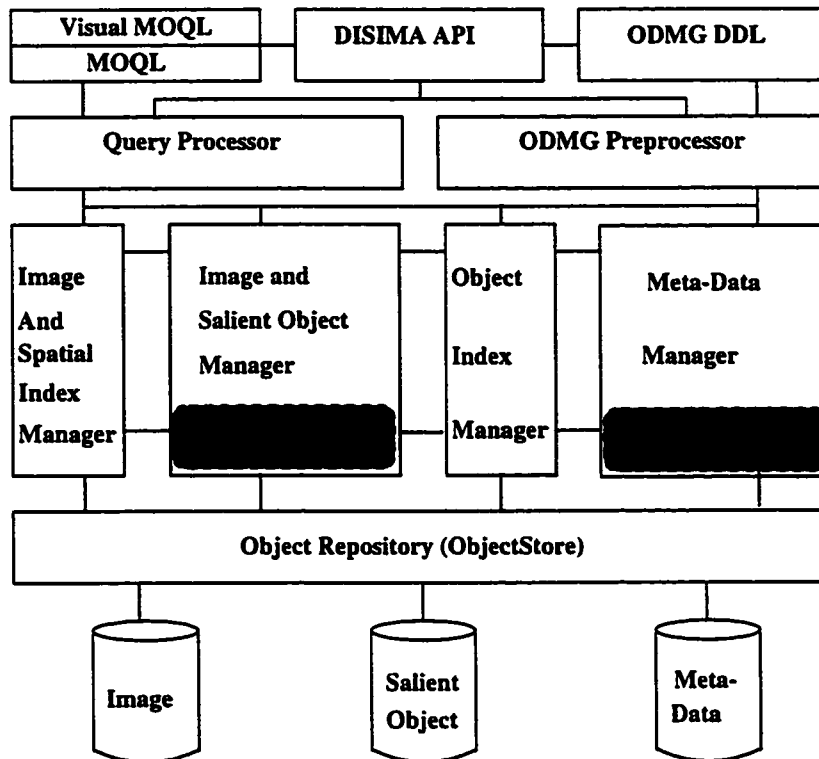


Figure 3.5: *The DISIMA architecture*

Meta-data is important in improving the availability and quality of the information to be delivered. It is a kind of on-line documentation. In some image applications [3], everything except the raw image data is included in meta-data. Based on object oriented concepts, the DISIMA model integrates the raw image and the alpha-numeric data linked to it. Meta-data in this context refers to meta-types and other types defined to keep track of all classes and behaviors (methods) defined by the user. The idea is to simulate, on top of ObjectStore, meta-types (i.e., types of types) that will be used by the user for schema definitions. DISIMA views the main components of the user-defined schema as objects that can be stored and queried using OQL.

As defined, the DISIMA meta-data is independent of any underlying object repository. Consequently, as everything is defined as an object, the same query language and query processor can be used to query both data and meta-data.

DISIMA provides an interoperable architecture to allow users to query multiple, and possibly remote, image sources. Due to the autonomy of each individual image source, DISIMA needs to have a wrapper built for each participating image data source. This wrapper which is responsible for transforming MOQL queries into local queries, executable at the individual site. When the individual image source is using a database (relational or object oriented) model, the transformation is simpler. However, translating content-based queries is not straightforward. When the image source is modeled as a file system or hyperlinks, the wrapper needs to provide search capabilities to support queries.

The interoperable architecture is designed using common facilities, as defined in the Object Management Architecture (OMA) [31]. CORBA provides transparencies at the platform and the communication levels. There remain two other levels: the database level where different data models can be found, and the semantic level to homogenize the meanings of the objects. The distribution framework is given in Figure 3.6. It involves homogeneous systems ( $S1$  talking to  $S2$ ) and heterogeneous systems ( $S1$  talking to  $S4$  or  $S3$ ). DISIMA builds wrappers that can transform MOQL queries into target queries executable at the target sources for the heterogeneous case.

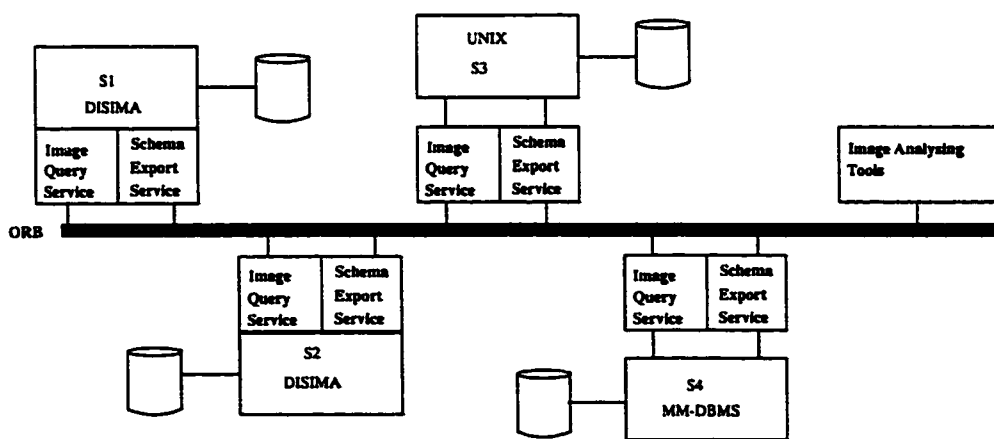


Figure 3.6: *Distribution in DISIMA*

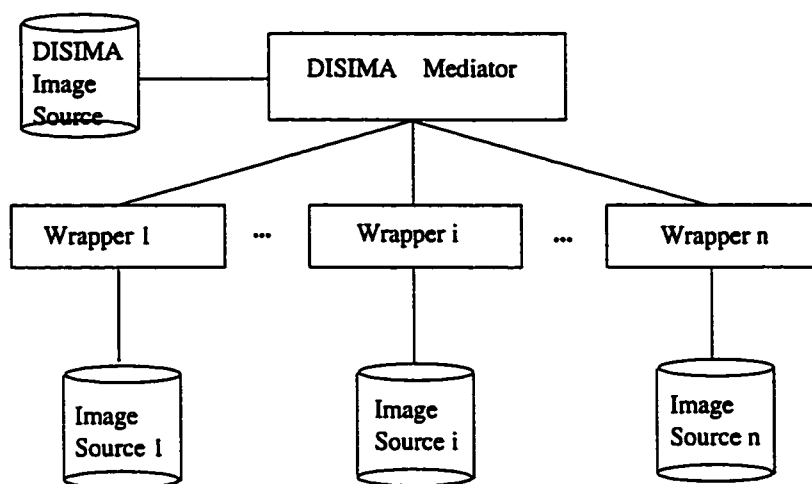


Figure 3.7: *Mediator-wrapper architecture*

The implementation of the *similar* predicate is application dependent. The current implementation of the *similar* predicate in the DISIMA project involves salient objects, color, and spatial relationships among salient objects. However, one of our objectives is to allow the user to set the *similar* predicate to the desirable checking (semantic checking, feature checking, and spatial relationship checking). The query processing benefits from the 2-*D-S*-tree [50] which is an index based on the 2 – *D* String approach [21]. For example, the processing of Query 4 in Section 3.5 will benefit from the use of the 2-*D-S*-tree [59].

DISIMA is being implemented on top of the ObjectStore [48] DBMS. The model, the architecture, and a complete query language, MOQL, have been defined. One of the research objectives is to study algebraic primitives which will support optimization of multimedia queries. In the current prototype, each algebraic operator is implemented in terms of ObjectStore functions. This establishes a link between the query processor and the underlying object repository, while enabling independent development of the query processor.

### 3.5 MOQL: A Multimedia Extension of OQL

In an image database system, users want to query images using not only conventional textual annotation, but also image content, which is hard to

define. The query language must be sufficiently sophisticated to allow fuzzy search, and support high-level notions and relationships. It has been shown [51] that the expressiveness of the traditional database query languages (e.g., SQL) is inadequate for multimedia data.

MOQL [51] is an object-oriented, general-purpose multimedia query language based on ODMG's Object Query Language (OQL) [17]. An OQL query is a function which returns an object whose type may be inferred from the operators contributing to the query expression. OQL allows users to query objects by using the object names as entry points into a database. The basic statement of OQL is:

```
select [distinct] projection_attributes
from query [ as identifier ] {, query [ as identifier ] }
[where query] [group by partition_attributes] [having query]
[order by sort_criterion {, sort_criterion}]
```

The language includes constructs to deal with spatial properties, temporal properties, and presentation properties. Most of the extensions introduced in OQL are in the *where* clause, in the form of four new predicate expressions:

- *spatial\_expression*

This expression includes spatial objects, spatial functions, and spatial predicates.

There are five spatial primitives (*point*, *line*, *circle*, *rectangle*, and *region*) in MOQL. Spatial relations can be categorized into *directional* and *topological* relations. Directional relations describe the direction from one point to another, such as east, top, northwest. Topological relations describe the relative position and location of contact among objects, such as separated, contains, connects, invades.

- *temporal\_expression*

This expression deals with temporal objects, functions, and predicates. It is not supported by the current version of VisualMOQL.

- *contains\_predicate*

This expression 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.

- *similarity\_predicate*.

This predicate checks if two images are similar with respect to some metric; such a metric can be based on salient objects, spatial relationships, colors, textures, or combinations of these. Different application domains may provide different implementation for these predicates.

In the following examples, some sample queries are given to demonstrate the use of MOQL to query image databases. The schema of the database against which these queries are specified is not defined here. The queries themselves are self-explanatory.

**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 President Clinton appears.

```
select  m
from    Images m, Politicians p
where   p.name = "Clinton"
        and m contains p
```

**Query 3** Find all images in which Clinton appears to be white.

```
select  m
from    Images m, Politicians p
where   p.name = "Clinton"
        and m contains p
        and p.color(m) = "white"
```

**Query 4** Find all images in which a Clinton is between Arafat and Netanyahu.

```
select    m
from      Images m, Politicians p1, Politicians p2 , Politicians p3
where     p1.name = "Clinton"
          and p2.name = "Arafat"
          and p3.name = "Netanyahu"
          and m contains p1
          and m contains p2
          and m contains p3
          and (    (p1.region(m) left p2.region(m)
                   and p1.region(m) right p3.region(m))
                or (p1.region(m) right p2.region(m)
                   and p1.region(m) left p3.region(m)))
```

**Query 5** Find all images that look like an image in which Clinton appears.

```
select    m1, m2
from      Images m1, Images m2, Politicians p
where     p.name = "Clinton"
          and m1 similar m2
          and m1 != m2
```

**Query 6** Select all the images in which there is a city within a 500km range of the point at longitude 60 and latitude 105 with populations in excess of 50000:

```
select    m
from      Images m, Cities c
where     m.name="Canada"
          and c.location inside circle(point(60,105), 500)
          and c.population>50000
```

DISIMA adopts MOQL as the query language for the following reasons:

- It is based on well-accepted SQL.
- It deals with spatial and temporal relationships.
- It is a general multimedia query language which can handle not only images, but also video data and multimedia documents.
- It is designed to be applied on general domains, rather than for a particular application.



# Chapter 4

## The User Interface

VisualMOQL is the visual query interface for the DISIMA project. It is based on the textual query language MOQL, and implements the image part of MOQL. A query specified using VisualMOQL is automatically translated into MOQL to make use of the MOQL parser and query processor. This chapter first describes the user interface in general, and then explains the semantics of VisualMOQL queries and query translation process. A comparison between VisualMOQL and some other image database query interfaces is also presented.

### 4.1 Introduction

Recently, with the growing popularity of the World Wide Web (Web) and other multimedia applications, there is a need for developing a user interface which is intuitive so naive users, such as private consumers or commercial business partners, can easily find and retrieve the large multimedia collections describing presented products and services. However, the traditional textual-based interface (e.g., SQL [26]) poses some disadvantages for image retrieval. First, it is not user friendly since users need to study database schema and query languages. Second, textual query languages do not visualize queries, so they do not allow users to match the query representation with their mental models [52]. Historically, several approaches have been taken to improve the user friendliness of database interfaces. Techniques include form-based (QBE [82]), diagrammatic (GUIDANCE [36], Gql [65]) and iconic (IconicBrowser

[80]). There have been attempts to develop a more natural interface using approaches such as cognition-based interfaces, that support query by image examples; and descriptive semantics-based approaches in which users pose queries by describing target image semantics using some keywords. These types of interfaces have the advantage of being natural to users, but have low precision, because queries are ambiguous.

In the last decade, visual query systems (VQS) have attracted much attention and have been gaining popularity. They greatly improve the effectiveness and user-friendliness of the human-computer interaction [9]. Because of this, they are oriented towards a wide spectrum of users, especially novices who have limited computer expertise and are generally ignorant of the inner structure of the accessed databases.

In order to best facilitate the human-computer interaction in an image retrieval system, it is important to first understand the tasks the users want to accomplish and then build a visual interface that is suitable for performing these tasks. In order to be successful, the visual interface should provide functionalities to allow the user to visualize queries. Generally speaking, an efficient visual interface for image retrieval should provide the following functionalities: [9, 52]

- A clear and concise representation of the organisation of the data stored in the database. The main purpose of adopting a visual representation in a query system is to communicate clearly to the user the information content of the databases, concentrating on essential features and omitting unnecessary details. This helps the user to grasp the internally structured information stored in the database easily without spending too much time learning it.
- A query paradigm that allows the user to naturally specify content-based queries. The construction of the visual query should be easy to use, and effective, so that users can express their visual queries using minimum steps; otherwise the user may as well learn a query language. Meanwhile, the interface has to be reasonably powerful, and it has to allow users to

visualize the process by which they are constructing a query.

- Algorithms and techniques for finding the best similarity measure according to the application context of the query.
- A way for users to express similarity measure, both on salient objects (e.g., their color, shape) and on the image as a whole. This is important because in the paradigm of content-based retrieval, images are not simply matched, but are ranked in order of their similarity to the query image.
- If a multimedia query language is defined and used for querying the database, then the visual interface should be able to render query language operators in terms of appropriate direct manipulation sequences.

## 4.2 VisualMOQL: The DISIMA Visual Query Interface

Since visual query languages are directed at naive users and often not based on a textual query language, they are not always as expressive as textual queries. One way to extend the capabilities of visual languages is to base them on powerful multimedia query languages, which themselves may be extensions of object or object-relational query languages. This provides a visual query language that enables easy querying of multimedia databases, while benefiting from the query facilities provided by the database management system.

Taking this approach, VisualMOQL provides advanced functionalities and high usability through strong emphasis on the expressiveness and intuitiveness of the user interface. It allows casual users to query an image database by expressing image semantics, which are based on the DISIMA model that views images as composed of salient objects with some properties. Several levels of refinement are offered, depending on the type of query and the level of precision users want.

In VisualMOQL, users start building visual queries by specifying the salient objects of interest. Then these queries can be refined by adding salient objects'

color, shape, and spatial relationships. Tools for defining image properties are also provided. All the query criteria can be grouped by using logical conditions to obtain a more complicated visual query. Finally, users can filter the resulting images by restricting the number of images to be returned and/or the minimum similarity value between the query image and the images returned.

#### 4.2.1 Interface Overview

The VisualMOQL main interface window (Figure 4.1) consists of the following components:

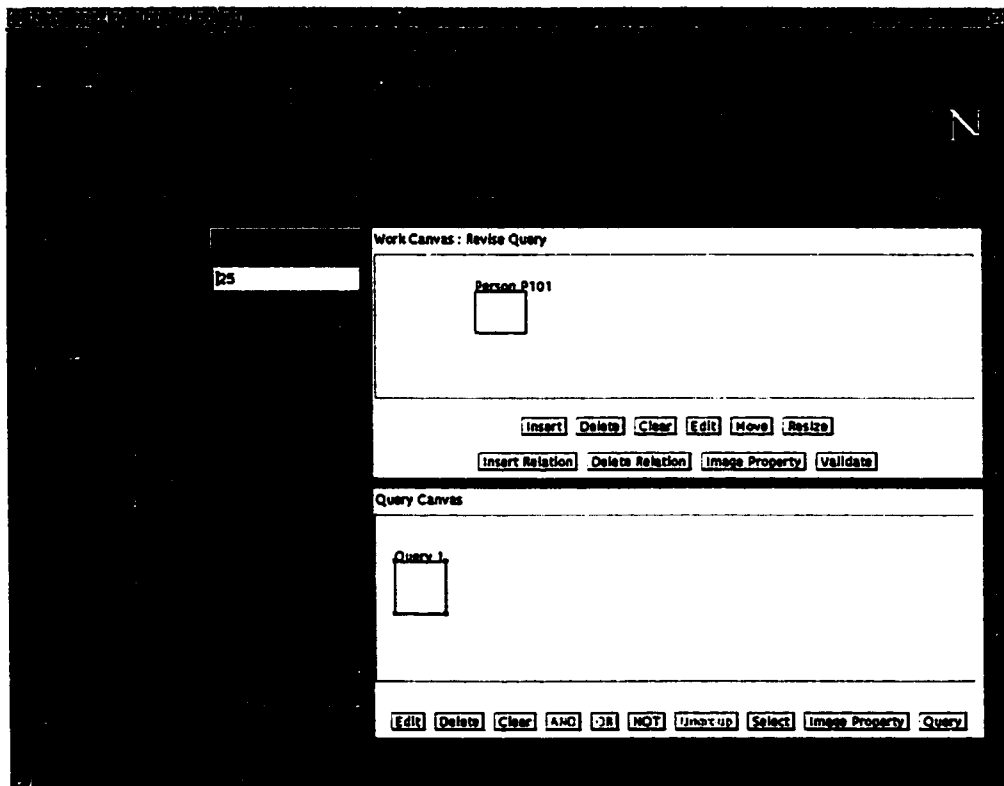


Figure 4.1: *Main window of VisualMOQL*

- An image class selector, which is implemented using a drop-down selection list.

Images stored in DISIMA are categorized into system-defined classes during database population time. The database administrator can cus-

customize the categories according to particular application domains. During query construction, users will select the image class that they want to query over. This step reduces the search space dramatically and speeds up the entire query process.

Although the image classes are defined and organized hierarchically in the database, the parent–children relationship is not illustrated in the current implementation of VisualMOQL. Because usually the number of image classes defined is relatively small, a simple drop–down selection list would provide the same functionality a complicated navigation browser does, but is much simpler and easier to use. This helps diminish the complexity of the query interface.

- A text field where users specify the maximum number of images that should be included in the query result. A scroll bar is not chosen because the value cannot be predetermined, so a scroll bar cannot provide the flexibility that is needed.

This number is not translated into MOQL, but passed to the query engine with the MOQL query, as a separate parameter used by the query result presentation interface.

- A horizontal scroll bar to let users specify the minimum global similarity between query image and images retrieved from the database.

This is not translated into MOQL, but passed to the query engine together with the MOQL query. It is a quality of service parameter used by the query result presentation interface.

- A salient object class browser that allows users to choose any predefined logical salient objects (LSO) that are of interest. This component is explained in more detail later.
- A working canvas where users can construct and modify simple queries. A detailed description is given in the next subsection.

- A query canvas where users can build compound queries based on simple queries, using logical operators — It is also described in more detail in the next section

### 4.2.2 Salient Object Class Browser

In DISIMA, image semantics are based on logical salient objects and their properties. These are identified and saved during the database population time, using the DISIMA image entry interface. Objects are organized into a salient object hierarchy, and the root is *Logical Salient Object (LSO)*.

The salient object browser, referred to as “browser” from now on, allows users to navigate through the hierarchy defined in DISIMA’s object block, and select the objects that they want. The first component in the browser is a drop-down list called parent class selector. Below it are three selection lists. Arranged from left to right, they are: class list, subclass list, and property list.

When the interface is first invoked, *LSO* is selected automatically by the parent class selector as the default active parent class. All its children classes are displayed in the class list. The user can search for the desired object by browsing the list with help of a vertical scroll bar. Once an object class is selected, it is highlighted in the class list and its subclasses and properties, including those that are inherited, will be shown in the subclass list and property list, respectively.

To traverse down the hierarchy tree, users can double click on any salient object in the class list. This will add the object to the parent class selector and set it as the active parent class. The object’s subclasses will then be displayed in the class list immediately. To travel up the tree, users need to click on the parent class selector and choose a selection from the drop-down list. This will update the class list as well.

VisualMOQL chooses this browser-oriented approach for a couple of reasons. First, it is a natural choice for representing the object-oriented salient object block in the data model used by the DISIMA system. Second, it eliminates the burden on users of being required to learn the internal database schema. Using this browser, the user can select any salient objects defined

in the salient object hierarchy. In addition, the browser can present three levels of the hierarchy tree at a time. So, even with little knowledge of the database structure, casual users can locate the desired object class in the tree very quickly. Lastly, the browser provides future extensibility. With more and more images being inserted into the database, the object class hierarchy tree can grow rapidly. Other facilities, like list, simple categorizing, or keyword searching, will not be able to provide the capability to support efficient navigation of the object hierarchy.

### 4.2.3 Working Canvas

In VisualMOQL, there are two kinds of visual queries: simple and compound. We concentrate on simple query in this subsection and will discuss compound query in the next subsection.

All VisualMOQL visual queries are built based on simple query, in which users select multiple salient objects, specify their properties and spatial relationships, and define image properties. A simple query can be translated into a valid MOQL query.

The working canvas is where users construct simple queries using the semantic and textual approach, referred to as query-by-drawing [18]. In this approach, users first select an object from the salient object class browser's class list. *LSO*, a system-defined class, is used as the default if nothing is specified. Then users click the *insert* button to introduce the object into the working canvas. The object is represented by its minimum bounding box (MBB), which is a rectangle, and a short string label, which is used as the name of the object. In comparison to representing objects by bullets, MBBs are more visible so it is easier for users to select them. Moreover, MBBs can be used to determine not only directional relationships but also topological relationships, as we will demonstrate later in this subsection. Object names are generated automatically and are different for each object in the visual query. They are used not only to help users distinguish objects, but also to provide VisualMOQL with the equivalent expressiveness of MOQL. This will be illustrated when we discuss object property specification.

Salient object property is another important component of image semantics defined in DISIMA and MOQL. In order to build a fully integrated image database system, an object property input window, shown in Figure 4.2, is provided. It is used to define the color, shape, and other textual properties of salient objects. To invoke this window, users first click the *edit* button then select the target object from the canvas.

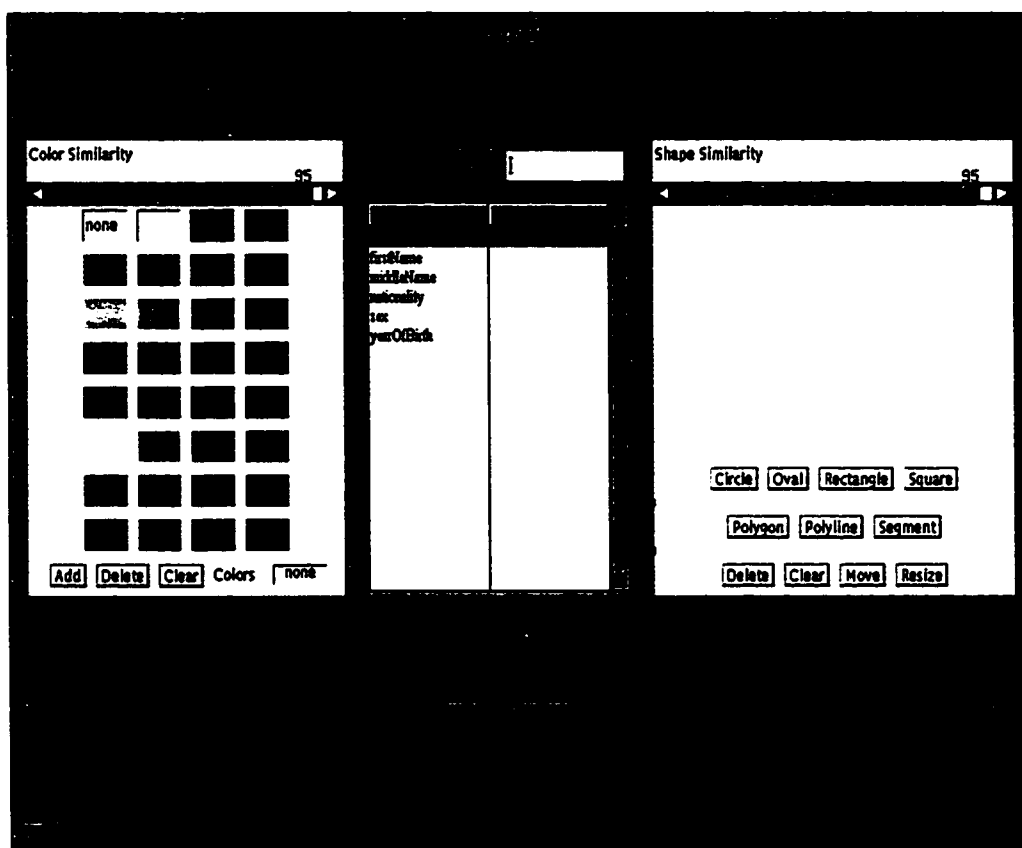


Figure 4.2: *Object property input window for object Person.*

The object property input window is further divided into three major components:

- A color panel where users can select colors from a predefined color template. Color plays a critical role in image similarity because of its perceptual importance and computational simplicity. A visual query interface must be equipped with tools to allow users to specify color similarity as part of the local image matching. In VisualMOQL, users can choose



colors from the templates and assign them to salient objects. A salient object may consist of multiple colors; for example, the national flag of France has three colors. In order to reflect this in the visual query specification, VisualMOQL supports multiple color assignment to a single object, i.e., one object can be red and green instead of being either red or green.

- An object property input panel. Every salient object stored in the database has some properties defined and values specified, such as the name and age of a *Person*. During query construction, users can type in alphanumeric values of salient object properties.

Since each object in the query has a unique name, and is displayed as part of the object icons in the working canvas, users can use it to express join operators on object properties. For example, “find images with 2 persons who have the same last name” can be expressed in VisualMOQL by following these steps: First, two salient objects of type *Person* are introduced in the working canvas (say, *P101* and *P102*). Then in the property input panel of *P101*, the value of its last name is specified as *P102.name*.

- The third component is the shape input panel. Geometric information is another important visual property used in image matching. VisualMOQL supports query specifications using four simple types of shape: circle, oval, square, and rectangle. Users can combine these simple shapes to form more complicated irregular shapes.

As explained in Chapter 2, color and shape searching are based on similarity, instead of exact match. Therefore, horizontal scroll bars are added to the color and shape input panels so that users can specify the similarity thresholds for them. Unlike the global image similarity threshold, these two similarity values are part of the visual query primitives, and are included in the translated MOQL query.

Spatial relationships between objects are critical for many image database

applications such as medical imaging systems and geographical information systems. The DISIMA system aims at developing a method to capture the behavior and characteristics, not only of images, but also of spatial applications. Spatial relationships can be categorized into topological and directional relationships. In MOQL, eight directional operators are defined: East, Northeast, North, Northwest, West, Southwest, South, and Southeast. Directional relationships are defined explicitly through a dialog box shown in Figure 4.3. Users specify which axes (x-axis and/or y-axis) matter by toggling the corresponding boxes. Currently, query conditions specified on the z-axis, and distance, are not translated by VisualMOQL since MOQL does not yet provide operators on these relationships.

Figure 4.4 shows some typical spatial arrangements of two objects in the working canvas, and their MOQL translations.



Figure 4.3: *Dialog window for defining directional spatial relationship*

To calculate the directional relationships among objects, the centroids of the object MMBs are used. When compared to using the edges of MBBs, this approach is much simpler. It is necessary to compare the locations of only two points, in contrast to eight, if the edges of MBBs are used.

Topological relationship makes up the other half of spatial relationships.

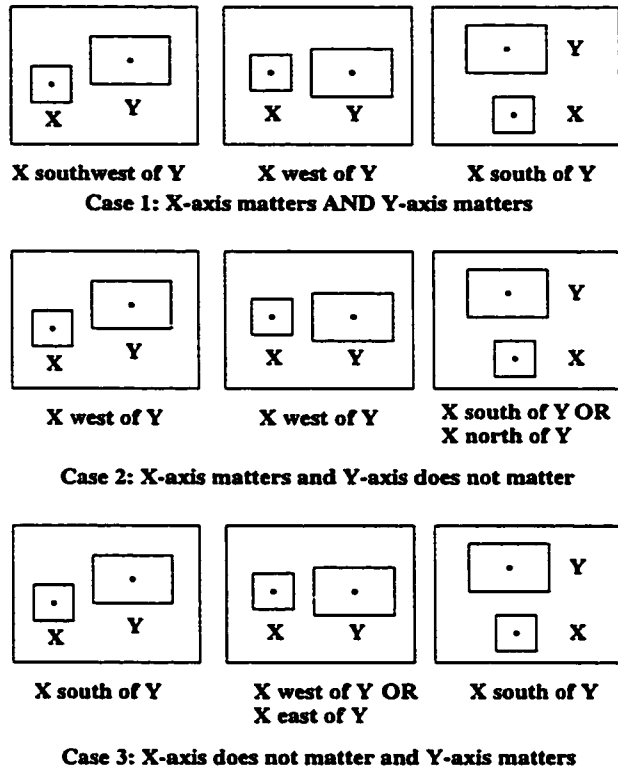


Figure 4.4: *Definitions of directional relations*

Eight fundamental topological relations are considered [27]: *inside*, *cover*, *touch*, *overlap*, *disjoint*, *equal*, *coveredBy*, and *contains*. However, two pairs of predicates are inverses: *cover* vs. *coveredBy* and *inside* vs. *contains*. Figure 4.5 shows the six basic topological relations.

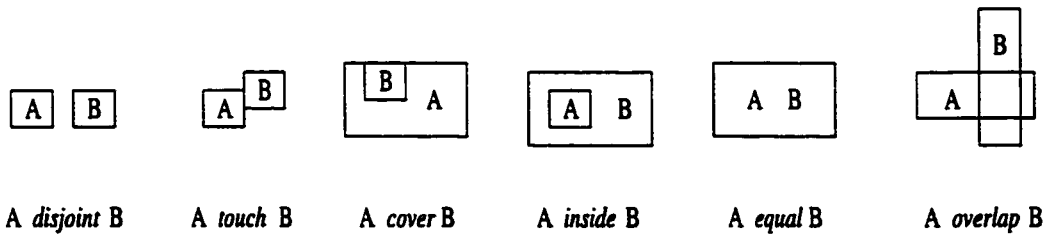


Figure 4.5: *Definitions of topological relations*

From Figure 4.5, we can see that the topological relationships are defined based on object's MBBs, not their centroids. If two objects' MBBs are intersected then the topological relationship is automatically detected and inserted into the visual query. This helps to reduce the number of errors caused by hu-

man mistakes. To remove a topological relationship, users just need to move the MBBs apart.

All query conditions defined in a simple query are conjunctive, i.e., they are *ANDed* in the final query translation. It is difficult to combine both conjunctive and disjunctive conditions in the same canvas, let alone building nested queries. To solve this problem, a second canvas, called a query canvas, is introduced into the interface. So a VisualMOQL query construction is done in two places. In the working canvas, users concentrate on low level details and local matching criteria such as salient objects and their color, shape, alphanumeric property, and spatial relationships. All these criteria are represented by conjunctive predicates in simple queries. After the simple query is moved into query canvas, it can be combined with other queries by using disjunctive, conjunctive, or negate operators to form more complicated compound queries. To move a simple query from the working canvas to the query canvas, users click on the *validate* button. This procedure is referred to as validating the simple query.

In addition to all we have mentioned so far, image semantics also include image properties. VisualMOQL implements an image property dialog window (Figure 4.6) which can be invoked by clicking the *Image Property* button in the working canvas. It is very similar to the object property dialog window.

#### 4.2.4 Query Canvas

Before discussing the design of the query canvas, we need to define a compound query. In VisualMOQL, a compound query is one of the following:

1.  $q_1$  AND  $q_2$  where  $q_1$  and  $q_2$  are simple or compound queries;
2.  $q_1$  OR  $q_2$  where  $q_1$  and  $q_2$  are simple or compound queries;
3. NOT  $q_1$  (NOT  $q_2$ ) where  $q_1$  ( $q_2$ ) is a simple or compound query.

The query canvas is where users build compound image queries. To do this, users first select the logical operators by clicking the corresponding button;

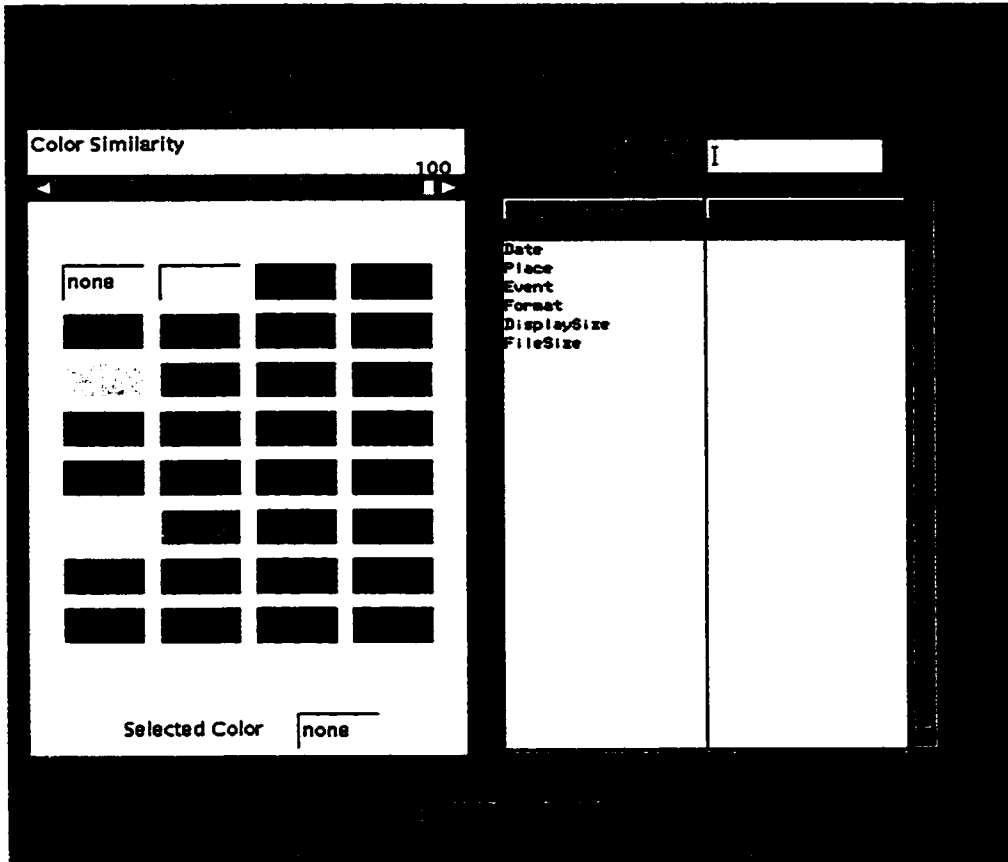


Figure 4.6: *Dialog window for defining image property*

they then choose the icons of the queries that they want to connect. If the operator is *AND* or *OR*, then icons of the two queries involved are grouped together inside a bracket with the operator displayed between them. If the new compound query is a negation with only one query involved, then the query icon is displayed following the *NOT* operator.

In VisualMOQL's two-phase query construction process, details of the simple queries are hidden from users in the query canvas. Thus, each validated simple query is represented by just a small square box, and a string which is used as the name of the simple query.

A visual query interface should have the flexibility to allow users to change queries at any time. In VisualMOQL, the task of query modification is carried out in both the working and query canvases. In the query canvas, the user can build and ungroup compound queries. If they want to modify the content of a validated simple query, they can click the *edit* button to bring the query canvas into *edit* mode, and then select a simple query to load its content into the working canvas. From there, they can add or remove salient objects, change query criteria, and revalidate the simple query.

## 4.3 VisualMOQL Query Semantics and Translation

### 4.3.1 VisualMOQL Query Semantics

VisualMOQL has a well-defined semantics based on object calculus [63]. A query at the working canvas level is viewed as a sub-query in the query canvas. If we view the images and the salient objects classes as complex value relations [1], a simple query from the query canvas is, in fact, a formula without any free variable. The general framework of such a formula is:  $(\exists m \in M \exists s_1 \in S_1 \dots \exists s_k \in S_k, cond(s_1, \dots, s_k) \wedge cond(m) \wedge m \text{ contains } s_1 \wedge \dots \wedge m \text{ contains } s_k)$  where  $M$  is an image class,  $S_1 \dots S_k$  are salient object classes,  $cond(s_1, \dots, s_k)$  expresses boolean conditions on salient objects,  $cond(m)$  expresses conditions on the image, and  $m$  contains the salient object  $s_i$ . There is only one image class in a simple query. The default image class is *Image*, the root of the

image class hierarchy, but this can be changed. As a formula without any free variables, a simple query is evaluated to *true* or *false* [63].

Since simple queries in the query canvas could be based on different image classes, not all of them can be grouped by the AND operator. Consider these two simple queries:

- Find natural images that contain a beach.
- Find people images that contain Bill Clinton.

*AND* cannot be applied to them because natural image and people image are not compatible, i.e., neither one is an ancestor or a descent of the other in the image class hierarchy tree. This rule also applies to compound queries. The image class of compound queries is calculated as follows: Assume we have two queries, *A* and *B*, and they are based on image classes  $M_a$  and  $M_b$ , then

- In *A AND B*,  $M_a$  and  $M_b$  must be compatible. If  $M_a$  is the ancestor of  $M_b$  in the image class hierarchy tree, then  $M_b$  will be set as the image class of *A AND B*.
- In *A OR B*,  $M_a$  and  $M_b$  may or may not be compatible. The image class of the resulting query will be the least common ancestor of  $M_a$  and  $M_b$  in the image type system. Since the DISIMA image type system is rooted, the common ancestor will be the image root class, in the worst case.

If users try to *AND* two queries that don't have compatible image classes, a warning window will pop up (Figure 4.7).

### 4.3.2 VisualMOQL Query Translation and Processing

To start searching the database, users select one and only one query, either compound or simple, from the query canvas and click on the "Query" button. With this approach, users can construct several similar queries at one time and compare the results, without deleting any queries and reconstructing them from scratch.

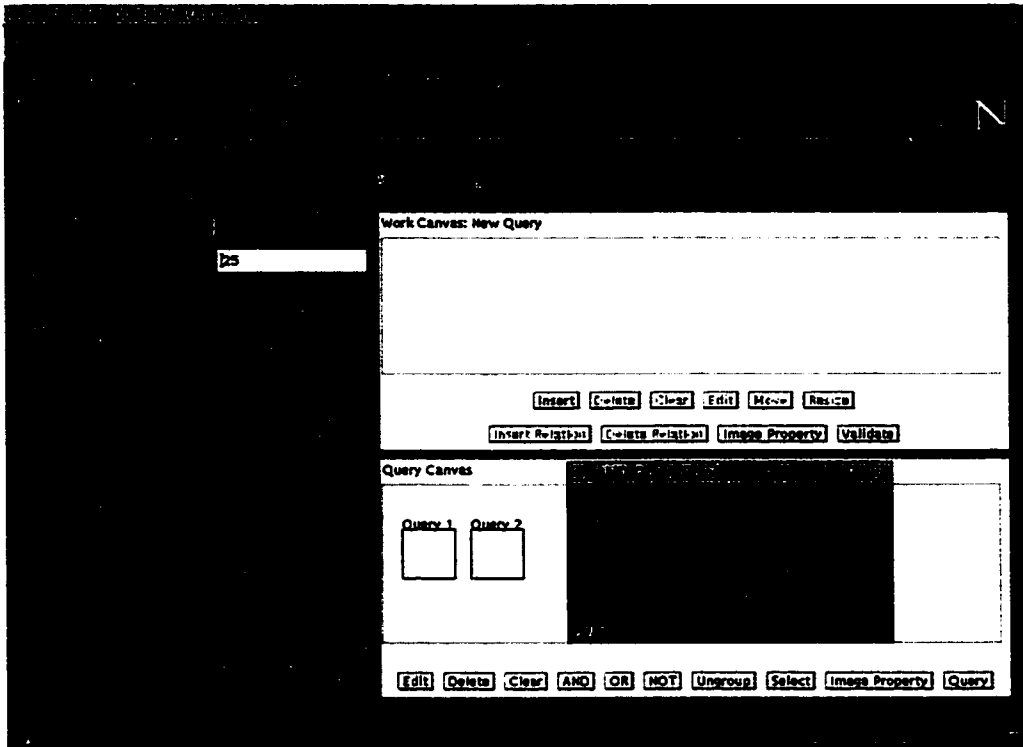


Figure 4.7: *Warning box appears when users try to AND queries containing incompatible image classes*

After users issue the query command, the system translates the Visual-MOQL queries to MOQL query strings. This is performed automatically by the MOQL generator so that users are not required to remember the complex syntax of MOQL. The pseudo-code of the translation algorithm is given in Appendix A.

After the equivalent MOQL query is generated, it is displayed by a separate pop-up window. If the user does not specify any visual queries or image properties before s/he clicks on the *Query* button, then a dialog window will be brought to the user to let her/him type in the MOQL query directly. For users who understand MOQL, and do not want to go through the visual construction process, this gives them the flexibility they want.

After all query criteria have been translated into appropriate formatted MOQL expressions, the MOQL string is passed to the search engine along with other two important parameters: the minimum similarity between the query image and resulting images, and the maximum number of images that



should be returned.

The result of query execution is a list of image thumbnails, and a URL linked to the real images ranked by their degree of similarity to the query image. These are presented to users in the same web browser in which they pose the visual query.

## 4.4 VisualMOQL and Other Content-Based Image Retrieval Systems

In this section, we compare VisualMOQL with some other content-based image retrieval systems.

QBIC, Virage, VisualSeek, and Photobook are some of the earliest content-based image retrieval systems. Their approaches have been explained in Chapter 2. They all use media-based search technology and support image matching on some image properties, such as color, texture, and shape. VisualSeek can also compare images based on spatial locations of salient image regions. But unlike VisualMOQL, these other systems do not capture image semantics on the salient object level and they do not support queries using a combination of textual description and visual examples to specify image semantics. Furthermore, VisualMOQL is based on a well-defined multimedia query language MOQL, and thus has enhanced capabilities.

The IFQ query interface [52] used by SEMCOG [53] is very similar to VisualMOQL. It too translates the visual query to a SQL-like query language called CSQL. The main difference between VisualMOQL and IFQ is that in IFQ, users do not have access to the database schema, so they can't assign values to object attributes. SEMCOG integrates a *facilitator* in charge of query reformulation. For example, the facilitator will consult the *Terminology Manager*, which is a dictionary, to replace "man" in the query by "person" if necessary. While in VisualMOQL users start a query by browsing the schema to find the salient objects and image class they are interested in, in IFQ users have to specify the spatial relationships between objects explicitly; but in VisualMOQL the relationships are determined automatically. From the users'

point of view, VisualMOQL is more user-friendly and from the implementation point of view, the VisualMOQL translator needs less work to translate user queries.

In IFQ, the CSQL translation is displayed in a panel just below the working canvas, where the visual query is constructed. At the same time as users change the visual query, the CSQL translation gets updated, so users can see it right after they make the changes.

VisualMOQL cannot support this feature because, unlike IFQ, which allows users to build only simple queries, VisualMOQL supports both simple queries and compound queries. During the construction stage, it is hard to find which query the user is interested in, and to display its translation. Another reason is that, since visual query interfaces are targeted at naive users, the query translation process should be transparent to them, and the query interface should be kept as simple as possible. Otherwise, the user may be overwhelmed by all the information provided by the interface at one time.

In conclusion, VisualMOQL is a declarative visual query language with a step-by-step construction of queries, close to the way people think in natural languages, and is based on a clearly defined semantics using an object calculus. This feature can be the basis of a theoretical study of the language. VisualMOQL combines semantic-based (query image semantics using salient objects) and attribute-based (specify and compare attribute values) approaches for image retrieval.

# Chapter 5

## Implementation Issues

In this chapter we describe some implementation issues for the user interface. VisualMOQL is implemented as a Java applet using Sun Java Development Kit (JDK) release 1.0.2 on a Unix Solaris 2.5 platform. The interface can be accessed by any Web browser (e.g., Netscape, Internet Explorer) anywhere.

### 5.1 VisualMOQL Translation

After users issue the query command, the system translates the VisualMOQL queries to MOQL query strings automatically. For compound queries, the first step is to organize the query into a query tree with all simple queries stored in leaf nodes, and logical operators in non-leaf nodes. The next step is to reduce the complexity of negated compound queries by transforming them into a normalized form so that the child of each negation is an existentially quantified formula (simple query). For example, if a query reads *NOT (A OR B)* where *A*, *B* are either compound or simple queries, it will first be transformed into *(NOT A) AND (NOT B)* before being passed to the visual query translation engine. This process is done from the root to the leaf in the query tree to ensure that at the end all negation operators are associated only with simple queries. This will simplify the translation process and make the final MOQL string more readable to system developers. Figure 5.1 shows the query tree of *(NOT (Query1 AND Query2)) OR Query3* and Figure 5.2 shows the same tree in the normalized form. The corresponding VisualMOQL screenshots are shown in Figure 5.3 and Figure 5.4.

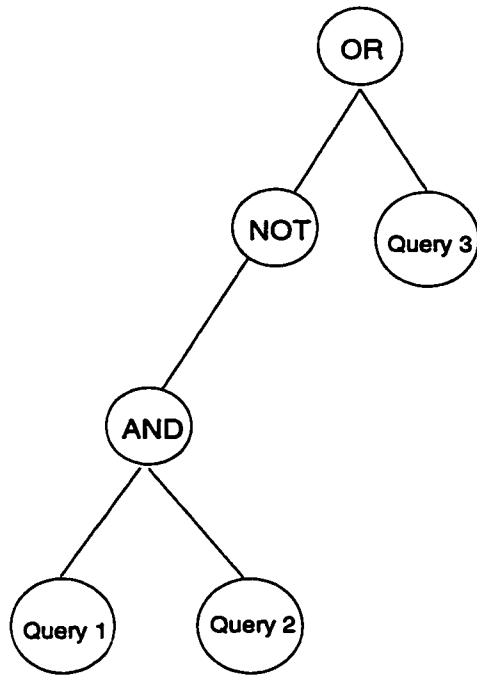


Figure 5.1: *The query tree of a compound query containing negator operator*

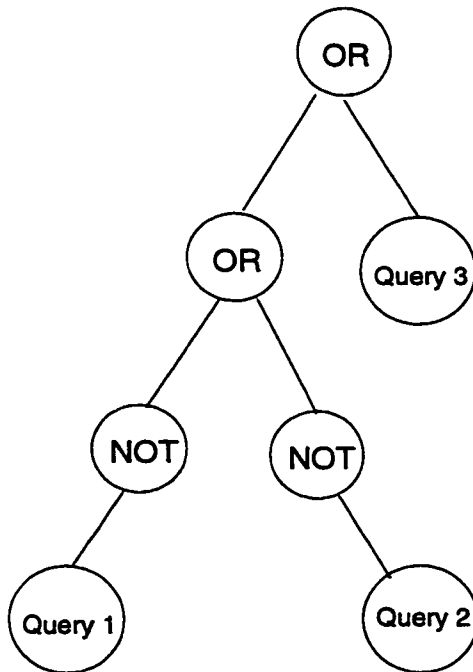


Figure 5.2: *The query tree of a compound query with negator in normalized form*

In the current implementation, the normalization is performed on the Java object that represents the VisualMOQL query directly. So if the user clicks the

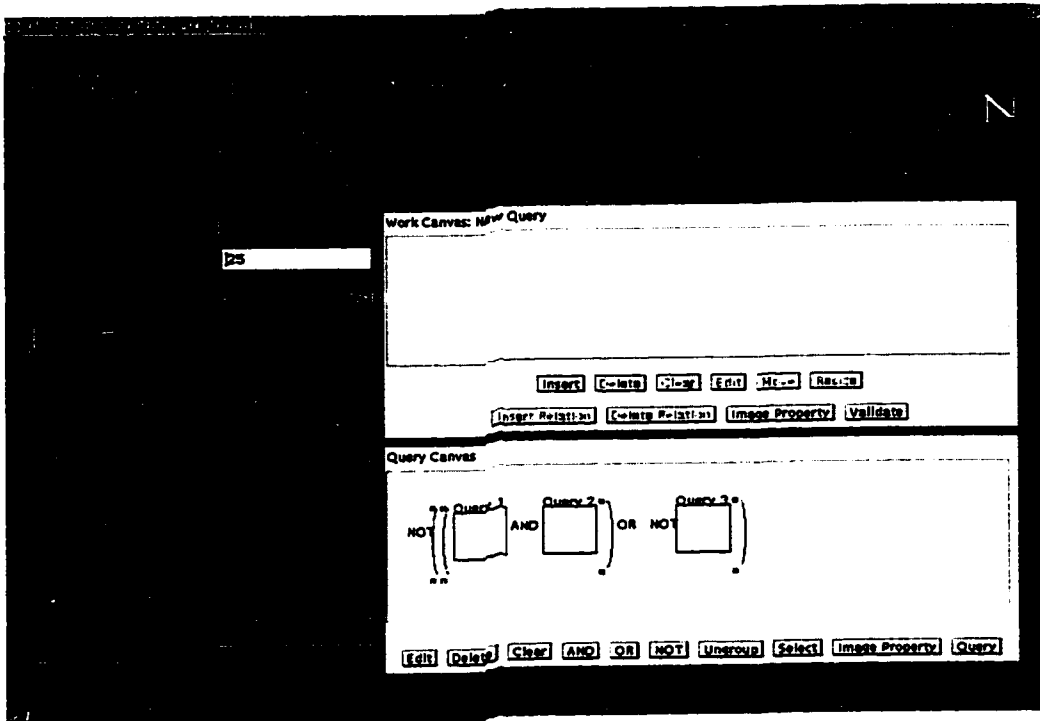


Figure 5.3: A VisualMOQL compound query with NOT operator

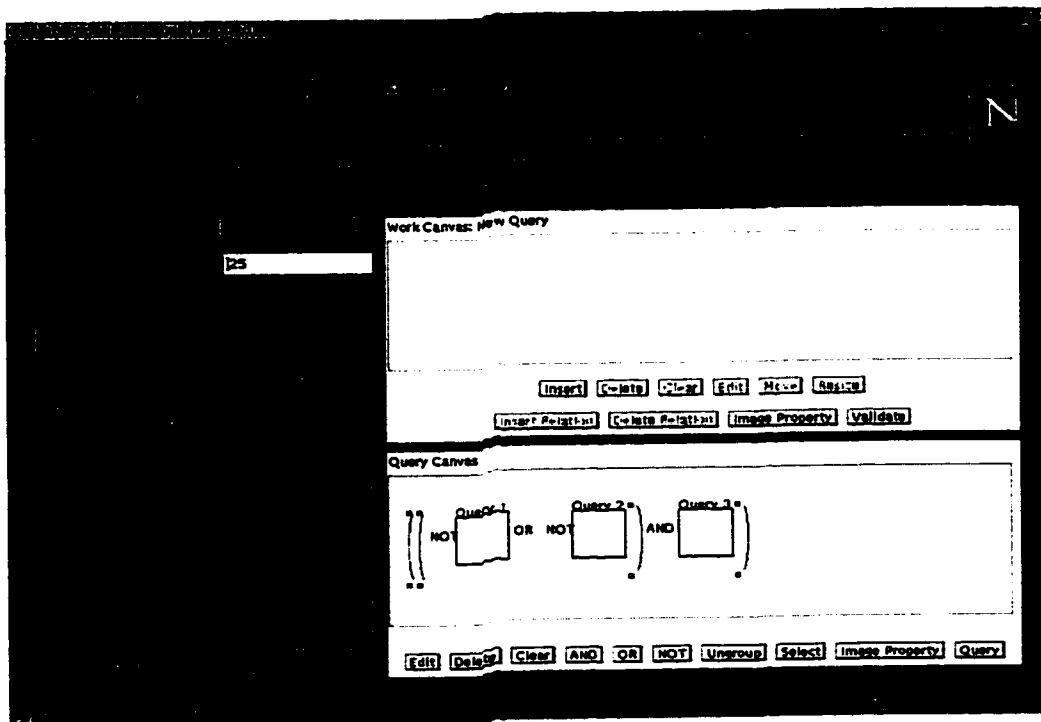


Figure 5.4: The VisualMOQL compound query in Figure 5.3 after being converted to normalized form.

'back' button in the browser after he sees the results, the screen in Figure 5.4 will be shown instead of that in Figure 5.3. However, this may cause some confusion to the user. A better way to do this is to apply the normalization on a clone of the VisualMOQL query object so the query representation in the query canvas will stay the same.

## 5.2 Query Parser and Query Engine

VisualMOQL is responsible for handling all interaction with the user and translating the visual query to MOQL. The DISIMA query facility also includes a MOQL parser and query engine. The MOQL parser, which is the subject of another M.Sc. thesis [22], communicates with the MOQL generator through a common gateway interface (CGI) program. It takes the MOQL string as input and generates an execution plan [22]. At this time, no optimization is performed, so multiple plans are not generated. This plan is executed by the query processor. The whole process is illustrated in Figure 5.5.

The search engine, implemented in C++, reads the query string and other arguments, e.g., image similarity threshold  $\phi_1$  and maximum number of images  $\phi_2$ . Once the MOQL string is parsed, ObjectStore queries are generated and each image in the database is compared with the query image  $M_q$ . The images whose similarity with  $M_q$  is higher than  $\phi_1$  will be filtered out and ranked. Finally, the top ranked  $\phi_2$  images will be returned as the result of the Visual-MOQL query. The search engine is also responsible for displaying the result by generating HTML files with thumbnails of images and links embedded.

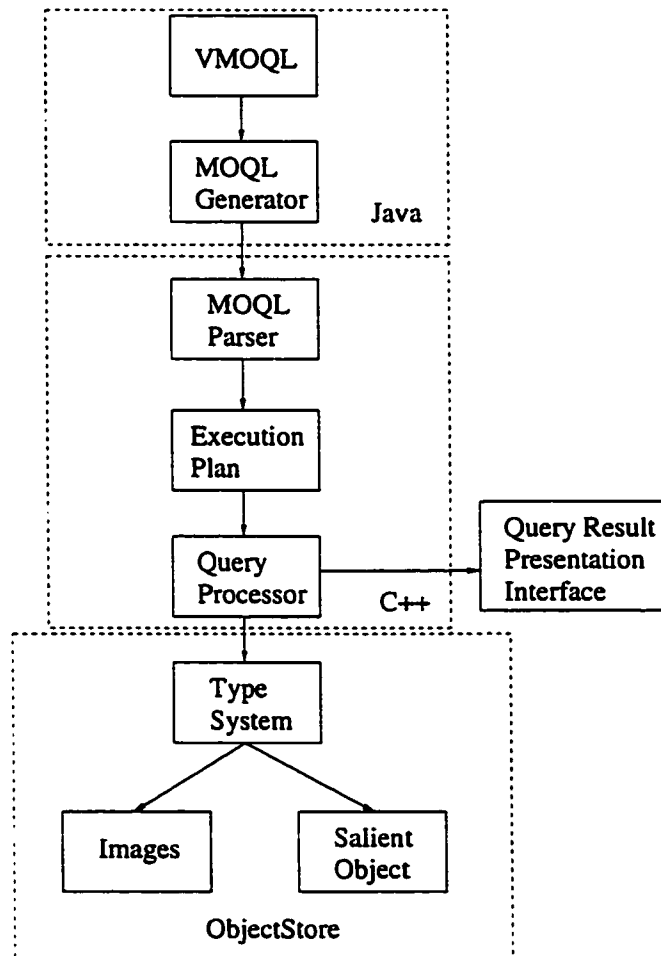


Figure 5.5: *DISIMA* query processing

# Chapter 6

## VisualMOQL Query Examples

This section uses some sample VisualMOQL queries to illustrate the query construction process, the translation of VisualMOQL to MOQL, and the query results of these queries.

**Query 1.** Find all the images that contain a person.

Query 1 is the type of query one can easily express in a textual-based query language using exact matchings on attribute values.

The visual query is shown in Figure 6.1. The equivalent MOQL query produced by the query translator is shown in Figure 6.2 and the query result is shown in Figure 6.3.



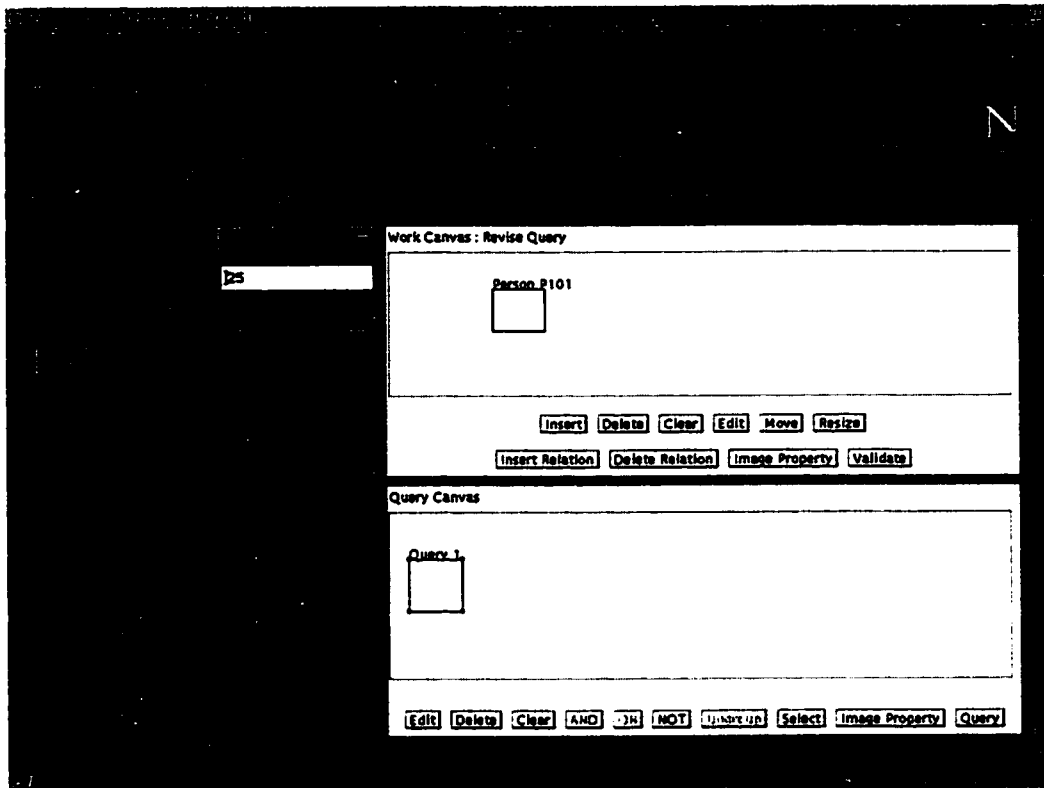


Figure 6.1: *VisualMOQL for query 1*

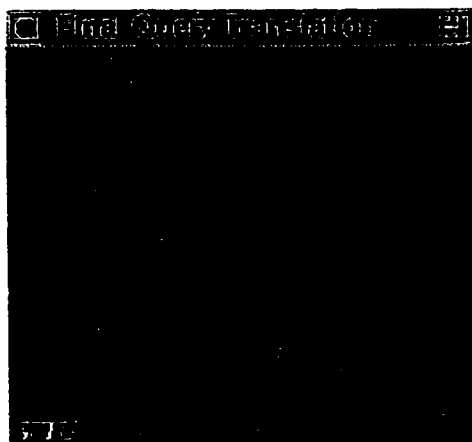


Figure 6.2: *Translated MOQL for query 1*

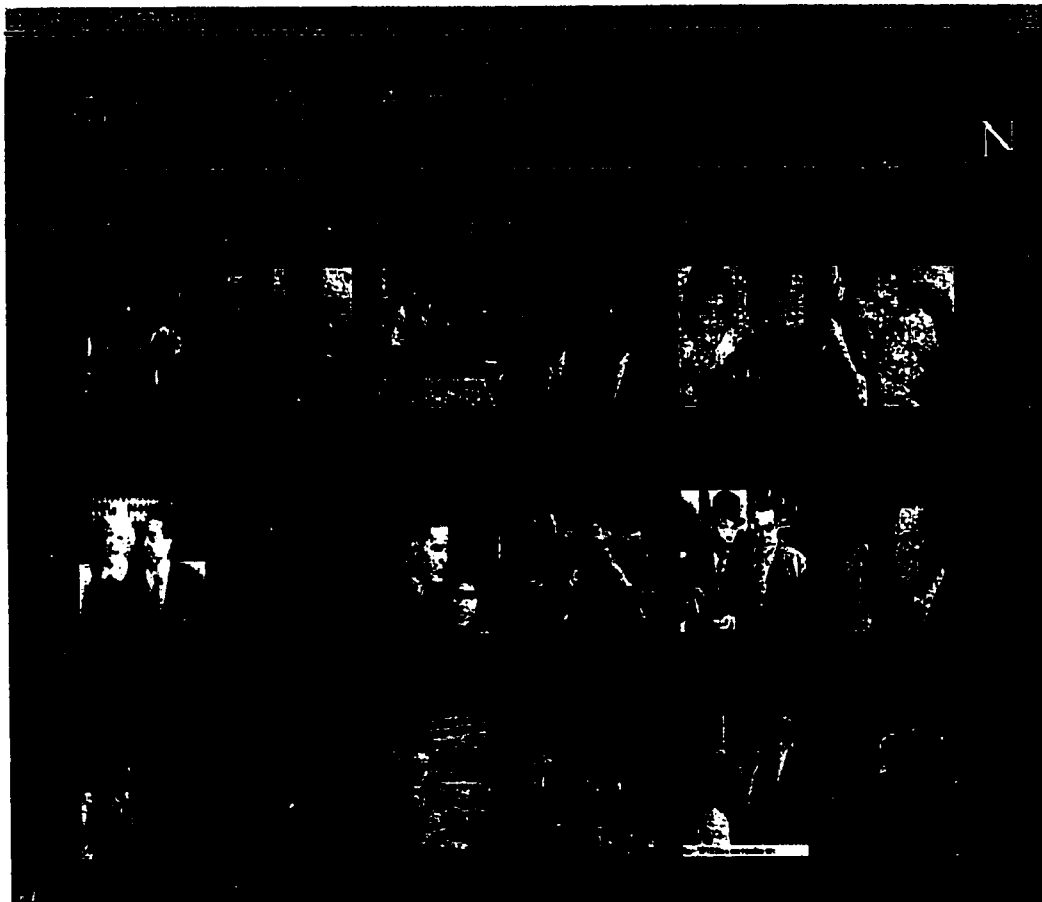


Figure 6.3: *Query result of query 1*

Query 2. Find all images where Bill Clinton appears to be wearing black.

The visual query expression is shown in Figure 6.4. An object property editing dialog box, as shown in Figure 6.5, is used to assign "Clinton" to attribute "lastname". To define the color of the object, the user selects the color he wants and the color is translated into an RGB value. For flexibility, a threshold is associated with the color to provide a search interval in the color histogram.

The equivalent MOQL query produced by the query translator is shown in Figure 6.6 and Figure 6.7 is the query result.

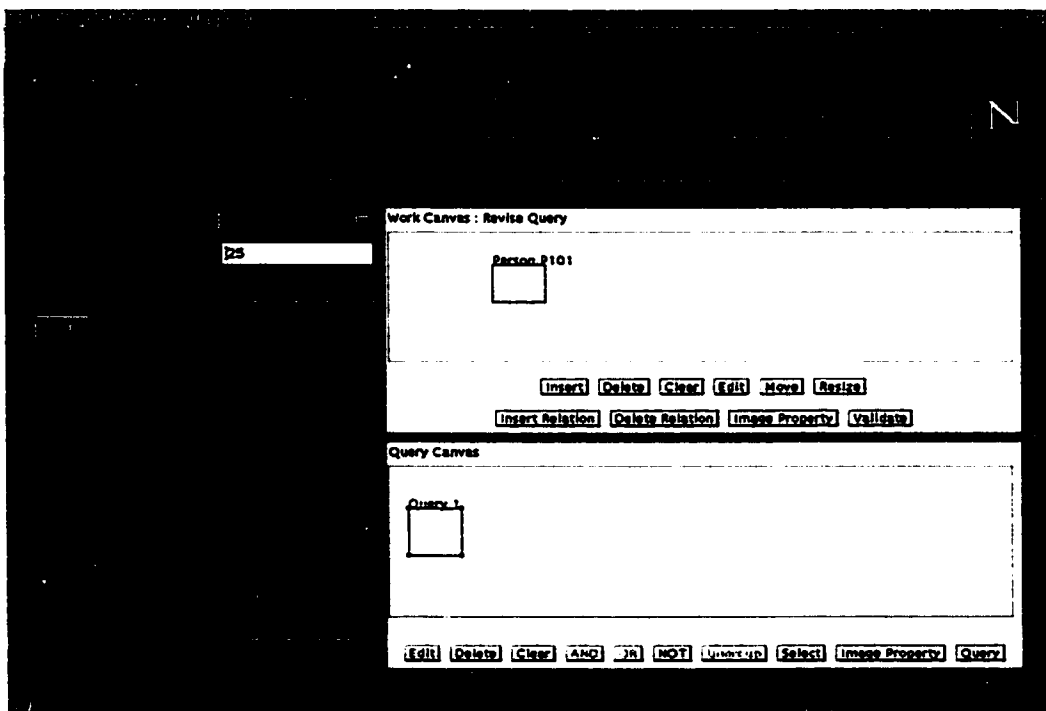


Figure 6.4: *VisualMOQL for query 2*

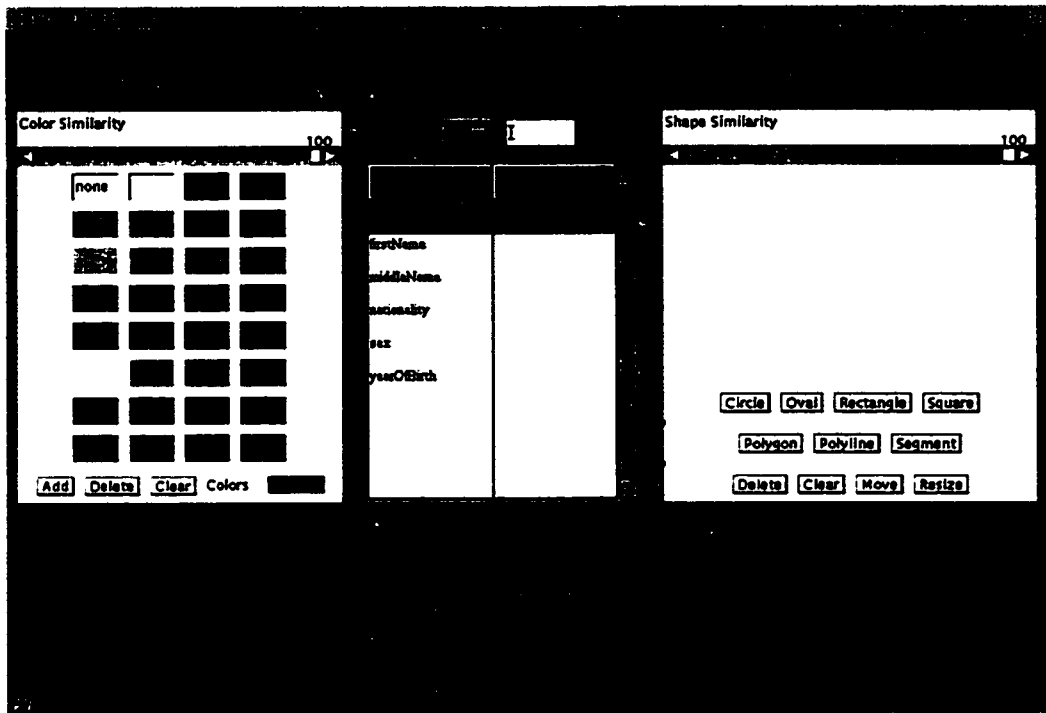


Figure 6.5: Object property window for query 2

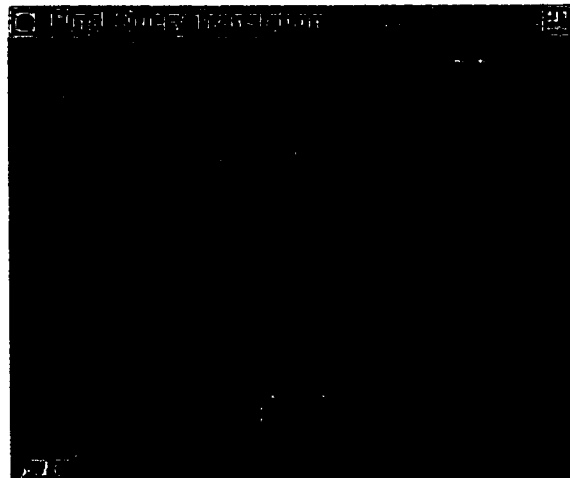


Figure 6.6: Translated MOQL for query 2



Figure 6.7: *Query result of query 2*

Query 3. Find images with Bill Clinton next to Jean Chretien.

The visual query expression is shown in Figure 6.8. To construct this query, the user first introduces two *Person* objects into the working canvas and assigns "Clinton" and "Chretien" as their lastname. The user then must to insert a directional relationship between the two *Person* objects, using the dialog window shown in Figure 6.9.

"Next" is not a simple spatial relationship. The user has to translate it into Bill Clinton at the right of Jean Chretien, or Bill Clinton at the left of Jean Chretien. The user can express this relationship by saying that the x-axis does not matter with the centroids of the two salient objects at the same level on the y-axis.

The equivalent MOQL query produced by the query translator is shown in Figure 6.10, and Figure 6.11 is the query result.

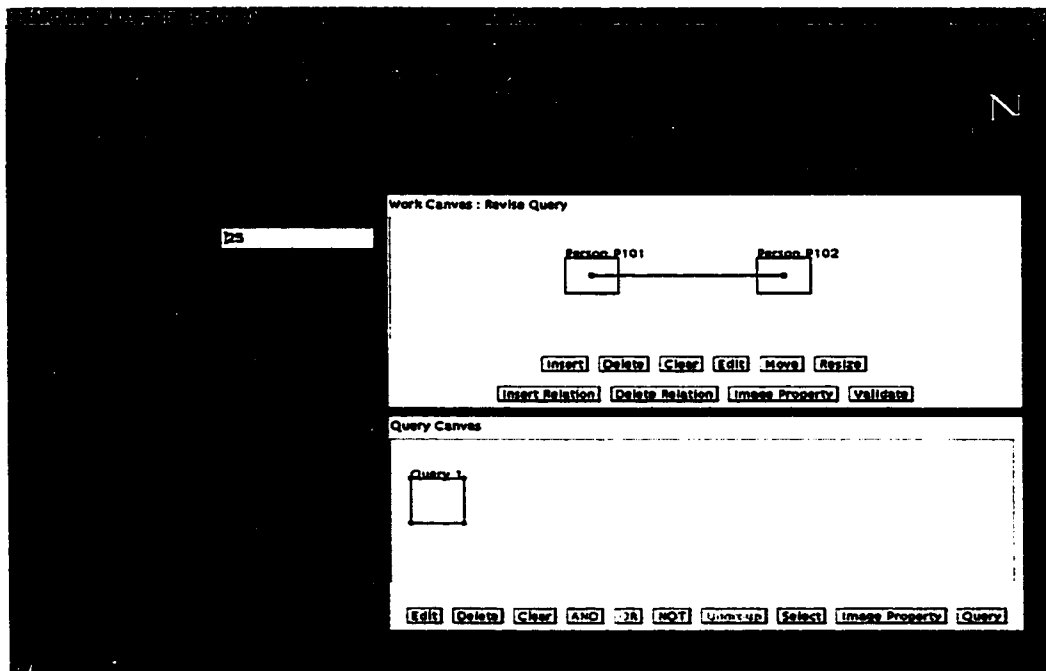


Figure 6.8: *VisualMOQL* for query 3

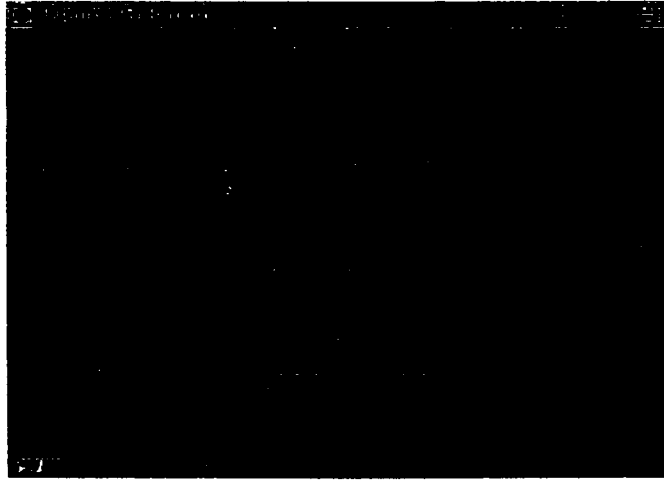


Figure 6.9: *Spatial relationship dialog window for query 3*

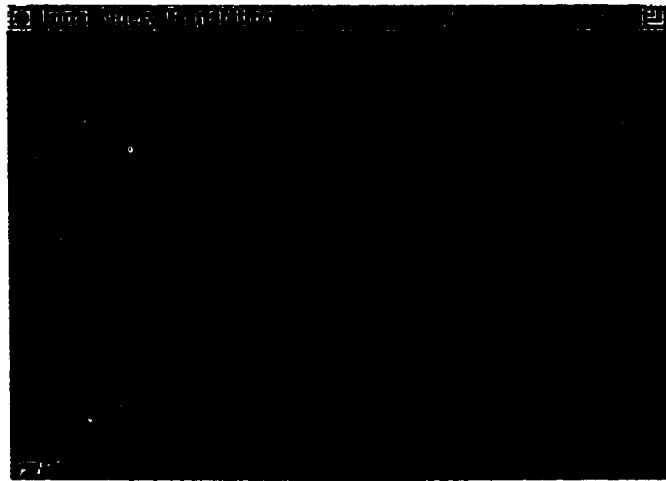


Figure 6.10: *Translated MOQL for query 3*

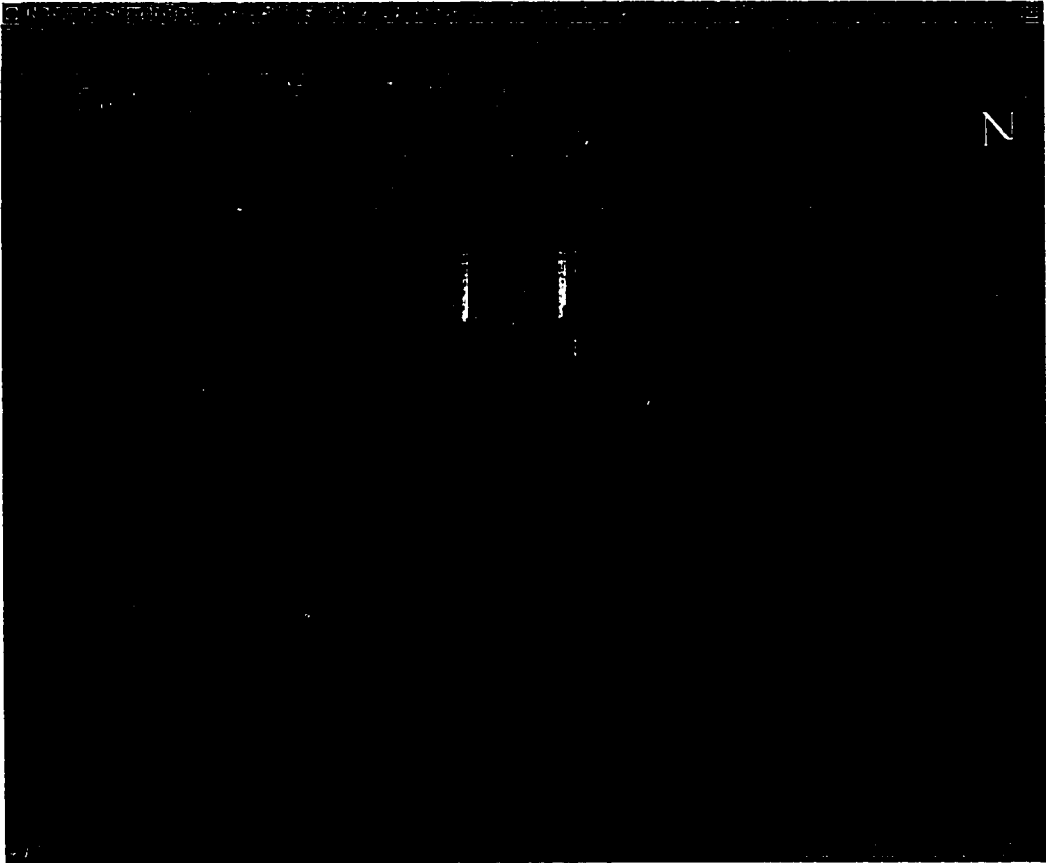


Figure 6.11: *Query result of query 3*



Query 4. Find images with Bill Clinton next to Jean Chretien or images with Hillary Clinton with Bill Clinton.

To construct this query, the user first builds query 3, then introduces a similar simple query specifying query conditions on Bill Clinton and Hillary Clinton. The two simple queries are then connected by the *OR* operator to form the final VisualMOQL query.

The visual query expression is shown in Figure 6.12. The equivalent MOQL query produced by the query translator is shown in Figure 6.13, and Figure 6.14 shows the query result.

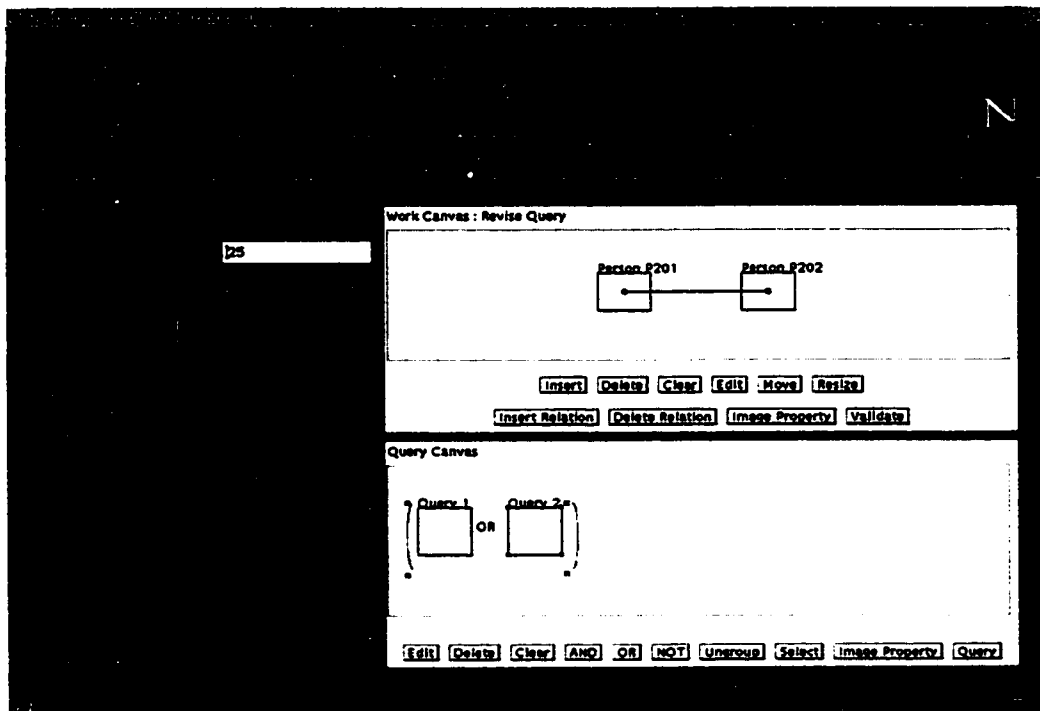


Figure 6.12: *VisualMOQL* for query 4



Figure 6.13: *Translated MOQL for query 4*



Figure 6.14: *Query result of query 4*

Query 5. Find images that look grey--ish or red--ish.

The visual query expression is shown in Figure 6.15. This is also a query that involves two simple queries, each constructed using the image property dialog window shown in Figure 6.16.

The equivalent MOQL query produced by the query translator is shown in Figure 6.17, and Figure 6.18 is the query result.

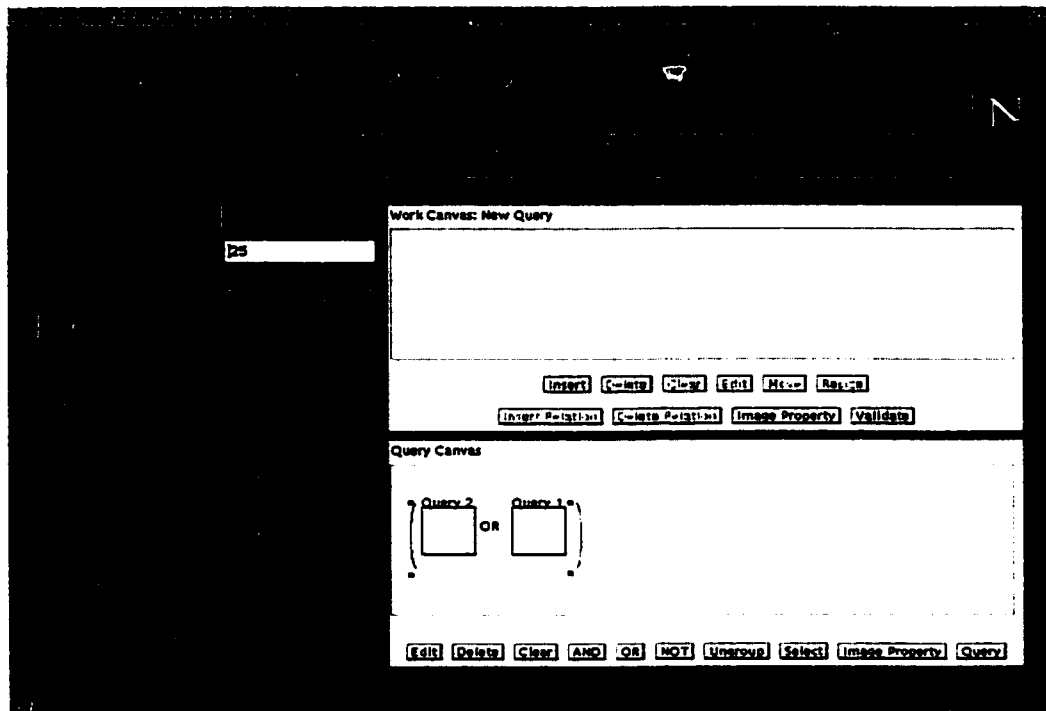


Figure 6.15: *VisualMOQL* for query 5

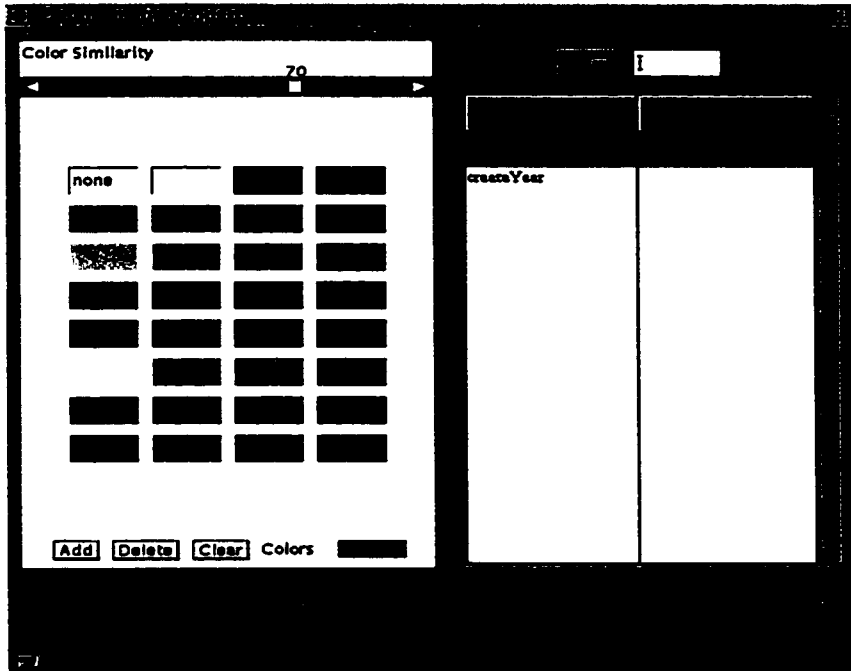


Figure 6.16: *Image property dialog window for query 5*

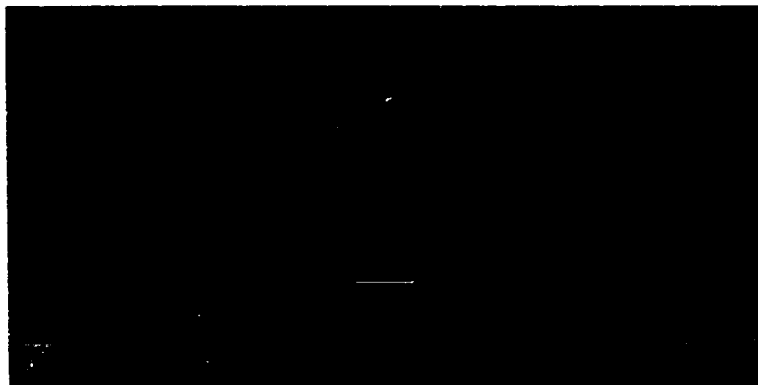


Figure 6.17: *Translated MOQL for query 5*



Figure 6.18: *Query result of query 5*

Query 6. Find images that contain Clinton but do not contain Hillary Clinton.

Two simple queries have to be constructed first for query 6. The first one contains Clinton and the second one contains Hillary, as in query 2. In the query canvas, a negate operator is added to the second query first. Next, the user groups the two simple queries using an AND.

The visual query expression is shown in Figure 6.19. The equivalent MOQL query produced by the query translator is shown in Figure 6.20, and Figure 6.21 is the query result.

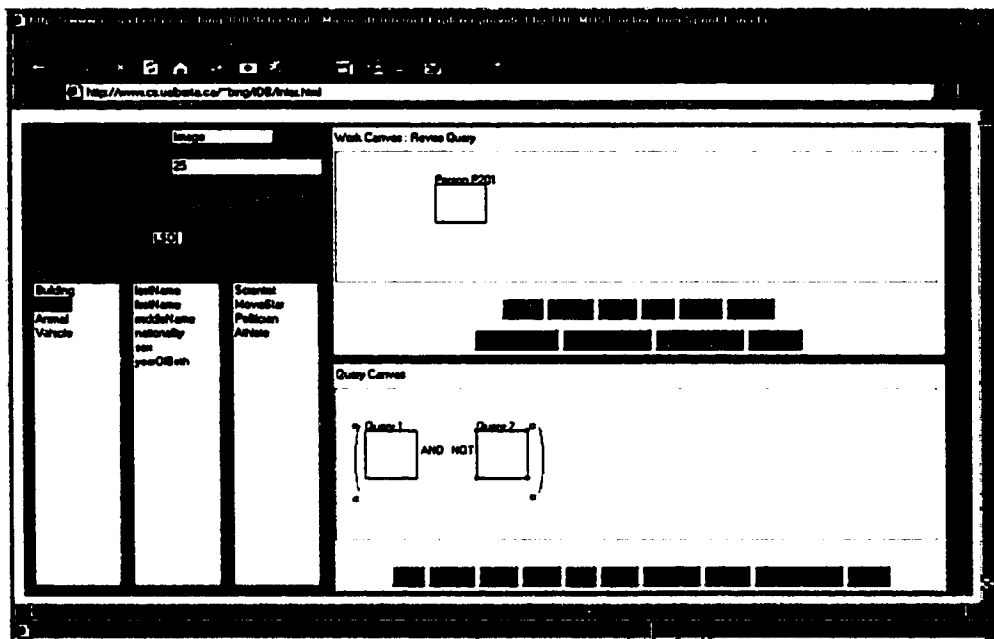


Figure 6.19: VisualMOQL for query 6

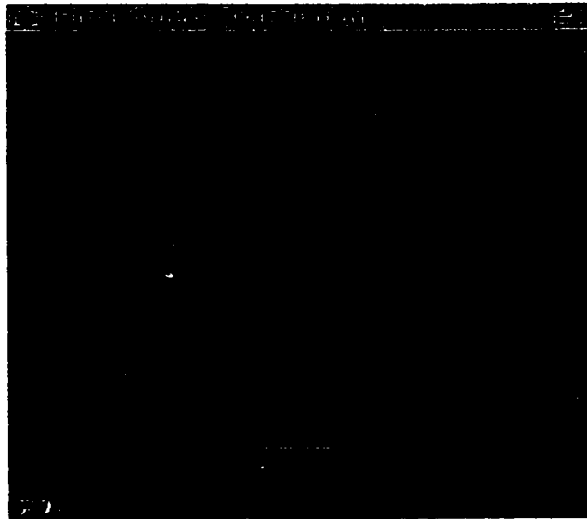


Figure 6.20: *Translated MOQL for query 6*

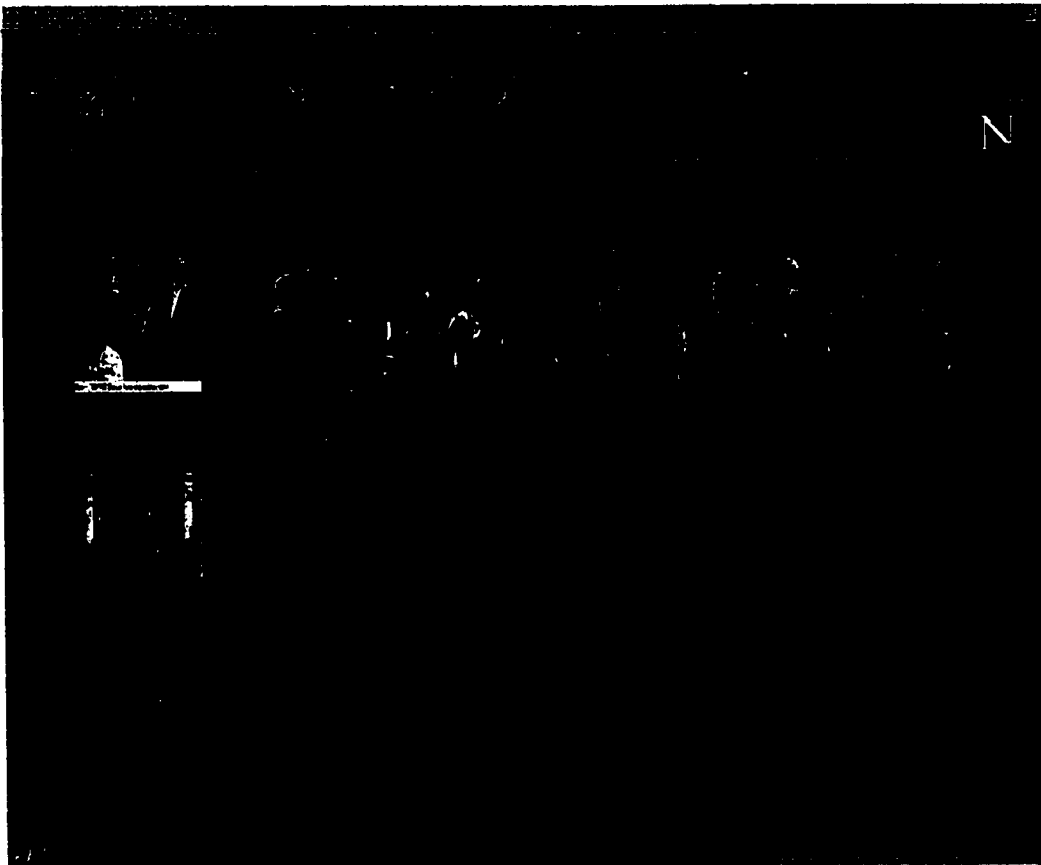


Figure 6.21: *Query result of query 6*

Query 7. Find Animal images that contain birds or Scenic images that contain a house.

This query demonstrates how to retrieve images from different image classes. Animal and Scenic are two image classes defined in the database. To build this VisualMOQL query, the user has to first construct two simple queries by selecting the correct image class from the image classes selector, and inserting the right object. Then s/he connects them using the OR operator.

The visual query expression is shown in Figure 6.22. The equivalent MOQL query produced by the query translator is shown in Figure 6.23, and Figure 6.24 is the query result.

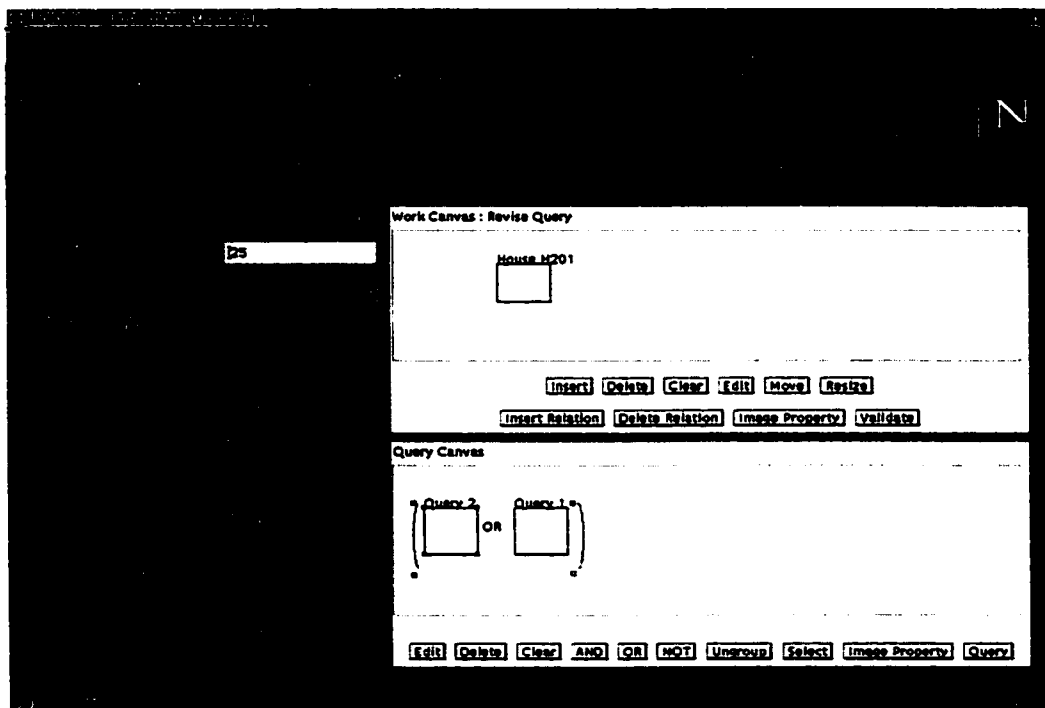


Figure 6.22: VisualMOQL for query 7



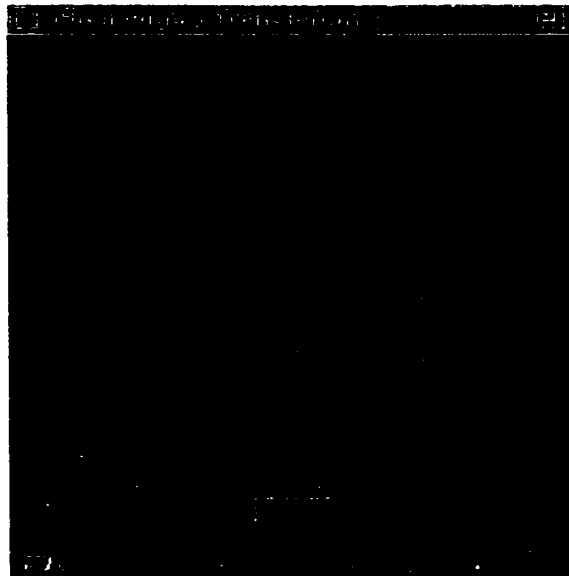


Figure 6.23: *Translated MOQL for query 7*



Figure 6.24: *Query result of query 7*

# Chapter 7

## Conclusions and Future Work

### 7.1 Conclusions

The popularity of visual query languages is gaining momentum; and they are seen as the next step in query language evolution. The aim of visual query languages is to provide users with a friendly interface that easily allows them to pose complex queries. Most of the existing visual query languages have a low expressive power due to the fact that they are often just GUIs designed for naive users. We argue that powerful query languages can provide significant help in simplifying multimedia database access. These languages must provide constructs for querying, based on the structure of multimedia data. For this purpose, we have defined MOQL (Multimedia Object Query Language). MOQL extends OQL by including extensions related to spatial properties, temporal properties, and presentation properties.

VisualMOQL is a visual query language that implements the image features of MOQL. It aims at integrating semantics and media-based search to give users a greater flexibility in posing queries. VisualMOQL combines several approaches, such as semantic-based (query image semantics) and textual-based (specify and compare attribute values). The image matching is based on salient objects, spatial relationships among salient objects, salient object colors, and shape. This gives users a greater flexibility in describing objects with finer granularity than a whole image, and specifying spatial relationships between objects. It bridges the gap between users' perceptive concepts and direct manipulation of query languages.

A query specified using VisualMOQL is translated into MOQL before execution. This distinction between the external query language (VisualMOQL) and the internal query language (MOQL) has several advantages. As a visual language VisualMOQL is easy to use, but provides the same query facilities as MOQL. Thus, users are not required to be aware of schema design and language syntax, which can be very complicated for multimedia data. Since a query specified using VisualMOQL is translated into MOQL before execution, it can make use of the MOQL query processor and optimizer (the optimizer is not yet implemented).

## 7.2 Future Work

Although VisualMOQL shows significant improvement over traditional image database query systems by allowing users to pose queries using combinations of concepts, semantics, visual properties, and spatial relationships, some research issues remain and need more investigation. These include:

- MOQL is a general-purpose multimedia query language, while VisualMOQL implements only the image component. Work is in progress to extend VisualMOQL beyond the image component. To keep the query interface simple, it might be preferable to provide a VisualMOQL interface for each type of application (e.g., video, document, image). These interfaces have to interact with each other to achieve the expressive power of MOQL. For example, a sub-query can be expressed using the VisualMOQL interface dedicated to images, and then combined with another one defined using the video interface. DISIMA, ultimately, will be a distributed and inter-operable image database management system. That means VisualMOQL has to be able to address other sites including the Web.
- In the current implementation of VisualMOQL, users cannot issue queries using sample images. To overcome this limitation, VisualMOQL needs to provide a general browsing tool to allow users to select sample images

from the database — either randomly, or using results from some earlier query. Furthermore, a mechanism must be provided to enable users to customize their definition of image similarities. For example, which color, texture, shape, or spatial relationship matters; and for those that do, what is the weight of each matching criteria? In addition, the data model and search engine must have the corresponding functionalities defined and implemented.

- More query criteria should be incorporated into the interface. For example, VisualMOQL does not support queries that include distance between salient objects. This limits VisualMOQL's usage on some application domains, especially geographical and medical image databases. Other useful features include enabling the user to select texture and more complicated shapes.
- Currently, VisualMOQL is implemented using Java 1.0.2, and runs as an applet inside the web browser on clients' machines. With Java's latest version 1.2, we can simplify the implementation of the interface and make it easier to use and more efficient. For example, with Java Swing, we can implement the salient object class browser using the tree structure. And the new event handling mechanism defined in Java 1.1 should improve the efficiency of the user interface.

The framework presented in this thesis does not fully support all MOQL primitives. However, its purpose is to demonstrate the power of the query language and visual interface, which has been accomplished.

# Bibliography

- [1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison Wesley, 1995.
- [2] R. Ahad and A. Basu. ESQL: A query language for the relational model supporting image domains. In *Proceedings of the 7th International Conference on Data Engineering*, pages 550—559, Kobe, Japan, 1991.
- [3] J. T. Anderson and M. Stonebraker. Sequoia 2000 metadata schema for satellite images. *SIGMOD RECORD*, 23(4):42—48, December 1994.
- [4] M. Angelacio, T. Catarci, and G. Santucci. QBD\*: A graphical query language with recursion. *IEEE Transactions on Software Engineering*, 16(10):1150—1163, 1990.
- [5] H. Arisawa, T. Tomii, and K. Salev. Design of multimedia database and a query language for video image data. In *Proceedings of IEEE International Conference on Multimedia Computing and Systems*, pages 462—467, Hiroshima, Japan, June 1996.
- [6] E. Arkin, L. P. Chew, D. P. Huttenlocher, K. Kedem, and J. Mitchell. An efficiently computable metric for comparing polygonal shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(3):209—215, 1991.
- [7] J. R. Bach, C. Fuller, A. Gupta, A. Hampapur, B. Horowitz, R. Humphrey, R. Jain, and C. Shu. The Virage image search engine: An open framework for image management. In *Proceedings of the SPIE*:

- Storage and Retrieval for Image and Video Databases IV*, pages 76—87, 1996.
- [8] N. H. Balkir, E. Sukan, G. Özsoyoglu, and Z. M. Özsoyoglu. VISUAL: A graphical icon-based query language. In *Proceedings of 12th International Conference on Data Engineering*, pages 524–533, 1996.
- [9] C. Batini, T. Catarci, M. F. Costabile, and S. Levialdi. Visual query systems: A taxonomy. In E. Knuth and L. M. Wegner, editors, *Visual Database System, II*, pages 153—168, The Netherlands, 1992. Elsevier Science Publisher B. V. (North-Holland).
- [10] E. Bertino, F. Rabitti, and S. Gibbs. Query processing in a multimedia document system. *ACM Transactions on Office Information System*, 6(1):1—41, 1991.
- [11] G. Beylkin, R. Coifman, and V. Rokhlin. Fast wavelet transforms and numerical algorithms I. *Communications on Pure and Applied Mathematics*, 44(4):141–183, 1991.
- [12] R. Bretl, D. Maier, A. Otis, and J. Penney. The GemStone data management system. *Object-Oriented Concepts, Databases and Applications*, 1989.
- [13] P. Brodatz. *Textures: a Photographic Album for Artists and Designers*. Dover Publications, New York, 1966.
- [14] M. L. Brodie. On the development of data models. In M. L. Brodie, J. Mylopoulos, and J. W. Schmidt, editors, *On Conceptual Modeling*, pages 19—48, New York, USA, 1984. Springer-Verlag.
- [15] P. Butterworth, A. Otis, and J. Stein. The Gemstone object database management system. *Communications of the ACM*, 34(10):64–77, 1991.
- [16] A. F. Cardenas, I. T. Jeong, R. K. Taria, R. Barker, and C. M. Breat. The knowledge-based object-oriented PICQUERY+ language. *IEEE Transactions on Knowledge and Data Engineering*, 5(4):644—657, 1993.

- [17] R. 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, San Francisco, CA, 1997.
- [18] A. E. Cawkell. Picture-queries and picture databases. *Journal of Information Science*, 19(6):409–413, 1993.
- [19] N. S. Chang and K. S. Fu. Query-By-Pictorial example. *IEEE Transactions on Software Engineering*, SE-6:519–524, August 1980.
- [20] S. Fu Chang, J. R. Smith, and H. Wang. Automatic feature extraction and indexing for content-based visual query. *Technical Report CU/CTR 414-95-20*, Columbia University, 1995.
- [21] S. K. Chang, Q. Shi, and C. Yan. Iconic indexing by 2D-string. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9:413–428, May 1987.
- [22] I. L. Cheng. Image databases: A content-based type system and query by similarity match. Master's thesis, Department of Computing Science, University of Alberta. Available as Technical Report TR-99-03, 1999.
- [23] W. Chu, I. Leong, and R. Taira. A semantic modeling approach for image retrieval by content. *VLDB Journal*, 3(4):445–477, 1994.
- [24] W. W. Chu, I. T. Leong, and R. K. Taira. A semantic modeling approach for image retrieval by content. *VLDB Journal*, 3(4):445–477, October 1994.
- [25] I. F. Cruz and W. T. Lucas. A visual approach to multimedia querying and presentation. In *Proceedings of Fifth ACM International Conference on Multimedia*, pages 109–120, Seattle, WA, November 1997.
- [26] C. J. Date. *A Guide to the SQL Standard*. Addison Wesley, 1987.

- [27] M. Egenhofer and R. Franzosa. Point-set topological spatial relations. *International Journal of Geographical Information Systems*, 5(2):161—174, 1991.
- [28] F. Golshani and N. Dimitrova. Design and specification of Eva: a language for multimedia database systems. In *Proceedings of the International Conference on Database and Expert Systems Applications*, pages 356–362, 1992.
- [29] Y. Gong, C. H. Chuan, and X. Guo. Image indexing and retrieval based on color histograms. *Multimedia Tools and Applications*, 2(2):133–156, 1996.
- [30] R. C. Gonzalez, T. Seppanen, and M. Pietikainen. An experimental comparison of autoregressive fourier-based descriptors in 2d shape classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 201–207, 1995.
- [31] Object Management Group. The common object request broker: Architecture and specification. *OMG Document*, (93.12.43), December 1993.
- [32] V. N. Gudivada and V. V. Raghavan. Content-based image retrieval systems. *Computer*, September:18—22, 1996.
- [33] E. J. Guglielmm and N. C. Rowe. Natural-language retrieval of images based on descriptive captions. *ACM Transactions on Information Systems*, 14(3):237—267, 1996.
- [34] A. Gupta, T. Weymouth, and R. Jain. An extended object oriented data model for large spatial databases. In *Proceedings of the 2nd International Symposium on Design and Implementation of Large Spatial Databases (SSD)*, pages 45–61, Barcelona, Spain, September 1991.
- [35] A. Gupta, T. Weymouth, and R. Jain. Semantic queries with pictures: The VIMSYS model. In *Proceedings of the 17th International Confer-*



ence on Very Large Databases, VLDB, pages 69–79, Barcelona, Spain, September 1991.

- [36] D. Haw, C. Goble, and A. Rector. Guidance: Making it easy for the user to be an expert. In *2nd Int. Workshop on Interfaces to Database Systems*, pages 19–43, 1994.
- [37] N. Hirzalla and A. Karmouch. The role of database systems in the management of multimedia information. In *Proceedings of International Workshop on Multimedia Database Management Systems*, Blue Mountain Lake, New York, August 1995.
- [38] C. Hsu, W. W. Chu, and R. Taira. A knowledge-based approach for retrieving images by content. *IEEE Transactions on Knowledge and Data Engineering*, 8(4):522–532, 1996.
- [39] W. Hsu, T. S. Chua, and H. K. Pung. An integrated color-spatial approach to content-based image retrieval. In *Proceedings of ACM Multimedia*, pages 305–313, 1995.
- [40] M. K. Hu. Visual pattern recognition by moment invariants. *IRE Transactions on Information Theory*, IT-8:179–187, 1962.
- [41] G. Iannizzotto and L. Vita. A new shape distance for content based image retrieval. In *Proceedings of the 22nd VLDB Conference*, pages 371–386, 1996.
- [42] C. E. Jacobs, A. Finkelstein, and D. H. Salesin. Fast multiresolution image querying. In *Proceedings of SIGGRAPH 1995*, pages 277–286, 1995.
- [43] H. V. Jagadish. A retrieval technique for similar shapes. In *ACM SIGMOD Conference*, pages 208–217, 1991.
- [44] H. V. Jagadish, A. O. Mendelzon, and T. Milo. Similarity-based queries. In *Symposium on Principles of Database System*, pages 36–45, 1995.

- [45] P. M. Kelly and T. M. Cannon. Experience with CANDID: Comparison algorithm for navigating digital image databases. In *SPIE Proceedings of the 23rd AIPR Workshop on Image and Information Systems: Applications and Opportunities*, pages 64–75, 1994.
- [46] P. M. Kelly, T. M. Cannon, and D. R. Hush. Query by image example: the CANDID approach. In *SPIE Storage and Retrieval for Image and Video Databases III*, pages 238–248, 1995.
- [47] T. C. T. Kuo and A. L. P. Chen. A content-based query language for video databases. In *Proceedings of IEEE International Conference on Multimedia Computing and Systems*, pages 456—461, Hiroshima, Japan, June 1996.
- [48] C. Lamb, G. Landis, J. Orenstein, and D. Weinreb. The ObjectStore database system. *Communications of ACM*, 34(10):19—20, 1991.
- [49] R. Laurini and D. Thompson. *Fundamentals of Spatial Information Systems*. Academic Press, 1992.
- [50] J. Z. Li and Y. Niu. Query processing and image indexing in multimedia databases. Poster Session, The Annual Conference of Canadian Institute for Telecommunications Research, August 1995.
- [51] J. Z. Li, M. T. Özsu, Duane Szafron, and Vincent Oria. MOQL: A multimedia object query language. In *The Third International Workshop on Multimedia Information System*, pages 19–28, Como, Italy, September 1997.
- [52] W. Li, K. Candan, K. Hirata, and Y. Hara. IFQ: A visual query interface and query generator for object-based media retrieval. In *Proceedings of the International Conference on Multimedia Computing and Systems*, pages 353—361, Ottawa, Canada, 1997.
- [53] W.-S. Li, K. S. Candan, K. Hirata, and Y. Hara. SEMCOG: an object-based image retrieval system and its visual query language. In *Proceed-*

*ings of ACM SIGMOD International Conference on Management of Data*, pages 521—524, Tucson, Arizona, May 1997.

- [54] G. Lu. Image retrieval based on shape. In *Proceedings of the International Conference on Visual Information Systems, Melbourne, Australia*, 1996.
- [55] S. Marcus and V. S. Subrahmanian. Foundations of multimedia database systems. *Journal of ACM*, 43(3):474—523, 1996.
- [56] R. Mehrotra and J. E. Gary. Similar-shape retrieval in shape data management. *IEEE Computer*, 28(9):57–62, 1995.
- [57] T. P. Minka. An image database browser that learns from user interaction. *Technical Report TR 365, MIT*, 1996.
- [58] W. Niblack, R. Barber, and W. Equitz et al. The QBIC project: Querying images by content using color, texture, and shape. In *SPIE Proceedings of Storage and Retrieval for Image and Video Databases*, pages 173–187, 1993.
- [59] Y. Niu, M. T. Özsu, and X. Li. 2-D-S-tree: An index structure for content-based retrieval of images. In *Proceedings of Multimedia Computing and Networking*, pages 110–121, 1999.
- [60] E. Oomoto and K. Tanaka. OVID: Design and implementation of a video-object database system. *IEEE Transactions on Knowledge and Data Engineering*, 5(4):629—643, 1993.
- [61] J. A. Orenstein and F. A. Manola. Probe spatial data modeling and query processing in an image database application. *IEEE Transactions on Software Engineering*, 14(5):611—629, 1988.
- [62] 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*, San Diego, California, December 1997.

- [63] V. Oria, M. T. Özsu, B. Xu, and L. I. Cheng. VisualMOQL: The DISIMA visual query language. In *IEEE International Conference on Multimedia Computing and Systems*, pages 536–542, Florence, Italy, 1999.
- [64] D. Papadias and T. Sellis. A pictorial query-by-example language. *Journal of Visual Languages and Computing (Special Issue on Visual Query Systems)*, pages 53–72, March 1995.
- [65] A. Papantonakis and P. J. H. King. Syntax and semantics of Gql, a graphical query language. *Journal of Visual Languages and Computing*, 6(1):3–25, 1995.
- [66] A. P. Pentland, R. Picard, and S. Sclaroff. Photobook: Tools for content-based manipulation of image databases. In *Storage and Retrieval for Image and Video Databases*, pages 34–47, 1994.
- [67] E. G. Petrakis and C. Faloutsos. Similarity searching in medical image databases. *IEEE Transactions on Knowledge and Data Engineering*, 9(3):435–447, 1997.
- [68] M. E. Ragab, A. M. Darwish, E. M. Abed, and S. I. Shaheen. Face recognition using principal component analysis applied to an Egyptian face database. In *Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, pages 540–549, 1999.
- [69] N. Roussopoulos, C. Faloutsos, and T. Sellis. An efficient pictorial database system for PSQL. *IEEE Transactions on Software Engineering*, 14(5):639–650, 1988.
- [70] B. Scassellati, S. Alexopoulos, and M. D. Flickner. Retrieving images by 2d shape: a comparison of computation methods with human perceptual judgements. In *Proceedings of SPIE Storage and Retrieval for Image and Video Databases*, pages 2–9, 1994.

- [71] S. Sclaroff, L. Taycher, and M. L. Cascia. ImageRover: A content-based image browser for the world wide web. In *IEEE Workshop on Content-based Access of Image and Video Libraries*, 1997.
- [72] J. R. Smith and S. F. Chang. Automated image retrieval using color and texture. *Technical Report CU/CTR 408-95-14, Columbia University*, 1995.
- [73] J. R. Smith and S. F. Chang. Tools and techniques for color image retrieval. In *Proceedings of SPIE Storage and Retrieval for Image and Video Database IV*, 1995.
- [74] J. R. Smith and S. F. Chang. Searching for images and videos on the world-wide web. *Technical Report 459-96-25, Columbia University Center for Telecommunications Research, New York, NY*, 1996.
- [75] J. R. Smith and S. F. Chang. VisualSEEK: a fully automated content-based image query system. *ACM Multimedia*, pages 87–98, November, 1996.
- [76] J.R. Smith and S.F. Chang. SaFe : A general framework for integrated spatial and feature image search. In *IEEE Workshop on Multimedia Signal Processing*, 1997.
- [77] A. Soffer and H. Samet. Pictorial queries by image similarity. In *Proceedings of ICPR*, pages 114–119, 1996.
- [78] M. Stricker and M. Orengo. Similarity of color images. In *Proceedings of SPIE 1995*, pages 381–392, 1995.
- [79] M. Swain and D. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, 1991.
- [80] K. Tsuda, M. Hirakawa, M. Tanaka, and T. Ichikawa. Iconic browser: An iconic retrieval system for object-oriented databases. *Journal of Visual Languages and Computing*, 1:59–76, 1990.

- [81] J. E. Ziegler and K. P. Fahrnich. Direct manipulation. *Handbook of Human-Computer Interaction*, pages 123–133, 1988.
- [82] M. Zloof. Query-by-example: A data base language. *IBM Systems Journal*, 16(4):324–343, 1977.

# Appendix A

## VisualMOQL Translation Algorithm

This appendix contains the pseudo-code of the VisualMOQL translation algorithm.

```
String function TranslateVisualMOQL (Q){

    //Input   Q is a VisualMOQL query.
    //Return  Result is the translated MOQL query of Q.

    if (Q is a simple VisualMOQL query) then
        return SimpleQueryToMOQL(Q);
    else
        return CompoundQueryToMOQL(Q);
}

String function CompoundQueryToMOQL (Q){

    //Input   Q is a compound query.
    //Return  Result is a MOQL query string

    Convert Q to normal form;

    Calculate the image class M that Q is based on;

    Result = "SELECT m FROM " + M + " m " + "WHERE ";

    String whereClause = CompoundQueryGetWhereClause(Q);

    Result += whereClause;

    return Result;
}
```

```

String function CompoundQueryGetWhereClause (Q){

    //Input    Q is a compound query in normal form;
    //Return   Result is the where clause of Q

    String s1, s2, Result;

    if(the left child query of Q is a simple query) then
        s1 = its where clause by calling SimpleQueryGetWhereClause();
    else
        s1 = its where clause by calling CompoundQueryGetWhereClause();

    if(the right child query of Q is a simple query) then
        s2 = its where clause by calling SimpleQueryGetWhereClause();
    else
        s2 = its where clause by calling CompoundQueryGetWhereClause();

    if(Q is an AND query) then
        String operator = " AND ";
    else
        String operator = " OR ";

    Result = s1 + operator + s2;

    return Result;

}

String function SimpleQueryToMOQL (Q){

    //Input    Q is a simple VisualMOQL query.
    //Return   Result is a MOQL query string

    calculate the image class M that Q is based on;

    Result = "SELECT m FROM " + M + " m " + "WHERE ";

    String whereClause = SimpleQueryGetWhereClause(Q);

    Result += whereClause;

    return Result;

}

```



```

String function SimpleQueryGetWhereClause (Q){

    //Input   Q is a simple VisualMOQL query.
    //Return  Result is the where clause of Q;

    String Result = "";

    if (Q is a negation) then{
        calculate the image class M that Q is based on;
        Result = "m not in (SELECT m FROM " + M + " m " + "WHERE ";
    }

    for (each salient object Obj in Q) {
        add "M contains Obj" predicate to Result;
        add color, shape, and property predicators to Result;
    }

    for (each spatial relationship Relation in Q) {
        add Relation to Result;
    }

    if (any image property is define){
        add image properties predicators to Result;
    }

    if (Q is a negation) then{
        Result += ")";
    }
    return Result;
}

```