Hierarchical image modeling for object-based media retrieval

Wen-Syan Li*, K. Selçuk Candan†, Kyoji Hirata, Yoshinori Hara

C&C Research Laboratories, NEC USA Inc., 110 Rio Robles, M/S SJ100, San Jose, CA 95134, USA

Received 19 August 1997; revised 24 October 1997; accepted 2 December 1997

Abstract

Images are more structurally complex than the data stored in the traditional DBMSs. Many systems perform media retrieval based on image feature matching; the fact that images have structures is ignored. On the other hand, we view images as compound objects containing many component objects. Each component object corresponds to a region(s) that is a visually and semantically meaningful entity (e.g. car, man, etc.). As images/objects truly have hierarchical structures and have both visual and semantics properties, we argue that image retrieval using either semantics and image matching alone is insufficient. In this paper, we introduce a hierarchical structure for image modeling to support image retrieval using combinations of semantic expressions and visual examples at both the whole image and object levels. We give formal definitions of a multimedia query language and a system implementation based on this image modeling. We also address associated query processing issues and discuss the portability and extendibility of our approach.

Keywords: Object-based image retrieval; image modeling; hierarchical structure; image databases; multimedia query language; SQL3

1. Introduction

Image retrieval is a key issue in many image database applications. We categorize the existing approaches to content-based image retrieval as follows:

- Browsing and navigational approach: Users are presented with directories where each directory contains a single category of images (e.g. business, people and sports). The users navigate through directories and then select a particular directory to browse through images of that category. An example of this approach can be typical clip art libraries on CDs for users to use in word processors or presentation tools.
- Syntactic keyword-based approach: Users input keywords and are presented with the images whose captions or keyword lists contain specified keywords. The tasks performed for this approach are the same as the tasks of query processing on traditional DBMSs since no image matching is performed.

* Corresponding author. E-mail: wen@ccrl.sj.nec.com
† This work was performed when the author visited the NEC Computers & Communications Research Laboratories.
• Descriptive semantics-based approach: Users specify the semantics of images they want to retrieve. This approach requires the semantics of queries and images to be explicit so that query processors can determine the similarity. Many systems are built to allow users to retrieve images based on semantics, such as described in [1]. This approach is adequate at image retrieval based on image semantics. However, it has the weakness of low visual expressive capability. Since image data is primarily visual and it is hard to describe image features, such as textures and shapes, using text alone, it is increasingly important to support image retrieval by visual examples.

• Cognition-based approach: Users may pose queries by providing drawings or image examples. The system retrieves images which are visually similar to the examples provided by the user. Features of images are extracted and are represented as vectors. Image matching are performed through comparing image feature representations. For example, a user may pose a query by drawing a picture in which there is a man next to a building and the system would retrieve images similar to his drawing. This approach has an advantage of using visual examples. However, one weakness of this approach is its lower precision because users’ drawings are usually not precise enough and current image recognition techniques are limited. Another weakness of this approach is that it cannot support queries on generalized concepts. For example, if a user wants to retrieve all images in which there is some kind of transportation, users need to provide drawings for all possible forms of transportation, such as car, truck, bus, boat, etc. This is not practical.

1.1. Problem statement

Image data is structurally more complex than the textual data stored in traditional DBMSs. However, the fact that images have hierarchical structures is usually ignored. Many existing approaches treat an image as a whole entity. Images are represented as vectors of features or lists of keywords for retrieval.

An experimental study conducted by H. Nishiyama et al. [2] points out that end-users have two patterns in their visual memory when they view paintings or images. The first pattern is to roughly view the whole image and focus on its objects which are visually or semantically interesting to the users and their layout. The second pattern is to concentrate on detail attributes of specific objects within the image, such as a man or a desk. These attributes can be visual or semantic.

With these patterns of human’s perception, we argue that both whole image and object-based image matching are essential for image retrieval, while most existing systems only support whole image matching. We also argue that image retrieval using either semantics and image matching alone is not sufficient because images and their component objects have both semantics and visual attributes.

To answer the problem statement addressed above, we introduce an approach to image modeling that captures hierarchical structures of images as well as their semantics and image properties at both the image and object levels. Fig. 1 shows an image Boy_bike.gif containing two objects, boy and bicycle, based on color/shape region division and their spatial relationship. After image processing or user specification, semantics is associated with both image and its two objects. On the right side of Fig. 1, Boy_bike.gif is given a title and two objects are specified with semantics as a boy and a bicycle.

We view an image as a compound object containing multiple component objects. Both the image
and component objects have two properties: *image property* and *semantics property*. Image property is images' visual attributes: how images appear to human's perception. Semantics property, on the other hand, is images' semantics meanings: how human interprets images' visual characteristics. For example, the image property of the image shown in Fig. 1 is *Boy_bike.gif* and its semantics property is ‘boy with a bicycle’. Similarly, the image property of two components are two regions and the semantics property of these two component objects are ‘boy’ and ‘bicycle’, respectively.

Fig. 2 illustrates our approach to image modeling. We model an image (e.g. Boy_Bike.gif) as a compound object with (1) image property, (2) semantics property, (3) component objects associated with contain relationships and (4) spatial relationships among component objects (left-to-right and top-to-bottom). With this image modeling, users can specify queries for image retrieval based on combinations of visual characteristics and semantics meanings at both the image and object levels. In another phrase, a query ‘retrieve images which contain transportation and it looks like this drawing’ can be posed.

In this paper, we give formal definition of the image modeling and its query language. We then present a multimedia database system based on this modeling and query language.
1.2. Paper organization

The remainder of this paper is organized as follows: We first give an overview of related work in this area. In Section 3, we give formal definition of our hierarchical image modeling and its query language. Section 4 presents a hybrid system architecture to implement the hierarchical image modeling and query language. In Section 5, we show how queries are specified using a visual query interface. Section 6 discusses issues associated with query processing. In Section 7 we use several query examples on an image database to confirm the advantage of our methodology. In Section 8, we discuss the portability and extendibility of our approach. Finally, we offer our conclusions.

2. Related work

There has been a great deal of work in image retrieval. In this section, we overview related work and classify it into two categories as follows.

2.1. Image retrieval systems and query/browsing interfaces

In [3], Hirata et al propose a concept of Content-oriented Integration to provide organized media contents and their operations. Content-oriented integration provides an integrated navigation environment that consists of both conceptual-based navigation and media-based navigation. Li et al. [4] introduces a system architecture to support content-oriented Integration through integrating semantics and cognition-based approaches for image retrieval. Later in [5], a visual query interface, called IFQ, is introduced to support complex query specification process.

C. Batini et al. [6] provide a survey of interfaces for general purpose database query systems and discuss the weaknesses of natural language and ‘direct manipulation’ language approaches. They argue that an unfriendly interface could prevent people from using database query systems and that query interfaces should be designed to adapt users rather than vice versa. C. Batini et al. also argue that query interfaces through visual communication are more natural since human perception is based on bi-dimensional visual signs.

H. Nishiyama et al. [2] propose an image-retrieval scheme based on the assumption that end-users have two patterns in their visual memory when they view paintings or images. The first pattern is roughly viewing the whole image, whereas the second pattern is then concentrating on specific objects within the image, such as a man or a desk. Their observations and assumptions are supported by some experimental results presented in their work. Nishiyama et al. further define the features of a picture into three levels: areas, objects and attributes. Users pose queries by visually specifying areas, objects and attributes. Query results are derived through image matching alone.

In [7], Amato et al. suggests an object-based multimedia data model which also addresses the structural properties of the images. They also discuss how the model can capture temporal information. Their image modeling is similar to our approach in this aspect. However, they do not present a language or a user interface to support their data model.

V.N. Gudivada et al. [8] categorizes one type of image retrieval as Retrieval by Spatial Constraints (RSC) that facilitates a class of queries that are based on relative spatial relationships among objects in an image. RSC queries are further categorized into relaxed RSC queries and strict RSC queries. We take a similar approach to compare spatial structures of users’ query specifications with stored images.
Virage [9] is a system for image retrieval based on visual features, such as image primitives, such as color, shape or texture and other domain specific features. Virage's has an SQL-like query language extended by user-defined data types and functions. Virage also provides users a form-based query interface called VIV. Virage supports image matching and keyword-based retrieval functionality on the whole image level.

QBIC [10], developed at IBM, is another system that supports image retrieval using visual examples. The image matching is based on features images, such as colors, textures, shapes, locations and layout of images. Similar to Virage, QBIC supports image matching and keyword-based retrieval functionality on the whole image level and both of them do not provide semantics-based access to objects in images.

Garlic [11] and PESTO [12] are two other projects at IBM related to multimedia, which focus integrating and browsing/querying images in heterogeneous and distributed information sources respectively. PESTO allow users to specify tables and attributes for join/project and other aggregation functionality for multimedia information retrieval.

SCORE [13,14] is a similarity-based image retrieval system developed at University of Illinois at Chicago. This work focuses on the use of a refined ER model to represent the contents of pictures and the calculation of similarity values based between E-R representations of images stored and query specifications. However, SCORE does not support image-matching.

VisualSeek [15] is a content-based image query system developed at Columbia University. VisualSeek uses color distributions to retrieve images. Although VisualSeek is not object-based, it provides region-based image retrieval: users can specify how color regions shall be placed with respect to each other. VisualSeek also provide image comparisons and sketches for image retrieval. However, VisualSeek is designed for image matching, it does not support retrieval based on semantics at either the image level or the object level.

Chabot project [16] at UC Berkeley is initiated to study storage and retrieval of a vast collection of digitized images from the State of California Department of Water Resources. Chabot provides a form based browser where users can either provide metadata, such as authors, keywords, concepts or color-distributions to retrieve images. Chabot also supports concept definition functionality. For example, users can define a range of a color spectrum using a keyword, say, Rose Red and then retrieve images based on these pre-defined keywords.

VISUAL [17] is an object-oriented graphical query language designed for scientific databases where the data has spatial properties and exploratory queries are common. VISUAL has a visual interface to allow users to pose queries based on an object-oriented query specification model. VISUAL's interface is rather a graphical query interface and it is too complicated for non-domain-experts to use.

Similar to VISUAL, ZOO [18] and DeVise [19] are two projects at University of Wisconsin. ZOO is designed to provide a generic environment for desktop experiment management using an OODBMS, while DeVise focuses on data visualization and exploration of large data sets from multiple sources. Although there is some similarity in data modeling and architecture design, the data sets used in ZOO and DeVise are textual data, rather than multimedia data that we focus here.

MARIE [20] is a project at Naval Postgraduate School. It concentrates on facilitating information retrieval from captioned databases. It supports image retrieval based on matching between users' queries in natural language processing and images' captions. It supports natural language processing using a statistical parser for English and it also supports some image processing capability, including shape recognition.
2.2. Extended SQL-like query languages

MQL [21] is a multimedia query language. The syntax of the MQL is \( \text{select } \langle A \rangle \langle V \rangle \text{ from } \langle R \rangle \text{ where } \langle C \rangle \) in which \( \langle A \rangle \) is a list of attributes to be retrieved, \( \langle V \rangle \) is the result of version, \( \langle R \rangle \) is the domain class and \( \langle C \rangle \) is a condition. [21] claims MQL can support complex object queries, version queries and nested queries (e.g. IN). MQL also supports a \text{Contain} predicate through pattern matching on images, voice or text. MQL is very similar to our query language in modeling and query language design. Our system further provides additional predicates, such as ISA and S-LIKE, to allow users to control the relaxation of query processing and this paper additionally addresses many issues associated with implementation.

LORE [22] is an effort to develop a lightweight object repository at Stanford University. The query language for LORE, LOREL, is a SQL-like query language designed for querying semi-structured heterogeneous information, such as WWW. LOREL extends the concept of column names to information/object path descriptions and further provides function calls, such as \text{grep} to support more flexible string matching in information retrieval. DataGuides [23] is a graphical interface to browse information structures. DataGuides and PESTO are similar to IFQ in our system, but neither supports multimedia search functionalities.

WebSQL [24,25] is a project to develop WWW query facilitation language at University of Toronto. It views WWW as a table of documents, in which URL, Title, Type, Last Modification Date are treated as columns. WebSQL is a query language that extends standard SQL by adding information related to WEB documents, such as URL and Title, as column names for query. Some user-defined functions, such as ‘MENTIONS’, are supported for more fuzzy textual string matching. This is similar to the work described here except our modeling is based hierarchical structures and our system supports multimedia search.

3. Hierarchical image modeling and its query language

Many existing approaches treat an image as a whole entity and image matching is based on feature vectors extracted from images. On the contrary, we view an image as a compound object which contains multiple component objects and their spatial relationships. Fig. 3 shows an image \text{Boy\_bike.gif}, whose structure can be viewed as a compound image with two image component objects (i.e. boy and bicycle) based on color/shape region division and their real world semantics.

The relationship between the image and its components is \text{contains}. For the example shown in Fig. 2, the image \text{Boy\_Bike.gif} contains two component objects, namely, \text{Obj1} and \text{Obj2} and has two tags of objects’ left-to-right and top-to-bottom spatial relationships. The spatial relationships described here is similar to the 2D-String representation described in [26]. The image has two properties: \text{image property} and \text{semantics property}. The image property represents the image \text{Boy\_Bike.gif} itself, while the semantics property is the description of the image, ‘Boy with a bicycle’, as shown in Fig. 3.

Similarly, the two component objects, \text{Obj1} and \text{Obj2}, also have both \text{image property} and \text{semantics property}; however, in finer granularity of object level. Note that Fig. 3 is similar to Fig. 2 in modeling the image \text{Boy\_Bike.gif} and \text{Obj2} except of \text{Obj1}. In Fig. 3, we further model \text{Obj1} as a compound object containing two component objects, namely, \text{Obj1-1} and \text{Obj1-2}. Therefore, \text{Obj1} has additional spatial relationships between \text{Obj1-1} and \text{Obj1-2}. 
The level of detail for modeling images depends on users' need. For example, users may specify an object as a house in one application, while further specifying the object as a house and its roof, doors and windows. On the other hand, the level of detail that an image can be modeled depends on the image's resolution.

Traditionally SQL is used to manipulate only textual data stored as tables. SQL3 extends SQL to support user defined data types and user defined functions for manipulating them. Since we model images as compound objects containing multiple component objects and both of them have both image and semantics properties, we implement image modeling as a hierarchically structured user defined data type as shown in Fig. 3.

In order to support manipulation of complex objects with image and semantics properties, the following functionalities are required:

- Traverse image hierarchical structure, from the compound object level to the component object level and vice versa, for comparisons at different levels.
- Perform comparisons based on object semantics properties.
- Perform comparisons based on object image properties.
- Perform comparisons based on spatial relationships.
• Operations allowing users to control the strictness of semantics term comparisons because users’ terminology may not be consistent with the semantics specified to image/objects.

In this section, we introduce the syntax of CSQL (Cognition and Semantics-based Query Language), an instance of the proposed SQL3 standard and its functionalities. We start with giving definitions of terminology used in CSQL.

3.1. Visual property and semantics property

An image or an object can be interpreted based on its semantics and/or image properties. A semantic interpretation of an image is the image’s real world meaning. A visual interpretation of an image is how this image looks based on perception of human or image matching engines. We define the semantic interpretation and visual interpretation of images as semantics property and image property, respectively.

Ideally the semantic interpretation and visual interpretation of an image should be consistent. However they may be inconsistent or of different levels of precision because of the limitations of the image recognition technique. For example, the semantic interpretation (meaning) of an image of a man is specified as ‘man’. However, if the system can only tell, based on its visual interpretation of this image, it is ‘human’. In this example, this image’s image interpretation is ‘human’ while its semantics is specified as ‘man’.

3.2. Entity classification

There are three types of entities used in the query language and query processing: semantic entities, image entities and dual entities and each of these entities can be further classified into compound and atomic. Examples of these entities are as shown in Fig. 4.

• Semantic entities: A semantic entity is an entity with only semantic property. Semantic descriptions of images/objects, or a semantic term are semantic entities. Examples of atomic semantics entities include boy, bicycle and car, which are associated with objects. A compound semantics entity is defined as a semantics entity which contains more than one semantics terms. Examples of compound semantics entities include a boy with a bicycle, and a man next to a car, which are associated with whole images.

• Image entities: An image entity is an entity with only image property. An atomic image entity is an image entity with only an image object, while a compound image entity is an image entity with more than one image object. The images with two balloons or the image with two ships in Fig. 4 are examples of a compound image entity.

Other than the semantic descriptions, users can also provide images or sketches as examples to retrieve visually similar images. These input images and sketches are also image entities. Whether or not they are treated as atomic or compound entities depends on the predicate associated with (i.e. being used for whole image matching or object level image matching). For example, when a user provides a sketch and wants to retrieve all the images which contain this sketch, the sketch is treated as an atomic image entity since object-level image matching is performed. On the other hand, if a user wants to retrieve all the images which are similar to the sketch, the sketch is treated as a compound image entity since whole image matching is performed.
### Dual Entity Classification

<table>
<thead>
<tr>
<th>Dual Entity</th>
<th>Compound</th>
<th>Atomic</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><img src="image1" alt="Image Property" /></td>
<td><img src="image2" alt="Image Property" /></td>
</tr>
<tr>
<td>Semantics Property</td>
<td><img src="boy" alt="boy with a bicycle" /></td>
<td><img src="park" alt="Two boys in a park" /></td>
</tr>
<tr>
<td>Atomic Entity</td>
<td>Compound</td>
<td><img src="boy" alt="Boy" /></td>
</tr>
<tr>
<td>Atomic</td>
<td>Boy</td>
<td>Balloon</td>
</tr>
<tr>
<td>Image Entity</td>
<td>Compound</td>
<td><img src="image" alt="Image" /></td>
</tr>
<tr>
<td>Atomic</td>
<td><img src="image" alt="Image" /></td>
<td></td>
</tr>
</tbody>
</table>

**Fig. 4. Classification of entities and examples.**

- **Dual entities**: Dual entities are polymorphic entities with both image property and semantics property. Dual entities can be classified as two categories: atomic dual entities and compound dual entities. Atomic dual entities are dual entities whose image entities are atomic and have corresponding atomic semantics entities. Compound dual entities are dual entities whose image entities are compound and as a result, their semantics entities are specified as compound accordingly. Examples of dual entities are shown at the top of Fig. 4.

3.3. **Selection criteria**

In this section, we will introduce the additional predicates (i.e. user-defined functions in the SQL3 standard) to extend SQL for object-based image retrieval. We categorize query selection criteria based on the types of entity and operators involved as follows:

- **Semantics-based selection criteria**: Users can specify semantic conditions to retrieve images. Given the semantic terminology is not expected to be standardized, we design CSQL to allow users to specify different levels of strictness for semantic conditions as follows:
  - *is*: The `is` predicate returns true if and only if both of the arguments have identical semantics (man vs. man, car vs. car).
  - *is_a*: The `is_a` predicate returns true if and only if the second argument is a generalization
(defined in the Terminology Database) of the first one. Examples of this relationship are ‘car is_a transportation’ and ‘man is_a human’.

\texttt{s\_like:} The \texttt{s\_like} (i.e. semantically like) predicate returns true if and only if both arguments are semantically similar (man vs. woman, car vs. truck).

Note that the arguments of the above predicates can be semantics, image or dual entities. In all cases, the comparison is based on their semantic property. For example, if the argument is a dual entity, its \textit{semantic property} is used. If the argument is an image entity, its semantics property must be extracted before the predicate can be executed.

- \textbf{Cognition-based selection criteria:} The cognition-based predicates introduced in CSQL is \texttt{i\_like} (i.e. image like). The two arguments to the \texttt{i\_like} predicate are either image entities or dual entities. In either case, comparisons are based on their image property. The input images of the first parameter can be either a whole image or an image object.

- \textbf{Spatial relationship-based selection criteria:} CSQL supports the following spatial predicates: \texttt{above\_of in}, and \texttt{below\_of in}, \texttt{to\_the\_right\_of in}, and \texttt{to\_the\_left\_of in}.

These predicates take two (semantic or dual) and one image entity as input. A predicate of the form \texttt{spatial\_predicate(E1, E2, I)} considers the image \texttt{I} and checks if the two entities \texttt{E1} and \texttt{E2} satisfy the spatial relationship implied by the name of the predicate. In case \texttt{E1} and/or \texttt{E2} is dual entity, there are two cases:

- If the dual entity corresponds to an object in the given image, the system uses both the semantic and image identities (i.e., the corresponding object in the image) to check the spatial matching.
- If the dual entity corresponds to an object in another image, the system uses only its semantic identity for the spatial comparisons.

Note that we focus on the spatial relationships that can be \textit{automatically} identified by most image processing engines. Some spatial relationships, such as \texttt{in\_front\_of} and \texttt{behind}, are difficult to be identified \textit{automatically}. One possible extension is to add user-defined relationships, such as \texttt{holding} and \texttt{approaching}.

The reason why we need \texttt{in} is that users may compare objects in different images and the same object may exist in multiple images. With \texttt{in}, users can specify particular images to check objects’ spatial relationships. \texttt{in} is not used in our image retrieval queries, but it is often required in video retrieval since objects often appear in multiple frames. More detail on usage of \texttt{in} is discussed later in Section 8.2.

We are adding a predicate \texttt{attached} that finds the adjacent objects. Other spatial relationships, such as \texttt{in\_front\_of} and \texttt{behind}, are more difficult to identify automatically. Other user-defined relationships, such as \texttt{holding} and \texttt{approaching}, can be also added to the system so that a query ‘retrieval images contain a man sitting on a rock’ can be posed. This type of object relationship is specified by users manually since automated specification process is an extremely difficult task. In this paper, we do not further address this since this type of relationships is semantics-based and can be processed using traditional DBMSs.

- \textbf{Containment selection criteria:} The first argument of this predicate must be a compound entity. The binary \texttt{contains} predicate returns true if the second argument, is contained in the first argument. The first argument is a compound dual entity while the second argument can be an image entity, or a semantics entity, or a dual entity. The reason why the first argument must be a compound entity is because the granularity of the first argument must be in the higher level than the second argument. Now look at the following three examples:
(1) \textit{select image }P\textit{ where }P\textit{ contains }X\textit{ and }X\textit{ i_like }\langle\text{a sketch}\rangle\textit{)

(2) \textit{select image }P\textit{ where }P\textit{ contains }X\textit{ and }X\textit{ is man}

(3) \textit{select image }P\textit{ where }P\textit{ contains }X\textit{ and }X\textit{ i_like }\langle\text{a sketch}\rangle\textit{ and }X\textit{ is man}

In the above three queries, \(P\) is a dual entity while \(X\) is declared as an image entity, a semantics entity and a dual entity, respectively.

3.4. Expressiveness of CSQL

Based on the predicates described above, CSQL supports the following types of operations and their combinations:

- \textit{Image retrieval based on image matching}: CSQL supports visual similarity-based image retrieval at both the whole image level and the object level. Examples of these two types of operations are as the following queries:

\begin{align*}
\textit{select image }P\textit{ where }P\textit{ i_like }\langle\text{a sketch}\rangle\textit{)} \\
\textit{select image }P\textit{ where }P\textit{ contains }X\textit{ and }X\textit{ i_like }\langle\text{a sketch}\rangle\textit{)} \\
\textit{select P where }P\textit{ i_like }\langle\text{a sketch}\rangle\textit{)}
\end{align*}

‘\textit{select image }P\textit{’ can be viewed as declaration of valuable type for }\(P\textit{ as an image. Since the image hierarchical structure can be recursive and has multiple levels, it is required to have a statement to declare a valuable representing the top level. For example, if a user poses a query as follows:

\begin{align*}
\textit{select P where }P\textit{ i_like }\langle\text{a sketch}\rangle\textit{)}
\end{align*}

the system does not know whether the user wants to match the sketch with the whole image or a particular object in the image. Now look at another example as follows:

\begin{align*}
\textit{select P where }P\textit{ contains }X\textit{ and }X\textit{ i_like }\langle\text{a sketch}\rangle\textit{)}
\end{align*}

In this query, the system knows that \(X\textit{ is a component object of }P\textit{ and }X\textit{ is in one level below }P\textit{. Since the image hierarchical structure is recursive, it is not clear which levels }P\textit{ and }X\textit{ are in. After the user declares }P\textit{ as an image using ‘\textit{select image }P\textit{’}, the system know }P\textit{ is at the top level and }X\textit{ is at the second level.
• **Semantics similarity with different levels of strictness:** Image retrieval by keyword is somewhat similar to the information retrieval problem in the sense that both need to deal with terminology heterogeneity issue. To provide higher flexibility, CSQL supports the $S\_Like$ predicate to resolve wording heterogeneity and the $IS\_A$ predicate to resolve terminology granularity heterogeneity. The $S\_Like$ predicate relaxes the semantical scope horizontally, while the $IS\_A$ predicate relaxes the semantical scope vertically.

• **Containment as a means for traversing image hierarchical structure:** With our image modeling, users can select those images that contain particular entities (a sketch, an image, or a semantics term). An example of this type of queries is (the underlined words corresponds to the terms or images provided by the user):

```sql
select image P
where P contains X
    and X is man
    and X i_like (a sketch)
```

The statement ‘$P$ contains $X$’ actually provides object hierarchy traversal functionality through variable binding. With ‘$P$ contains $X$’, we declare $X$ is an object one level lower than $P$ so that the next operation ‘$X$ is man’ can perform comparisons on the object level rather than the whole image level.

The above query can be rewritten as follows:

```sql
select image P
where PX is man
    and PX i_like (a sketch)
```

This query is similar to the syntax of LOREL [22] to traverse semi-structured data. $PX$ is a more implicit way to specify the structural relation between $P$ and $X$, while $P$ contains $X$ is a more explicit specification.

• **Spatial relationship:** With spatial relationship predicates, users can specify the spatial relationship conditions between objects to describe image’s layout. For instance, a real estate broker may ask a query to retrieve all images of candidate real estate in which there is a house to the right of a tree. A query can be posed in CSQL as follows:

```sql
select image P
where P contains X
    and P contains Y
    and X is house
    and Y is tree
    and X to_the_right_of Y
```

• **Extracting semantics and image properties:** This functionality is similar to the $Project$ operator in the traditional relational data model. CSQL also allows users to extract semantics and image
properties at both the image and object levels by adding semantics and image property attribute names in select statements.

For instance, a user wants to retrieve images containing a transportation and also like to know exactly what type of transportation is and how it looks like. The query can be posed in CSQL as follows:

\[
\text{select image } P, ?1, ?2 \\
\text{where } P \text{ contains } X \\
\quad \text{and } X \text{ is}_a \text{ transportation} \\
\quad \text{and } X \text{ is } ?1 \\
\quad \text{and } X \text{ i}_\text{like } ?2
\]

where ?1 and ?2 are system assigned valuables and ?1 and ?2 a semantics entity and an image entity respectively. The conditions of ‘X is ?1’ and X i_like ?2’ are satisfied by replacing the values with the semantics property and the image property of X. Through replacement-and-satisfaction process, our system supports semantics and image property extraction.

Assume the attribute names of semantics and image property of a dual entity are assigned as semantics and image, respectively, the semantics property and the image property of X can be represented as X.semantics and X.image, respectively. The query above can be viewed as follows:

\[
\text{select image } P, X.\text{semantics}, X.\text{image} \\
\text{where } P \text{ contains } X \\
\quad \text{and } X \text{ is}_a \text{ transportation}
\]

Please note that the statement ‘X is_a transportation’ is the same as the statement ‘X.semantics is_a transportation’. Although X has dual properties, the query processor only compare the semantics property of X with transportation’.

<table>
<thead>
<tr>
<th>Predicate</th>
<th>1st argument</th>
<th>2nd argument</th>
<th>3rd argument</th>
<th>Discussion</th>
</tr>
</thead>
<tbody>
<tr>
<td>is</td>
<td>s/d/l/i</td>
<td>s/d/l/i</td>
<td></td>
<td></td>
</tr>
<tr>
<td>is_a</td>
<td>s/d/l/i</td>
<td>s/d/l/i</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s_like</td>
<td>s/d/l/i</td>
<td>s/d/l/i</td>
<td></td>
<td></td>
</tr>
<tr>
<td>i_like</td>
<td>d/l/i</td>
<td>d/l/i</td>
<td></td>
<td></td>
</tr>
<tr>
<td>contains</td>
<td>d</td>
<td>s/d/l/i</td>
<td></td>
<td></td>
</tr>
<tr>
<td>to_the_right_of</td>
<td>s/d</td>
<td>s/d</td>
<td>i</td>
<td>Semantics are compared</td>
</tr>
<tr>
<td>to_the_left_of</td>
<td>s/d</td>
<td>s/d</td>
<td>i</td>
<td>Semantics are compared</td>
</tr>
<tr>
<td>above_of</td>
<td>s/d</td>
<td>s/d</td>
<td>i</td>
<td>Semantic and/or visual characteristics of a dual entity is searched within image</td>
</tr>
<tr>
<td>below_of</td>
<td>s/d</td>
<td>s/d</td>
<td>i</td>
<td>Positions of entities are compared: if argument is a semantic entity or a dual entity outside of image i semantics of the argument is used, otherwise image identity is used.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 1</th>
<th>Predicates and their usage</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicate</td>
<td>1st argument</td>
<td>2nd argument</td>
</tr>
<tr>
<td>is</td>
<td>s/d/l/i</td>
<td>s/d/l/i</td>
</tr>
<tr>
<td>is_a</td>
<td>s/d/l/i</td>
<td>s/d/l/i</td>
</tr>
<tr>
<td>s_like</td>
<td>s/d/l/i</td>
<td>s/d/l/i</td>
</tr>
<tr>
<td>i_like</td>
<td>d/l/i</td>
<td>d/l/i</td>
</tr>
<tr>
<td>contains</td>
<td>d</td>
<td>s/d/l/i</td>
</tr>
<tr>
<td>to_the_right_of</td>
<td>s/d</td>
<td>s/d</td>
</tr>
<tr>
<td>to_the_left_of</td>
<td>s/d</td>
<td>s/d</td>
</tr>
<tr>
<td>above_of</td>
<td>s/d</td>
<td>s/d</td>
</tr>
<tr>
<td>below_of</td>
<td>s/d</td>
<td>s/d</td>
</tr>
</tbody>
</table>
In this section, we have defined and classified entities and properties used in our multimedia query language. We also defined predicates that extend SQL with operations on the hierarchical image modeling structure including: image matching, semantics-based comparisons, spatial relationship checking and hierarchical structure traversal.

The predicate usage is summarized in Table 1. The columns 1st argument and 2nd argument, we use s, d and i to indicate the types of entities are allowed. The s, d and i stand for semantic entity, dual entity and image entity, respectively. The complete syntax definitions of the CSQIL query language is given in Appendix A.

4. Multimedia database system based on hierarchical image modeling

In this section, we introduce an image database system, SEMCOG (SEMantics and COGnition-based image retrieval), which supports our image modeling described above. SEMCOG is an object-based multimedia system based on the proposed hierarchical image modeling developed at NEC, C&C Research Laboratories.

4.1. Overview

SEMCOG aims at integrating semantics and cognition-based approaches and allows queries based on object-level information. SEMCOG allows users to pose a query ‘Retrieve all images in which there is a person who is to the right of an object that looks like this image’. To support this functionality, the traditional SQL needs to be extended to perform image matching and spatial relationship checking. The above example query can be specified as shown on the left of Fig. 5, in which three types of queries are involved: semantics-based, cognition-based and scene-based. Contains is an operation that provides traversal functionalities in the hierarchical structure, such as traversal from the image level to the object level. The user’s query can be visualized as a graph as shown on the top left-hand side of Fig. 5.

Fig. 5 shows two candidate images. The arrows indicate how the variables X, Y and P are bound with images and component objects in images based on semantics and image property comparisons. (X, Y) is bound based on spatial relationship comparisons. The candidate images are presented to users as a set of images ranked by their relevancy to the user’s query. The relevancy is calculated by matching both semantics meaning and visual characteristics of component objects and their spatial relationships (layout) with users’ query. How to compute it is described in Section 6.2.

Please note that the ISA predicate is a relaxed query condition. The two candidate images contains man and woman, respectively, rather than ‘person’. Therefore, our system features a terminology database to store hierarchical structure of concepts, shown on the right of Fig. 5, for query relaxation.

4.2. System architecture

To support the above functionalities, SEMCOG requires multiple modules to manipulate semantics and image property comparisons and spatial relationship checking. Fig. 6 shows the operation and
Visual Query Specification

Query Language

Select image P where P contains X and P contains Y

(1) X isa person (Semantics-based Query)
(2) Y i_like (Cognition-based Query)
(3) X to_the_right_of Y (Scene-based Query)

Query Processing

Candidate images

Fig. 5. Example query and query processing in SEMCOG.

data flows in SEMCOG. Three layers of functions are provided. The client functions supported by an interface for query specification, result display and query generating. These functions are provided by IFQ (In Frame Query) [5]. With IFQ, queries are posed by specifying image objects and their layouts through a visual query interface using icons and menus in a drag-and-drop fashion rather than using the complicated query language directly. The data storage layer is a database management system that we use to build SEMCOG on top of it and store textual data as well as multimedia data. The server functions are supported by five modules to extend the underlying database system multimedia manipulation and search capability. We next present SEMCOG’s building modules.

4.3. Components of SEMCOG

SEMCOG is based on a hybrid architecture to incorporate multiple modules for various functionalities. The components of SEMCOG, shown in Fig. 6, include Facilitator, Image Data Query Processor, Image Semantics Editor, Terminology Manager and Textual Data Query Processor. Their functionalities are as follows:
4.3.1. Image data query processor

COIR (Content-Oriented Image Retrieval) [3] is the image processing module employed in SEMCOG to perform image matching task. COIR is capable of performing image matching
functionality in the object level granularity by identifying regions defined as homogeneous (in terms of colors) and continuous segments identified in an image. An object may consist of one or more image regions. For example, a man consists of a head and a body, which are identified by COIR as two regions. The main task of COIR is to identify image regions. Note that COIR, alone, cannot identify the objects within an image. COIR needs to consult with Image Component Catalog for matching with existing image samples.

4.3.2. Image semantics editor

Image metadata, such as colors, shapes and semantics, are pre-extracted so that image matching and query processing can compare images’ metadata without accessing images directly. When an image is registered at SEMCOG, some image descriptions, such as the image size, image format, registration date, are automatically extracted, while some metadata, such as semantics of images, cannot be extracted fully automatically. Image Semantics Editor interacts with COIR to specify/identify the semantics of an image.

The interactions between COIR and the editor are shown in Fig. 7. The steps involved in image semantics extraction and editing are as follows: (1) COIR identifies regions in an image. (2) COIR recommends candidate corresponding semantics of identified components by consulting with Image Component Catalog. (3) The user confirms or modifies the results recommended by COIR. (4) The user may select objects to store in Image Component Catalog as ‘representative samples’ for future use. Image Component Catalog is built incrementally in this manner. (5) COIR identifies spatial relationships among the objects based on their weight centers.

For example, in Fig. 7, COIR identifies the semantics of balloons in the first image since the sample image of balloon is similar to those two balloons in the input image (similarity is 0.95 and 0.92, respectively). On the other hand, the system fails to identify the possible semantics of the second
image. In this case, user interaction is needed. After the user specifies the semantics of the second image, airplane, he/she decides to store the image of the airplane into Image Component Catalog for future references.

The image semantics database is conceptually visualized in Fig. 8. The semantics of an image is represented as a linked list of pointers pointing to the image and its components. Each image has a tag describing the whole image, such as ‘boy with a bike’ and ‘balloon festival’. Each object also has tags describing its semantics, such as boy and bicycle.

Note that SEMCOG stores both locations of objects and their relative spatial relationships (horizontal and vertical) as described in 2-D strings [27]. With 2-D strings, we convert relative-spatial-relationship-based queries into text-based queries without computing objects’ spatial relationship during the query processing.

4.3.3. Terminology manager

Terminology Manager maintains a terminology base that is used for term relaxation for queries with is_a and s_like (i.e. semantically similar) predicates. Consulting with Terminology Manager, the query processor can find relaxed terms for s_like and is_a predicates. Terminology Manager returns multiple resulting terms with their ranking. Note that we do not need to build such a terminology database. Currently WordNet [28], a sophisticated lexical reference systems, is employed.

4.3.4. Textual data query processor

Textual Data Query Processor processes queries concerning image or object semantics, which are ‘non-image-matching operations’. All the query statements except those with i_like (image matching predicate) and contains are semantics-based. Textual Data Query Processor is Semantic Query Processor. The query processing performed here is the same as that in traditional relational DBMSs, except the additional task associated with term relaxation predicates, such as is_a, which requires interaction with Terminology Manager.

4.3.5. Facilitator

As SEMCOG is designed as a hybrid architecture of multiple component processing modules, coordination of interactions among these modules. Facilitator coordinates the interactions between components. It performs the following tasks: (1) Query relaxation: It reformulates queries with is_a or s_like predicates by consulting with Terminology Manager, (2) Query forwarding: It forwards
query statements containing *i_like* or *contains* (visually) predicates to *Image Data Query Processor* to perform image matching while forwarding other non-image-matching statements to *Textual Data Query Processor*, and (3) Merging Results: It merges the results of image-matching query statements and non-image-matching operation statements. The tasks include computation of overall similarity values and elimination of images based on specified filtering constraints.

### 4.4. System design consideration

We intend to design SEMCOG as an open hybrid architecture so that many modules can be plugged in as components. In SEMCOG, media-independent data is processed by the *Textual Data Query Processor* while media-dependent data is manipulated through media-dependent modules, such as *Image Data Query Processor*. Other media that can be introduced to our system includes audio and video data. New modules to handle temporal relationships and audio processing are then required.

Like most query optimizers in traditional database systems, *Facilitator* also periodically collect database statistics to improve query processing performance; especially using selectivity for query statement execution sequence arrangement. Note that the selectivity is constructed hierarchically and maintained through view maintenance. For example, the selectivity for *isa(vehicle)* is the summation of *is(truck)*, *is(bus)* and *is(car)* as shown in Fig. 9.

### 5. Query interface, modeling and generating

Much work in multimedia databases has focused on developing more powerful languages or modeling scheme. In contrary, research in query interfaces is not equally promoted. As a result, multimedia database applications cannot fully utilize these advanced languages and modeling scheme since they are too complicated for users to use directly or understand.

*IFQ (In Frame Query)* [5] is a query generator and a visual query interface, rather than a ‘graphical’ query interface used by most existing systems. *IFQ* allows users to pose queries using icons in the drag and drop fashion. *IFQ* also visualizes target images and generates corresponding queries as specification process progresses. In this section, we will show how we design *IFQ* to match with our image modeling and query language syntax.
5.1. Query specifications

The query specification process in IFQ consists of three steps: introducing image objects, describing them and specifying their spatial relationships. In IFQ, objects are represented as bullets and descriptors, small bullets attached to these objects to describe objects’ semantics and/or image property. Fig. 10 shows a query ‘Retrieve all images in which there is a man to the right of a car and the man looks like this image’ posed using IFQ. Please note that the IFQ frame corresponds to the whole image level compound object.

The IFQ query is posed as follows: The user introduces the first object in the image and then further describes the object by attaching ‘i_like {image}’ and ‘is man’ descriptors. After a user specifies a image path or provides a drawing, the interface automatically replaces the descriptor with the thumbnail size image the user specifies. Then, the user introduces another object and describes it using the ‘is car’ descriptor. Finally, the user describes the spatial relationship between these two objects by drawing a line, labeled by to-the-right-of, from the man object to the car object. While the user is
specifying the query using IFQ, the corresponding CSQ query is automatically generated in the CSQ window. Users can pose queries simply by clicking buttons and dragging and dropping icons representing entities and descriptors.

5.2 Query parameters

The first type of interaction between users and IFQ is the modification of query parameters to further specify users’ preference of target images in terms of color versus shape matching, image layout versus individual object matching and other filtering parameters. Fig. 10 shows the query parameters that user can specify in IFQ. When a user presses the button Image Related Parameters, a new window will pop up for him/her to specify meta-data, such as titles, authors, film types. This information is in textual form and to search images based on these parameters is treated as a standard database query. Thus, we do not further discuss it here. Other image related parameters are as follows:

- Ratio of color importance to shape importance that is used for image matching, such as comparisons with \(i\_like\). Please see [3] for details.
- Ratio of object-matching importance to structure-matching importance (used to rank images). Fig. 11 shows examples of candidate images that match user’s query. The leftmost image displays the IFQ specification of the target image. In this example, the specification describes the three objects in the image and their layout in the image space. The numbers next to the objects in other images are the similarity values for the image matching of the corresponding objects. The comparisons on spatial relationships yield only a value of 0 or 1 for ‘no match’ and ‘match’, respectively. The images (a) and (b) are partially matched images. The image (a) satisfies object image matching requirement but its layout does not perfectly match the user’s specification. The image (b) satisfies image layout requirement but one object does not perfectly match the user’s specification. Image (c), on the other hand, has a full structural match but object match. Our system considers all three images as candidates.
- Filtering parameters, such as a minimum degree of relevancy and maximum number of images to retrieve.

5.3 Query modeling and generating

The way that we map CSQ query to visual specifications in IFQ is similar to the ER (Entity-Relational) data model. The entities and relationships in the ER model correspond to the objects and spatial relationships in IFQ. The attributes of objects in our model can be images or semantics.

![Fig. 11. Examples of image matching.](image-url)
The guide lines of modeling CSQL queries to IFQ visual specifications are as follows:

1. 'Select image P where' is a default CSQL statement generated when an IFQ window is initialized.

2. The Containment type of CSQL specifications are modeled by adding objects in the IFQ window. That is, by adding an object in the IFQ window, a CSQL statement 'P contains object_variable' is generated. The corresponding object_variable is assigned by the query generator.

3. Object description type of CSQL specifications are modeled by adding object descriptors and attaching object descriptors to objects. The object descriptors can be 'is man', 'is_a transportation', 'i_like tree.bmp', 's_like man', and so on. When the user attaches an object descriptor, say 'isman', to an object, say X, a CSQL statement 'X is man' is generated.

4. Spatial relationship type of CSQL specifications are modeled by adding lines between two objects and specifying spatial meanings of the lines. When the user draws a line between two objects, say object_variable1 and object_variable2 and specifies the line as to_the_right_of, a CSQL statement object_variable1 to_the_right_of object_variable2 is generated.

On the left side of Fig. 12, we illustrate how each IFQ specification corresponds to the CSQL query statements. The right side of Fig. 12 shows the CSQL query generated for the given IFQ specification. For more detail descriptions of IFQ functionality, please see [5].

6. Issues in query processing

SEMCOG has been implemented on top of a deductive DBMS and a commercial ORDBMS. The issues in query processing in these two implementations are quite different. In this paper, we only address the query processing issues related to implementation using the deductive DBMS.

CSQL can be viewed as a logic-based language consisting of a single rule of the form

$$Q(Y_1,\ldots,Y_n) \leftarrow P_1 \land P_2 \land \cdots P_p,$$

where $Y_1,\ldots,Y_n$ are variables and $P_1,\ldots,P_p$ are the predicates defined in Table 1. For the query
shown in Fig. 12, there are six predicates: two containment predicates, two semantics-based predicates, two image matching predicates and one spatial relationship predicate.

In CSQL, the solution of a query \( Q \) is defined as an ordered set \( S \) of \( n \)-tuples of the form \( (X_1, X_2, \ldots, X_n) \), where (1) \( n \) is the number of variables in query \( Q \), (2) each \( X_i \) corresponds to a variable \( Y_i \) in \( Q \), (3) each \( X_i \) satisfies the binding rules (or more formally the type constraints of the predicates) of the corresponding predicate in \( Q \) and (4) when \( X_i \)'s are simultaneously substituted to the places of \( Y_i \)'s on the right-hand side of the query rule, each and every predicate in \( Q \) are satisfied. The order of the set \( S \) denotes the relevance ranking of the solutions.

Because \( P_1, \ldots, P_p \) may operate on whole images or objects within images, the multiple levels of granularity of the image retrieval and integrating results from various selection criteria on multiple media introduce new challenges to query processing including the following:

- Traditional query execution methods need to be adapted to handle similarity values generated from image matching and semantics-based comparisons. The \( \land \) operator in \( Q(Y_1, \ldots, Y_n) \) has different meaning when the values of \( P_1, \ldots, P_p \) are not either \( true/false \) or \( probability \). Instead they are similarity and relevance values.

- The query processor must rank the relevancy of candidate images to a query based on both object matching and image structure matching above in such a way that users are most satisfied.

- Need to support query relaxations for image matching, semantics comparisons and spatial relationships. Because there may not be a full match, the system needs to consider all possible partial matches. As a result, the query processor cannot eliminate images from being considered.

Methods to handle large intermediate results are required.

In the rest of this section, we discuss possible solutions we have explored for these challenges.

### 6.1. Resolution methods

Although CSQL can be viewed as a logic-based language, it is not truly based on traditional logic, because image matching and semantic relaxation involves similarity values. In standard logic a predicate is analogous to a propositional function and since propositions are either true or false so that predicates have one of the two values: \( true \) or \( false \).

However, in CSQL, predicates do not correspond to propositional functions but to similarity functions which return values between 0.0 and 1.0. Such a deviation in the semantics of the predicate concept causes many challenges. When we substitute the values in an \( n \)-tuple \( t \), of the form \( (X_1, X_2, \ldots, X_n) \), into the corresponding variables in predicates \( P_1, \ldots, P_p \) of \( Q \), these predicates return similarity values \( v_1, \ldots, v_p \) where \( 0.0 \leq v_j \leq 1.0 \). One question is how \( t \) forms a solution to \( Q \).

We see two possible solutions as follows:

1. The first possible solution is transforming similarity functions into propositional functions by choosing a cutoff point, \( r_{true} \), and by mapping all the real numbers in \([0.0, r_{true}]\) to false and the real numbers in \([r_{true}, 1.0]\) to \( true \). Hence, the output of similarity functions can be mapped into either \( true \) or to \( false \). Intuitively, such a cutoff point will correspond to a similarity degree which denotes dissimilarity. An \( n \)-tuple which is simultaneously accepted by all predicates is a solution. In this approach, predicates can refute solutions as soon as they are evaluated so that optimizations can be employed. However, partially matched images, such as Fig. 11(a) and Fig. 11(b) will not be considered.
(2) The second possible solution is to choose to have a more holistic approach by leaving the final decision not to the constituent predicates but to the n-tuple as a whole. We can view this strategy as relaxing conjunction constraints during the resolution process. This approach requires a function, \( \mathcal{R}_\beta \) for evaluating overall similarity of candidate images based on the values of \( P_1, \ldots, P_p \). A solution (image) is accepted if its value of \( \mathcal{R}_\beta \) is greater than a given threshold.

The choice of the solution method is critical. In the first approach, predicates can refute solutions as soon as they are evaluated. This approach is the same as the traditional relational DBMS query processing. If users only consider full matching, this method is preferred because it allows query optimization by early elimination of the search space.

If, however, users are interested in both partial matches and full matches, the second method is more suitable. SEMCOG currently uses the second method since it supports both complete and partial matching by relaxing image matching, semantics comparisons and spatial constraints. We also feel this is a reasonable assumption. Since image retrieval is similar to information retrieval, in real world applications users get partial matches with ranking, rather than full matches.

6.2. Ranking relevance of images

The order of the set \( S \) denotes the order of relevance of the solutions in \( S \) based on their overall similarity. A relevance value of a solution to a user query must be calculated based on both object matching and image structure matching. We argue that the approach of simply taking an average, used by most systems such as [29], is not desirable for relevance value calculations in many cases. One reason is that similarity values are relative, not absolute. The similarity values from image matching, semantics-based comparisons and spatial relationship comparisons are on different scales.

We also explored the methods proposed in [30] as one possible solution for the problem addressed here. Graveno and Garcia-Molina [30] discusses strategies for merging ranks from heterogeneous Internet sources. An assumption condition of their method is that a source must satisfy so that a meta-broker can extract the top objects for a query from the source. However, this assumption condition does not hold true in our query processing since SEMCOG supports partial matching. For example, an image may fail to be a top ranked candidate image in all conditions, but it is a top ranked candidate image based overall similarity calculated. We see two approaches as follows:

6.2.1. User specification approach

The first approach is to allow users to specify how to combine these similarity values—weight of each aspect comparison. In Fig. 10, we allow users to specify ratio of color importance to shape importance and ratio of object-matching importance to structure-matching importance. The ranking function for a result for query \( Q \), \( \mathcal{R}_\beta(\alpha_1, \alpha_2) \), can be specified as:

\[
\alpha_1 \times \text{Object-Matching-Ranking} + \alpha_2 \times \text{Structure-Matching-Ranking}
\]

where \( \alpha_1 \) and \( \alpha_2 \) are between 0.0 and 1.0 and \( \alpha_1 + \alpha_2 = 1.0 \).

The value of the function \( \text{Object-Matching-Ranking} \) is calculated based on object semantics matching, color matching and shape matching. \( \text{Structure-Matching-Ranking} \) is defined as numbers of edge (representing spatial relationships) match between query specification and image layout. How to determine the suitable values for \( \alpha_1 \) and \( \alpha_2 \) and how to derive the right formula for \( \text{Object-Matching-Ranking} \) and \( \text{Structure-Matching-Ranking} \).
Matching_Ranking and Structure_Matching_Ranking truly depend on applications and experiments. For example, for applications in TV stations and newspaper publishers, object matching is more important than layout matching, while for applications in the advertisement industry, layout matching may be more important.

6.2.2. Learning approach

Another approach is to design the system which is capable of learning users’ preference and then adapting the values of $\alpha_1$ and $\alpha_2$, the importance of object matching versus structure matching. The learning is performed through interactions between users and SEMCOG. In Fig. 11, we have three types of partial match. The images that match user query perfectly are categorized as ‘complete match’. When SEMCOG could not find complete match images, SEMCOG presents candidate images instead and ask for user’s response. If the users tend to choose Fig. 11a type of images, the system will give a heavier weight for object match similarity values. On the other hand, if users tend to choose Fig. 11b type of images, the system will give a heavier weight for structure match similarity values.

6.3. Performance issues and considerations

Highlights of our architecture design is its modularity by assigning media-based and semantics-based tasks to different modules, so that different types of tasks are executed by corresponding specialized modules.

Media-based tasks, such as image matching, are usually more expensive and it is harder to estimate their processing costs. On the other hand, semantics-based tasks require mainly traditional database query processing and many existing indexing and query optimization techniques can be applied to gain better performance. We assign these two types of tasks to separate components, the Image Data Query Processor and the Textual Data Query Processor, respectively, and employ a facilitator module to coordinate the interactions between modules.

---

1 Based on our initial study and experiments, the relevance function that we use is:

$$R(t, r_{true}, \alpha_1, \alpha_2, \beta) = \alpha_1 \times \frac{((\Pi_{v_{i,j} > r_{true}} v_{i,j}) \times (\Pi_{v_{i,j} < r_{true}} \beta))^{1/p} - \beta}{1 - \beta} + \alpha_2 \times \frac{\Sigma_{v_{i,j} > r_{true}} 1}{p}$$

where $p$ is the number of predicates in $Q$, $\alpha_1$ and $\alpha_2$ are between 0.0 and 1.0 such that $\alpha_1 + \alpha_2 = 1.0$, $\beta$ is a value greater than 0 and less than $r_{true}$, $r_{true}$ is the similarity cutoff point and $v_{i,j}$ is the similarity value returned by predicate $P_j$ for n-tuple $t$.

The first term corresponds to the calculation of relevancy of object matching based on semantics and visual signatures while the second term corresponds to the relevancy of structural matching. Here, the term $((\Pi_{v_{i,j} > r_{true}} v_{i,j}) \times (\Pi_{v_{i,j} < r_{true}} \beta))^{1/p}$ returns a value between $\beta$ and 1.0 where 1.0 corresponds to the highest relevance and $r_{true}$ corresponds to the minimum predicate relevance. $\beta$ is a offset for preventing the final relevance value to be 0 when one of the predicates has a similarity value of 0 and the use of power of $1/p$ is to prevent the multiplicative term to decrease fast.

In the first term, the subtraction of $\beta$ from $((\Pi_{v_{i,j} > r_{true}} v_{i,j}) \times (\Pi_{v_{i,j} < r_{true}} \beta))^{1/p}$ and the division of the result by $1 - \beta$ is to normalize the result back to between 0.0 and 1.0 from between $\beta$ and 1.0. The second major component of the function, i.e. $(\Sigma_{v_{i,j} > r_{true}} 1)/p$ returns the fraction of the predicates which are matched. We use this term to evaluate results of structural matching.
We assigned the tasks of coordination, query reformulation, forwarding and result integration to the **Facilitator**, because it has a more complete knowledge of the query execution statistics compared to the other individual modules in the architecture, the facilitator can provide a better query execution plan. Another reason for our choice is that the **Facilitator** provides a uniform way of handling all different types of rewrites, such as term relaxations, adjective relaxations and query rewritings (query execution sequence re-arrangement). As a result, the **Facilitator** can generate a more efficient execution plan.

Please note that the query processor has very limited information to estimate the cost of each predicates (user defined functions), which involve image matching and spatial relationship comparisons for the following reasons:

- The selectivity of image matching is hard to estimate and it cannot be recorded for future use since the image matching requires many parameters and these parameters usually vary every time. Furthermore, we cannot build an index on image metadata since it is a set of pixels stored as BLOB.
- The selectivity of spatial relationship comparisons cannot be estimated in advance because it depends on the results of object matching. To illustrate this, we use the query in Fig. 12 as an example. The selectivity of the spatial condition *Y to_the_right_of X* is dependent on the selectivity of *X* and *Y*. The results of *X* are dependent on the selectivity of *is man* and *i_like me.gif*, while we cannot estimate the selectivity for *i_like me.gif*.

As to selectivity statistics for *IS, ISA* and *S_LIKE*, we have developed a method to construct a hierarchical selectivity structure as shown in Fig. 9. The selectivity is updated periodically or updated in real time through view maintenance techniques depending on the frequency of file maintenance.

### 7. Query functionality demonstration

In this section, we use some examples to show the capabilities and to demonstrate the usability of SEMCOG. Fig. 13 illustrates how COIR, the image-processing engine used in SEMCOG, specifies image regions and their semantics. On the left of the figure, we show a set of original images. COIR, first, extracts regions in these images based on color homogeneities and segment continuities. COIR, then, shows the region extraction results to the user for verification. In this example, for the fourth and fifth images, the user overrides the system’s recommendation and specifies proper region segmentations. After confirmation or modification of the segments, the user specifies the semantics of each region. The segmented regions and the corresponding semantics are then stored in the database.

In Fig. 14, we show a query specification and its result. In this example, the user wants to retrieve all images containing some types of transportation and the semantic and visual attributes of the transportation object, which correspond to the columns *image, ?1* and *?2* in the result window. The confidence values are calculated based on the formula presented in Section 6.2. The user then clicks on the *image* column and the *?2* column of the fourth image to view the real images. The user also clicks on the *?2* column of the second image to see the object *boat*.

In Fig. 15, the user tightens the query criteria in the example in Fig. 15 by adding an additional condition: ‘the transportation object must be similar to the new image provided’. As a result, the
Fig. 13. Region specification.
number of results reduces to three. Note that the second and third images in the image column are the same. Since there are two different balloons in this image, two results of different values for the ?2 column appear.

In Fig. 16, we show a query which asks for images containing multiple objects (i.e. transportation and human). Note that there are only three distinct images retrieved out of a total of six results. The reason is because there are multiple semantic meanings assigned to the object and more than one semantic meaning can match human. In this example, boy and girl, woman and girl and woman and girl are assigned to these three distinct images respectively. This is shown in the columns ?4 and ?5 in the result window.

In Fig. 17, the user further refine the query by changing ISA(human) to IS(woman) and adding a spatial relationship constrict: the woman must be on top of the transportation. As a result, only two images are returned.
Fig. 15. Query for retrieving images containing transportations with visual characteristics.
Fig. 16. Query for retrieving images containing multiple objects.
Fig. 17. Query for retrieving images containing multiple objects with spatial relationship.
8. Portability and extendibility

In this section, we discuss the portability of our approach and discuss how to extend the hierarchical image modeling to video modeling and possible implementation.

8.1. Portability

CSQL is designed as a declarative query language. Queries in CSQL can be translated into logic-based queries or SQL3 queries. Now we use examples to illustrate its portability. A query for retrieving images in which there is a man and a car and the man looks like the image ‘/home/monet/images/man.gif’ and is to the right of the car can be posed in CSQL as follows:

```
select image P
where P contains X
  and P contains Y
  and X is man
  and Y is car
  and X i_like '/home/monet/images/man.gif'
  and X to_the_right_of Y
```

This query can be translated into the following query in the first logic form:

```
Image_Retrieval(P, X, Y)←contains(P, X)
  ∧ contains(P, Y)
  ∧ is(X, man)
  ∧ is(Y, car)
  ∧ i_like(X, '/home/monet/images/man.gif')
  ∧ to_the_right_of(X, Y)
```

where \( P, X \) and \( Y \) are variables.

We can also implement the predicates as user-defined functions so that SEMCOG can run on top of any DBMS that supports user-defined functions and user-defined data types (part of the SQL3 standard). For example, schemas of an image database are defined as follows:

```
Image_Object_Table(Image_Name, Image_Path)
Obj_Semantics_Table(Image_Name, Object_Name, Semantics, Object_Image_Path)
Top_Down_Spatial_Relationship_Table(Image_Name, Object1_Name, Orientation, Object2_Name)
Left_Right_Spatial_Relationship_Table(Image_Name, Object1_Name, Orientation, Object2_Name)
```

where Image_Name is the unique identification of images while Object_Name, Object1_Name and
Object2_Name are the unique identification of objects in images; Image_Path and Object_Path are pointers pointing to image and object actual file storage locations; and Orientation can be 'above', 'below', 'to_the_right_of' and 'to_the_left_of'.

Based on these schemas, the example query discussed above can be rewritten in SQL3 as follows:

```sql
select a.Image_Name, Display_Image(a.Image_Path)
from Image_Object_Table a, Obj_Semantics_Table b, Obj_Semantics_Table c,
     Left_Right_Spatial_Relationship_Table d
where a.Image_Name = b.Image_Name
  and a.Image_Name = c.Image_Name
  and a.Image_Name = d.Image_Name
  and b.Semantics = 'car'
  and c.Semantics = 'man'
  and Image_Matching(c.Object_Image_Path, '/home/moet/images/man.gif')
  and b.Object_Name = d.Object1_Name
  and c.Object_Name = d.Object2_Name
  and d.Orientation = 'to_the_right_of'
```

where `Display_Image()` is a function for viewing images given their location paths and `Image_Matching()` is a function which performs image matching on two images.

The above query has no variable and it is a SQL language with user-defined functions, `Display_Image()` and `Image_Matching()`. As a result, each line of the query can be executed independently by the query processor.

There are two types of functions in DBMSs. One is system defined standard SQL functions, such as `.=`. Another type of function is user defined functions, such as `i_like` and `s_like` we introduce in CSQL. After these functions are defined and implemented, the users register these functions to `Function Manager`, who maintains a system table of available functions, their parameters and locations of their object codes. The query interpreter accepts users’ queries and consults `Function Manager` on how to execute user defined functions. Then, the query interpreter dynamically loads the object code of user-defined functions to execute. As a result, SEMCOG can be ported and implemented on any DBMSs that support user-defined types and functions (methods) defined as part of the SQL3 standard. This is shown on the right side of Fig. 18.

With users’ query specifications, our system features multiple query generators; namely, `CSQL Query Generator`, `Logic-based Language Generator` and `SQL3 Query Generator`, can generate various corresponding queries, which is an intermediate language. The `facilitator` receives the CSQL queries and executes them. We can also replace the CSQL query generator with a first logic function language generator so that SEMCOG can be also built on top of mediator-based multimedia databases, such as HERMES [31]. The third option is to build SEMCOG on top of existing DBMSs by implementing the predicates introduced in this paper as user defined functions as we discuss above.

As shown in Fig. 18, our image modeling is language independent and it can be implemented on top of deductive databases, such as HERMES [31] as well as commercial ORDBMSs (Object-Relational Database Management Systems) which support SQL3.
8.2. Extendibility to video modeling and retrieval

Our hierarchical image modeling can be extended to video data. Our goal is to translate the video retrieval process into object-based image retrieval and hence to build a video retrieval system on top of SEMCOG by extending existing CSQL predicates and IFQ.

Fig. 19 shows a three-level model of video representation. At the first level (object level), we model
the video objects. An object consists of two parts: semantics and visual identity. Objects, along with
the corresponding spatial information, form an image. An image with additional information, such as
frame#, appearing_time and caption, forms a video frame. A sequence of video frames along with
additional information, such as Title, Length and temporal information, forms a video clip. Please note
the temporal information available in the video level because the temporal information is derived from
frame# and appearing_time information in the frame level.

For example, a query of the form ‘retrieve video clips in which there are a car and a man and the
car which is moving to the right passes the man’ can be posed using IFQ as shown in Fig. 20. Fig. 20
shows two still images representing this query by two frames in the video separated with at most 6
seconds. In our video modeling, this video retrieval query can be translated into two image retrieval
queries for the two still images with temporal relationship constraint (<6 seconds apart) at the video
level. The IFQ query shown in Fig. 20 can be translated into a Video CSQL (VCSQL) query as
follows:

```sql
select video V
where V contains Frame1
    and V contains Frame2
    and (Frame2.appearing_time - Frame1.appearing_time) < 6
    and (Frame2.appearing_time - Frame1.appearing_time) > 0
    and Frame1 contains X
    and Frame1 contains Y
    and X is car
    and Y is man
    and Y to_the_right_of X in Frame1
    and Frame2 contains X
    and Frame2 contains Y
    and X is car
    and Y is man
    and X to_the_right_of Y in Frame2
```

One way to answer this query is to allocate two frames that match the query specifications and then
check their temporal relationships which can be specified in seconds or numbers of frames. Of course,
we can increase the efficiency of this process by using temporal indices.

Fig. 20. Example IFQ query for video retrieval.
9. Conclusions

In this paper, we introduce an image modeling, in which an image is viewed as a hierarchical structured complex object with both semantics and visual properties at the image level and object level. An image may contain multiple component objects. Each component object also has both its semantics and visual properties. Based on this modeling, we present a novel approach to image retrieval which supports queries using combinations of semantics, visual and spatial conditions at the object level.

In this paper, we give formal definition of the hierarchical image modeling and its query language. We present a multimedia system based on this image modeling. We also address some implementation issues and discuss the portability and extendibility of our approach.

The contributions of our work include

- Hierarchical structure for image modeling based on both semantics and visual property and the object level spatial relationships.
- System architecture and implementation for the proposed image modeling.
- Formal definition and implementation of a multimedia language for incorporating semantics, scene and cognition-base query.
- Matching visual query interface and query generator for image retrieval.

Appendix A. Query language syntax in BNF

query ← select input_description where query_description
query_description ← atomic_query_description|composite_query_description
composite_query_description ← (query_description and query_description)
atomic_query_description ← semantics_based_query_description|scene_based_query_description|
cognition_based_query_description
semantics_based_query_description ← (entity_description is a entity_description)|
entity_description is a entity_description|
entity_description is like entity_description
scene_based_query_description ← (sd_entity_description to the right of sd_entity_description
in image_entity_description)|
sd_entity_description to the left of sd_entity_description
in image_entity_description)|
sd_entity_description above of sd_entity_description
in image_entity_description)|
sd_entity_description below of sd_entity_description
in image_entity_description)
image_based_query_description ← (id_entity_description is like id_entity_description)
containment_based_query_description ← (dual_entity_description contains ids_entity_description)
entity_description ← semantic_entity_description|dual_entity_description|
image_entity_description
sd_entity_description ← semantic_entity_description|dual_entity_description
id_entity_description ← image_entity_description|dual_entity_description
semantic_entity_description ← color_adjectives semantic_constant_description|
color_adjectives semantic_variable_description
dual_entity_description ← color_adjectives image_adjectives dual_variable_description
image_entity_description ← image_adjectives image_constant_description|
image_adjectives image_variable_description
semantic_constant_description ← term|term|...
image_constant_description ← image|image|...|best-of-the query|one of the query
References


---

Wen-Syan Li is currently a research staff member at the Multimedia Software Department of Computers & Communications (C&C) Research Laboratories, NEC USA Inc. He received his Ph.D. in Computer Science from Northwestern University in December 1995. He also holds an MBA degree. His main research interests include multimedia/hypermedia/document databases, heterogeneous databases, object-relational database systems and user interfaces.

Kasim Selçuk Candan is an assistant professor at the Computer Science and Engineering Department at the Arizona State University. He received his B.S. degree in Computer Science from Bilkent University in Turkey in 1993. He received his Ph.D. from the Computer Science Department of the University of Maryland at College Park. His research interests include multimedia document authoring/presentation/retrieval; he has also worked extensively on related topics, such as video indexing, image and video retrieval, video server scheduling and heterogeneous information integration. He has published various articles in respected journals and conferences in related areas.

Kyoji Hirata is currently a research staff member at the Multimedia Software Department of C&C Research Laboratories, NEC USA Inc. He received his B.E. degree in Electric Engineering and his M.E. degree in Electronic Engineering from the University of Tokyo in 1987 and 1989, respectively. He joined NEC Corporation in 1989 and has been engaged in the research and development of multimedia databases. His current research interests include multimedia databases, content-based information retrieval, hypermedia systems and multimedia applications.

Yoshinori Hara received his B.E. and M.E. degrees in the University of Tokyo in 1981 and 1983, respectively. He joined NEC Corporation in 1983. Since then, he has been conducting research on hypermedia/multimedia information systems. He is currently Senior Research Manager, Multimedia Software Department, C&C Research Laboratories, NEC USA Inc. He was visiting at the Computer Science Department, Stanford University, from 1990 to 1991. His present interests include information circulation service on the distributed hypermedia environment.