

Sequential Processing for Content-based Retrieval of Composite Objects*

Chung-Sheng Li, John R. Smith, Lawrence D. Bergman, and Vittorio Castelli
IBM T.J. Watson Research Center, P. O. Box 704, Yorktown Heights, NY 10598
email:{csli,jrsmith,bergman,vittorio}@watson.ibm.com
Telephone: (914) 784-{6661,7320,7946,7665}

ABSTRACT

It is becoming increasingly important for multimedia databases to provide capabilities for content-based retrieval of composite objects. Composite objects consist of several simple objects which have feature, spatial, temporal, semantic attributes, and spatial and temporal relationships between them. A content-based composite object query is satisfied by evaluating a program of content-based rules (i.e., color, texture), spatial and temporal rules (i.e., east, west), fuzzy conjunctions (i.e., appears similar AND is spatially near) and database lookups (i.e., semantics).

We propose a new sequential processing method for efficiently computing content-based queries of composite objects. The proposed method evaluates the composite object queries by (1) defining an efficient ordering of the sub-goals of the query, which involve spatial, temporal, content-based and fuzzy rules, (2) developing a query block management strategy for generating, evaluating, and caching intermediate sub-goal results, and (3) conducting a best-first dynamic programming-based search with intelligent back-tracking. The method is guaranteed to find the optimal answer to the query and reduces the query time by avoiding the exploration of unlikely candidates.

Keywords: Content-based retrieval, image databases, spatial indexing, composite objects, query processing.

1. INTRODUCTION

There is great need to extend the content-based image query paradigm to include content-based retrieval of composite objects. The objective is to provide more sophisticated querying capabilities by integrating spatial and temporal image querying with feature-based querying. Potential applications include:

1. *Environmental epidemiology*: retrieve locations of houses which are vulnerable to epidemic diseases such as Hantavirus and Denge fever based on a combination of environmental factors (e.g. isolated houses that are near bushes or wetlands), and weather patterns (e.g. a wet summer followed by a dry summer).
2. *Precision farming*: retrieve locations of cauliflower crop developments that are exposed to clubroot, which is a soil-borne disease that infects cauliflower crop. Cauliflower and clubroot are recognized spectral signature, and exposure results from their spatial and temporal proximity.
3. *Medical image diagnosis*: retrieve all MRI images of brains that have tumors located within the hypothalamus. The tumors are characterized by shape and texture, and the hypothalamus is characterized by shape and spatial location within the brain.
4. *Real estate marketing*: retrieve all houses that are near a lake (color and texture), have a wooded yard (texture) and are within 100 miles of skiing (mountains are also given by texture).
5. *Interior design*: retrieve all images of patterned carpets which consist of a specific spatial arrangement of color and texture primitives.

*Proc. SPIE/IS&T Symposium on Electronic Imaging: Science and Technology – Storage & Retrieval for Image and Video Databases VI, Jan. '98

Until recently, content-based query and spatial query paradigms have been largely distinct. On one hand, there has been extensive investigation of using logical representations to facilitate efficient processing of spatial and temporal queries of symbolic images,^{1,2} and videos.³ The various logical representations such as 2D-strings,¹ the Θ - \mathcal{R} representation,² and the spatial orientation graph (SOG)² allow indexing and retrieval based upon spatial and temporal relationships.

On the other hand, content-based image retrieval systems, such as Virage,⁴ Photobook⁵ and QBIC,⁶ allow querying based upon image features. Examples of the visual features supported by these systems include color, texture, shape, edges, and so forth.

The integration of content-based and spatial image query methods is only recently being investigated.⁷⁻¹² The SaFe spatial and feature image query engine being developed at Columbia University⁸ computes the spatial and feature queries by decomposing the composite query into parallel, content-based region queries. After the joining the results, the spatial relationships are evaluated only for the surviving images using query-time 2D-string projection and comparison. SaFe is also being extended to spatial and temporal querying of video.¹⁰ In the IBM/NASA satellite image retrieval system, we provide a framework for the querying of earth observation imagery by incorporating semantic and feature-based object descriptions into a rule-based spatial query framework.

The difficulty in content-based querying of composite objects results from the combinatorial explosion in candidate composite objects, which number on the order of $\mathcal{O}(L^K)$ for composites of K simple objects selected from a database of L simple objects. Furthermore, content-based querying requires significant computational processing in retrieving features from the database and computing feature similarities. In order to design a sufficiently scalable and powerful solution, the system needs to efficiently manage the search process in order to answer the queries.

In this paper, we propose SPROC, a method of *Sequential PROC*essing for content-based queries of composite objects. SPROC allows for the composite objects to be retrieved, in general, by spatial, temporal and boolean relationships of simple objects with spatial, temporal, feature and semantics attributes.

The SPROC query method generates an efficient linear ordering of the component descriptions (sub-goals) of the composite object (query-goal) and generates a sequential query processing schedule. Since the content-based components of the query involves the assessment of spatial, temporal and features similarities, the system builds and maintains an overall similarity score between each candidate composite retrieved from the database and the query composite object. The SPROC system selects the candidate composite objects for evaluation in a best-first search. Since the similarity scores are monotonically non-decreasing, we show that by partially back-tracking through only the stages which involve content-based querying, the SPROC method is guaranteed to find the optimal solution in an efficient manner.

The paper is organized as follows: In Section 2, we define the preliminary concepts for representing spatial, temporal, and feature attributes of simple objects. We also describe the construction of composite objects through content-based, spatial, temporal, and fuzzy rules. In Section 3, we discuss the options for pre-materializing composites and indexing simple and composite objects. In Section 4, we describe the proposed sequential processing method for retrieving composite objects (SPROC). Finally, in Section 5, we illustrate SPROC through an example query.

2. PRELIMINARY

In the following discussion, we consider an image database \mathcal{Z} that contains Z images. A typical image contains N simple objects.

2.1. Simple Objects

We define simple objects as spatial regions within the images, or spatial/temporal regions within an image sequences. Simple objects have semantic, feature, spatial and/or temporal attributes. For example, a simple texture object corresponding to a region with homogeneous texture is determined by automatic texture image segmentation.¹³ In this case, we consider that the texture of the i^{th} region in an image is characterized by a texture feature vector \mathbf{v}_i . The content-based similarity between two texture regions \mathbf{v}_i and \mathbf{v}_j is measured by a similarity function $S(\mathbf{v}_i, \mathbf{v}_j)$, defined by:

$$S(\mathbf{v}_i, \mathbf{v}_j) = exp^{-(\mathbf{v}_i - \mathbf{v}_j)^T(\mathbf{v}_i - \mathbf{v}_j)}. \quad (1)$$

Consequently, identical feature vectors have similarity measure of unity, i.e., $S(\mathbf{v}_i, \mathbf{v}_i) = 1$. Feature vectors with infinite Euclidean distance have a similarity measure of zero.

2.2. Extensional Database (EDB)

The spatial, temporal, feature, semantic and relational data for the simple objects is stored in a database. Consider the tabular schema in Table 1 for representing simple objects.

KEY1	KEY2	SEMANTICS	SPATIAL		TEMPORAL		FEATURE			
IMAGE	OBJECT	NAME	LOCATION	SIZE	LOCATION	SIZE	COLOR	TEXTURE	SHAPE	MOTION
IMAGEID	OBID	OBTYPE	X, Y	W, H	TX, TY	TW, TH	\mathbf{v}_c	\mathbf{v}_t	\mathbf{v}_s	\mathbf{v}_m

Table 1. Extensional database (EDB) for simple objects.

This information forms the extensional database (EDB) for the simple objects. In this table, IMAGEID uniquely identifies each image, and OBID uniquely identifies each simple object in each image. Together, IMAGEID and OBID form the primary key for simple objects in the database. The tuple (X,Y) denotes the spatial location of the object centroid. For photographic images, (X,Y) denotes pixel coordinates. For geo-registered images, (X,Y) denotes the latitude and longitude.

The spatial width and height of the minimum bounding rectangle (MBR) of the object is given by (W,H). Similarly, (TX, TY) and (TW, TH) give the temporal location and temporal size, respectively, in terms of temporal bounding rectangles.

Several feature attributes are stored for each object such as color, texture, shape and motion. These features are given by \mathbf{v}_c , \mathbf{v}_t , \mathbf{v}_s , and \mathbf{v}_m , respectively. Semantics information such as a semantic label for each object (such as a “house”) is also stored in the database.

2.3. Composite Objects

Composite objects are defined by the composition of K simple objects. The composition involves spatial, temporal and fuzzy relationships between simple objects. In general, composite objects may be nested within other composite objects. However, in this paper, we consider only the case of content-based querying based upon compositions of simple objects. Fig. 1 illustrates an example composite object consisting of four simple objects: A, B, C, D. The spatial configuration is depicted by the pairwise spatial relationships between the four simple objects.

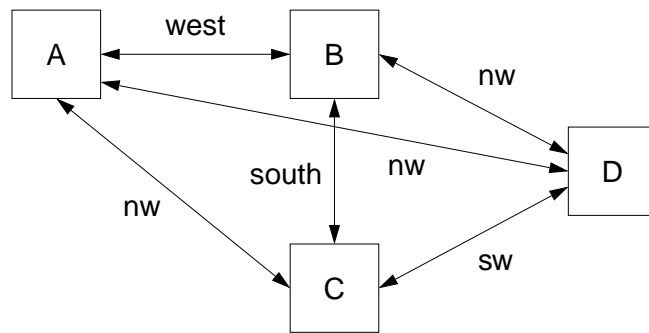


Figure 1. Example composite object consisting of four simple objects with spatial attributes, and spatial relationships between them.

2.4. Composite Object Query

A composite object query is formed by giving a set of simple objects and the attributes and relationships of interest between them. The attributes and relationships are given in the form of sub-goals. Each of the sub-goals is evaluated by the rules of the system.

Fig. 2 illustrates a composite object query which defines a spatial arrangement of the four simple objects: A, B, C, D. We see that the simple objects are given as follows: A in terms of color v_c , B in terms of shape v_s , C in terms of spatial location (x, y) , and D in terms of texture v_t . The spatial composition of A, B, C and D is given by the following spatial relationships: WEST(A, B), SOUTH(C, B), and SW(C, D).

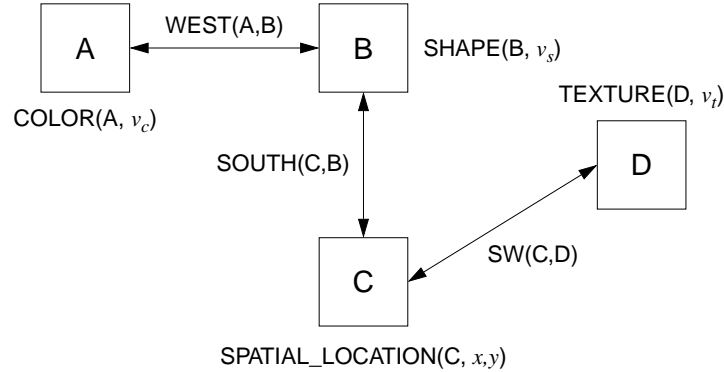


Figure 2. Example composite object query that specifies four simple objects with various feature and spatial attributes, and spatial relationships between them.

As illustrated in Figs. 1 and 2, the composite object query involves only a subset of the attributes and relationships in the candidate composite objects. Each candidate composite object is assessed by measuring its similarity to the query composite object. This overall similarity is determined by the similarities in the attributes and relationships of simple objects between the query and candidate composite objects. These similarities are determined by evaluating sub-goals by spatial and temporal rules, content-based rules, and the fuzzy conjunctions between them.

2.5. Spatial and Temporal Rules

The spatial and temporal relationships of simple objects are determined by a set of rules. For example, consider the following spatial rules:

$$\begin{aligned}
 \text{WEST}(A, B) &\leftarrow A.x < B.x \\
 \text{NORTH}(A, B) &\leftarrow A.y > B.y \\
 \text{NORTHWEST}(A, B) &\leftarrow \text{WEST}(A, B) \wedge \text{NORTH}(A, B).
 \end{aligned}$$

We see that each rule contains a head, i.e., NORTHWEST(A, B), and a body. The body consists of the conjunction of a list sub-goals, i.e., WEST(A, B), and NORTH(A, B). Each of the sub-goals consists of a predicate symbol, i.e., NORTH, and arguments, i.e., A, B. In general, the predicates are either defined by the stored relations (EDB) or by the rules.¹⁴

In order to evaluate the sub-goals, the simple objects A and B are either passed-in explicitly in the arguments of the sub-goals, or are retrieved from the EDB in evaluating the rule. For example, we distinguish between two types of variables which are passed as arguments in the sub-goals: *bounded* and *free*.

2.5.1. Bounded and free variables

In the case of passing bounded variables as arguments, the values are already in hand, and the sub-goal merely acts as a filter in evaluating the function logic. In the case of passing free variables as arguments, objects are first retrieved from the EDB by the system, and then the function logic is evaluated for the retrieved objects. In general, the evaluation of the sub-goals may involve a combination of bounded and free variables.

For example, consider again the sub-goal: NORTHWEST(A, B). We make the following evaluation of the sub-goal with two bounded variables as follows: NORTHWEST(“The White House”, “The Lincoln Memorial”). The arguments of the sub-goal A and B are bounded in the call because the values are given explicitly. Next, consider

the sub-goal: NORTHWEST(“The White House”, y). In this case, only the argument A is bounded, but, B is free and requires an unconstrained lookup in the EDB. Finally, consider the query goal: NORTHWEST(x, y), which asks for all objects which are north-west of another object. Obviously, this requires free lookups of A and B .

There are several techniques for optimizing the evaluation of a set of logical sub-goals involving free and bounded variables (see¹⁴). One technique, called “bound-is-best,” involves the reordering of the sub-goals to give the sub-goals with bounded variables priority over those with free variables. The method includes sideways information passing between sub-goals. Once a free variable is bound, the bindings are retained in subsequent sub-goals.

We use the method of rewriting and ordering the sub-goals in generating the query program, which we discuss in Section 4. We first complicate the matter by including a set of fuzzy spatial and content-based rules, which introduce a new class of fuzzy variables.

2.5.2. Fuzzy Spatial/Temporal Rules

The spatial relationships are implemented in a fuzzy framework as follows: we define the angle between two simple objects A and B by

$$\angle\theta(A, B) = \arctan\left(\frac{A.y - B.y}{A.x - B.x}\right). \quad (2)$$

In the fuzzy framework, the spatial relationships are evaluated by the following fuzzy rules:

$$\begin{aligned} \text{WEST}(A, B) &\leftarrow 0.5(\cos(\angle\theta(A, B) - \pi) + 1) > \text{THRESH} \\ \text{NORTH}(A, B) &\leftarrow 0.5(\cos(\angle\theta(A, B) - \pi/2) + 1) > \text{THRESH} \\ \text{NORTHWEST}(A, B) &\leftarrow 0.5(\cos(\angle\theta(A, B) - 3\pi/4) + 1) > \text{THRESH}. \end{aligned}$$

Alternatively, the fuzzy spatial rules generate K nearest neighbors, as follows:

$$\begin{aligned} \text{WEST}(A, B, K) &\leftarrow 0.5(\cos(\angle\theta(A, B) - \pi) + 1) \in K \text{ smallest} \\ \text{NORTH}(A, B, K) &\leftarrow 0.5(\cos(\angle\theta(A, B) - \pi) + 1) \in K \text{ smallest} \\ \text{NORTHWEST}(A, B, K) &\leftarrow 0.5(\cos(\angle\theta(A, B) - 3\pi/4) + 1) \in K \text{ smallest}. \end{aligned}$$

2.6. Content-Based Rules

The features (i.e., color, texture, shape, motion) of simple objects in the query are given by feature vectors. The content-based rules evaluate the similarity of feature vectors through similarity metrics. For example, consider the following content-based rules:

$$\begin{aligned} \text{COLOR}(A, B) &\leftarrow S(A.v_c, B.v_c) \\ \text{TEXTURE}(A, B) &\leftarrow S(A.v_t, B.v_t). \end{aligned}$$

The content-based rules compute the similarity score S from the distance between feature vectors in a metric space. Unlike the logical rules which return true or false, the similarity score is real-valued between 0 and 1. The content-based rules are defined in terms of similarity thresholds, as follows:

$$\begin{aligned} \text{COLOR}(A, B, \text{THRESH}) &\leftarrow S(A.v_c, B.v_c) > \text{THRESH} \\ \text{TEXTURE}(A, B, \text{THRESH}) &\leftarrow S(A.v_t, B.v_t) > \text{THRESH}. \end{aligned}$$

Alternatively, the content-based constructors generate the K nearest neighbors, as follows:

$$\begin{aligned} \text{COLOR}(A, B, K) &\leftarrow S(A.v_c, B.v_c) \in K \text{ smallest} \\ \text{TEXTURE}(A, B, K) &\leftarrow S(A.v_t, B.v_t) \in K \text{ smallest}. \end{aligned}$$

In both cases, the rules evaluate or generate pairs of simple objects based upon their feature attributes.

2.6.1. Fuzzy Conjunction

We consider the conjunction of sub-goals (corresponding to spatial/temporal and content-based rules, and EDB records), which is performed in the fuzzy framework. For example, consider the query $Q(A, B, a, b)$, which specifies the fuzzy conjunction of three sub-goals as follows:

$$Q(A, B, a, b) \leftarrow \text{TEXTURE}(A,a) \text{ fuzzyAND } \text{COLOR}(B,b) \text{ fuzzyAND } \text{WEST}(A,B).$$

We define the fuzzyAND between sub-goals X_i ($i = 1, \dots, N$) by the weighted sum of the truth-value (or similarity score) of the sub-goals as follows:

$$Q = \sum_{i=1}^N \omega_i S_{X_i}, \quad (3)$$

where S_{X_i} is the fuzzy membership function of X_i , and ω_i is the weight of the i^{th} sub-goals. We also have that ω_i normalized as follows: $\sum_{i=1}^N \omega_i = 1.0$.

3. STORAGE, INDEXING AND QUERY OPTIONS

There exists a wide range of tradeoffs between the storage space and the processing time in order to execute the composite objects queries.

3.1. Tradeoff between Storage Space and Processing Time

We consider that the system, in general, builds various indexes of simple objects, simple object features and composite objects, which support the evaluation of the content-based composite object queries.

We discuss three examples, which demonstrate the extremes in storage and processing requirements: the indexing of simple objects, the indexing of all pairwise combinations of simple objects, and the indexing of all combinations of K simple objects.

3.1.1. Simple Object Indexing

Consider first that the system indexes only simple objects. For example, the simple objects are indexed by absolute spatial location and features using a multi-dimensional index. In this case, the evaluation of relationships between simple objects needs to be performed entirely at query time.

For example, consider a database with Z images; each image contains, on average, N simple objects. The total space required for storing the simple objects is $\mathcal{O}(ZN)$. The exhaustive evaluation of all pairs of simple objects requires the retrieval of $\mathcal{O}(ZN^2)$ composite pairs.

We see that, in general, the complexity of the search process grows combinatorially with the number of simple objects in each composite. Although the storage requirements are low, at query time, the system must generate and evaluate $\mathcal{O}(ZN^K)$ combinations of K simple objects for the database of Z images in order to answer queries involving compositions of K objects.

3.1.2. Pairwise Relationship Indexing

The amount of processing is reduced by pre-materializing and indexing all pairs of simple objects. For example, consider that each image has N simple objects, then there are N^2 pairs. We represent the spatial relationship between object pair (i, j) by $(R_{i,j}, \theta_{i,j})$, where $R_{i,j}$ is the distance between the centroids of the objects and $\theta_{i,j}$ is the angle between them.

Altogether, the system stores $\mathcal{O}(ZN^2)$ records defining pairwise relationships between simple objects, which is an increase by a power of two over the case of indexing the simple objects. However, the query processing is reduced since the system does not need to generate candidate composite pairs at query time.

3.1.3. Composite Object Indexing

The final alternative is to pre-extract and index all combinations of K simple objects directly. In this case, each composition of K simple objects is specified by a record of the following form:

$$(R_{1,2}, \theta_{1,2}, \dots, R_{i,j}, \theta_{i,j}, \dots, R_{K-1,K}, \theta_{K-1,K}),$$

which is formed by concatenating the records of all the combinations of K objects.

The amount of storage required increases significantly, to $\mathcal{O}(ZN^K)$ composite records. However, querying may be performed directly on the composite objects, which reduces processing at query time.

We see that none of these indexing and query solutions alone is acceptable since they do not scale in terms of either storage or processing with respect to the number of simple objects in the database or number of simple objects within each composite object.

3.1.4. Incomplete Index Support

In a typical database environment, there is incomplete index support in that only a subset of attributes and relationships are indexed. For example, features such as color and texture are indexed using multi-dimensional indexing methods. As demonstrated above, pairs of simple objects can be indexed based upon spatial relationships.

In general, we need to define a content-based composite object query framework which does not rely on specific indexing structures but rather adapts to those indexes that are being used.

4. SPROC: SEQUENTIAL PROCESSING OF QUERIES

The SPROC query method consists of the following three procedures:

1. **Sub-goal ordering:** the composite object query is first linearized into a set of sub-goals. The sub-goals are assigned an order based upon available index structures and dependencies. As a result of this ordering, each of the sub-goals is assigned a role as either a retrieval function or a filter.
2. **Query block management:** for each sub-goal, processes are designated for retrieving blocks of the best L matches, second-best L matches, and so forth, and caching the results. The query block processing system then selects combinations of blocks to be processed by the SPROC algorithm.
3. **SPROC dynamic program:** the SPROC algorithm performs the best-first search in the trellis formed by the selected query blocks to determine the best K matches to the query.

4.1. Linearization of Composite Objects

Each linear composite object, such as the one shown in Fig. 2, is represented as a graph. Each node and arc in the graph represents an object component and a pairwise relationship, respectively. The composite object given in Figure 2 is a special class of composite objects without loops or branches. We will focus on this type of composite objects first, in which the translation into a linear graph is trivial.

We now consider a composite query which consists of content-based and logical sub-goals. For each composite query, we generate a sequential rule-goal program by ordering the sub-goals.

4.2. Sub-goal Ordering

For each composite query, we generate a logical rule-goal program. The rule-goal program defines the order in which the sub-goals are processed to answer the query. For example, consider the following query, Q ,

$$Q(A, B, C, D, E, F, G) \leftarrow \text{EAST}(B, D), \text{NW}(A, B), \text{SW}(B, C), \text{NEAR}(C, D), \text{COLOR}(A, E), \text{TEXTURE}(C, F), \text{SHAPE}(D, G).$$

We modify the bound-is-best assumption¹⁴ to include, bound-is-best, then fuzzy, then free. That is, sub-goals with bounded variables are moved to the front of the program, followed by those with fuzzy variables, then those with free. Using this approach, the sub-goals of query Q are evaluated in the following order, with free, bounded and fuzzy variable bindings given by [f], [b] or [z], respectively:

1. NEAR[--bb---](C, D)
2. COLOR[f---b--](A, E)
3. TEXTURE[--f--b-](C, F)
4. SHAPE[---f--b](D, G)
5. SW[-fz----](B, C)
6. EAST[-b-z---](B, D)
7. NW[zb-----](A, B).

4.3. Query Block Management

In general, since the database is large, it is difficult to evaluate the program directly. The second procedure decomposes each stage of the program into a list of query blocks. The query blocks are evaluated in conjunction with other query blocks by the SPROC algorithm. The query block management system handles the generation, caching and release of query blocks.

We define a query block as follows: given a sub-goal, i.e., $COLOR(A, B)$, the sub-goal is evaluated and results are returned in blocks as follows: the first best L , the second best L , the third best L , and so forth. For each sub-goal, the results are cached temporarily for use by SPROC.

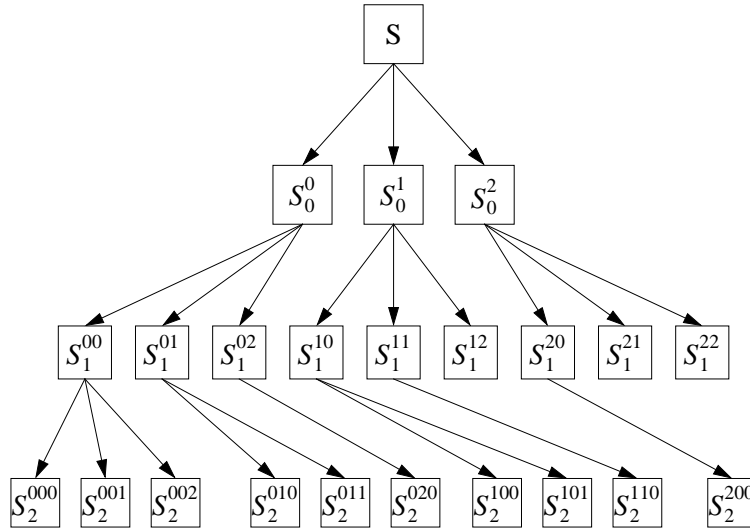


Figure 3. Query block management partitions the results of evaluating sub-goals into blocks of size L .

The query program consists of a sequence of sub-goals S_0, S_1, S_2, \dots , as illustrated in Fig. 3. The query block management system evaluates the sequence of sub-goals by retrieving blocks of size L . Initially, at step S_0 , the system generates blocks $S_0^0, S_0^1, S_0^2, \dots$. At step S_{i+1} , following path from block S_i^j , the system generates blocks $S_{i+1}^{j0}, S_{i+1}^{j1}, S_{i+1}^{j2}, \dots$.

The objectives of the query block management system are to provide paths for deeply descending towards the query answer while managing the back-offs required as the sub-goal evaluations need to be broadened. For example, the deepest descent is illustrated in path $S_0^0, S_1^{00}, S_2^{000}$, which uses the first best L matches from each of the sub-goals.

If this path does not answer the query, then the query block management system handles the back-tracking, i.e., the system next evaluates path $S_0^0, S_1^{00}, S_2^{001}$, then $S_0^0, S_1^{00}, S_2^{002}$, then $S_0^0, S_1^{01}, S_2^{010}$, and so forth.

In general, the system does not need to generate and evaluate all combinations of query blocks if they are evaluated in this best-first manner. Furthermore, the query block management system defines an overall strategy for generating query blocks as needed at different points in the query. For example, the following sequence illustrates the order in which the query blocks in Fig 3 are generated by the system:

$$S_0^0, S_1^{00}, \boxed{S_2^{000}}, \boxed{S_2^{001}}, S_1^{01}, \boxed{S_2^{010}}, S_0^1, S_1^{10}, \boxed{S_2^{100}}, \boxed{S_2^{002}}, \boxed{S_2^{011}}, S_1^{02}, \boxed{S_2^{020}}, \boxed{S_2^{101}}, S_1^{11}, \boxed{S_2^{110}}, S_0^2, S_1^{20}, \boxed{S_2^{200}}.$$

The blocks are generated, cached and released as needed. Each combination of blocks is passed to the SPROC algorithm, which provides a fast dynamic programming method for efficiently retrieving the best K matches from the set of blocks, while guaranteeing no false dismissals.

4.4. SPROC Completeness Bound

Given a combination of query blocks, the SPROC algorithm evaluates the sequence of sub-goals and retrieves the best K matches. The following two theorems show that SPROC does not need to evaluate all the combinations of component objects (query blocks) to locate the best K answers to the query.

Let L_A, L_B be two ranked lists. The elements of L_A and L_B are ranked by the similarity to A and B , respectively. The similarity score of the j^{th} member in L_A is denoted as $S_{L_A,j}$. The first member always has the highest similarity score. Furthermore, $S_{R_{AB},j}$ denotes the score of the j^{th} member in the ranked list R_{AB} between a member $a \in A$ and a member $b \in B$. Let $L_{A,K}$ and $L_{B,K}$ denote the partial lists which contain the top K ranked members.

Theorem 1: Assume that L and L' are generated as follows:

$$\begin{aligned} L &\leftarrow L_A \text{ fuzzyAND } L_B \text{ fuzzyAND } R_{AB}, \\ L' &\leftarrow L_{A,M} \text{ fuzzyAND } L_{B,M} \text{ fuzzyAND } R_{AB,M}, \end{aligned} \tag{4}$$

where $M \geq K$. Then $L_K = L'_K$ if

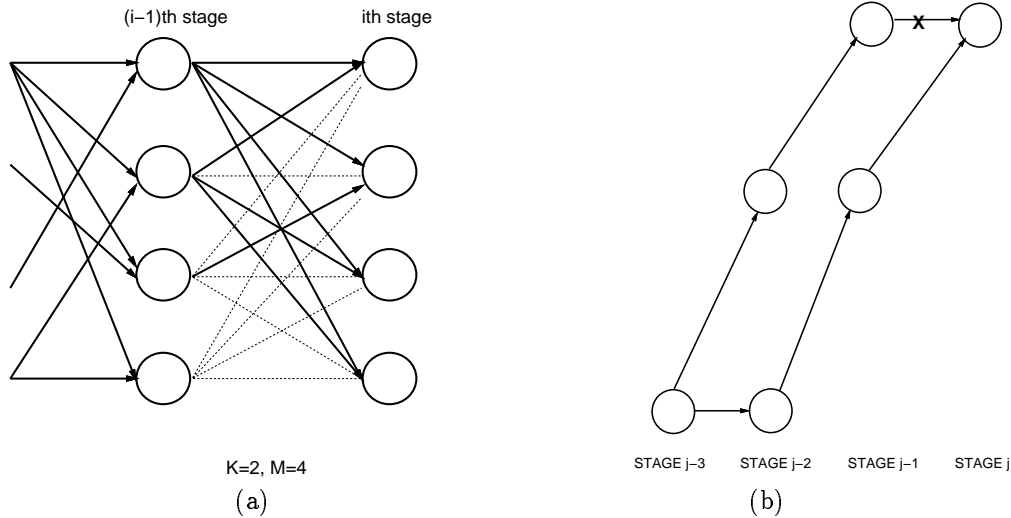
$$S_{L,K} \geq \max\{\omega_A S_{A,M+1} + \omega_B S_{B,1} + \omega_A B \omega_A S_{A,1} + \omega_B S_{B,M+1} + \omega_{AB}\},$$

given the fuzzyAND definition in Eq. 3.

This theorem is deduced from the fact that any members in L that involves the $(M + 1)^{th}$ member in A or B have a score no greater than either of them. Consequently, if $S_{C,K} \geq S_{A,M+1}$ and $S_{C,K} \geq S_{B,M+1}$, the members in C_K must come from A_M and B_M .

The above theorem is generalized to more than two ranked lists. In this case, an M exists such that the K best combinations extracted from the top M candidates from each list is the same as the K best combinations extracted from the entire list.

In the following, we will develop a sequential processing algorithm for the fuzzyAND definition in Eq. 3. This algorithm retrieves the best K combinations for a composite object consisting O simple objects, where $O \geq 2$.



4.5. SPROC Algorithm

SPROC Algorithm: In this algorithm, we assume that the composite object is represented as a linear graph (e.g. Fig. 2). Each component is represented as a *stage* in the trellis diagram, as shown in Fig. 4(a). Each combination of resulting answer is then represented as a *traversal path*. Let $D_{i,j,m}$ denote the similarity score of the j^{th} path, entering into the m^{th} object at the i^{th} stage, and each object component stores a total of K paths. Furthermore, $S_{i,m}$ denotes the similarity score of the m^{th} object at the i^{th} stage, and $S_{R_{i,j,m,n}}$ denotes the similarity score between the m^{th} object of the i^{th} stage and the n^{th} object of the j^{th} stage.

1. Initialize $D_{1,*m} = S_{1,m}$. Set $i = 2$.
2. Evaluate all the paths entering the component object m ($m \leq M$, where M is determined from Theorem 1) at stage i by using the following expression:

$$D_{i,j,m} = D_{i-1,j,n} \text{ fuzzyAND } S_{R_{i,j,m,n}} \text{ fuzzyAND } S_{i,m} \quad (5)$$

Note that a total of KM evaluations are required for each component object candidate.

3. For component object m , store a total of K paths with the largest $D_{i,j,m}$ among all the evaluated paths, and eliminate all the other survivors.
4. if $i = O$, Select K paths with the largest $D_{O,j,m}$ from KM candidate paths, stop. Otherwise, $i = i + 1$, go back to step 2.

The major issue of this algorithm is whether a path which should have survived at the final stage is eliminated earlier. This is impossible, as shown below.

Theorem 2: The final survivors in the SPROC algorithm have the maximum similarity scores.

We prove this theorem by contradiction. Assume that one of the K paths with the highest similarity scores is eliminated by the algorithm at stage j , as illustrated in Fig. 4(b). This implies that the partial path metric of the survivor exceeds the one which should have been chosen.

If the remaining portion of the path is appended to the survivor at stage j , the total metric in the end will exceed the total metric of the one that should have been chosen (assuming that the weighted sum is used to implement the fuzzyAND). But this contradict the definition of the maximum similarity score paths. Hence the maximum similarity paths cannot be eliminated by the algorithm.

4.6. Composite Objects with Loops and Branches

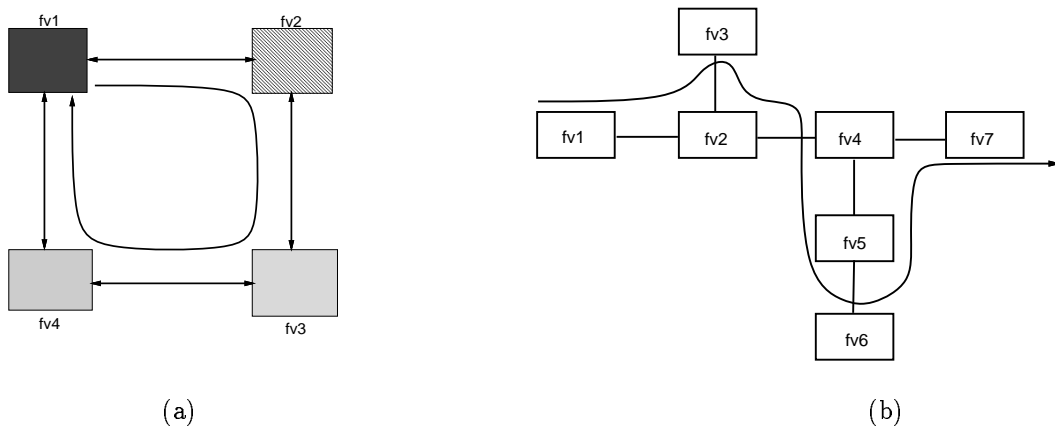


Figure 5. (a) Example of a circular composite object, and (b) an example of a composite object with branches.

When a composite object is represented as a graph with either loops (Fig. 5(a)) or branches (Fig. 5(b)), the algorithm established in the previous subsection can still be utilized. After extracted a minimum spanning tree from the original graph, a depth-first or a breadth-first traversal of the spanning tree can be used to translate the original graph into a linear representation.

The linear representation of Fig. 5(a) is shown in Fig. 6(a). It consists five object stage with one of the object stage (fv1) being visited twice.

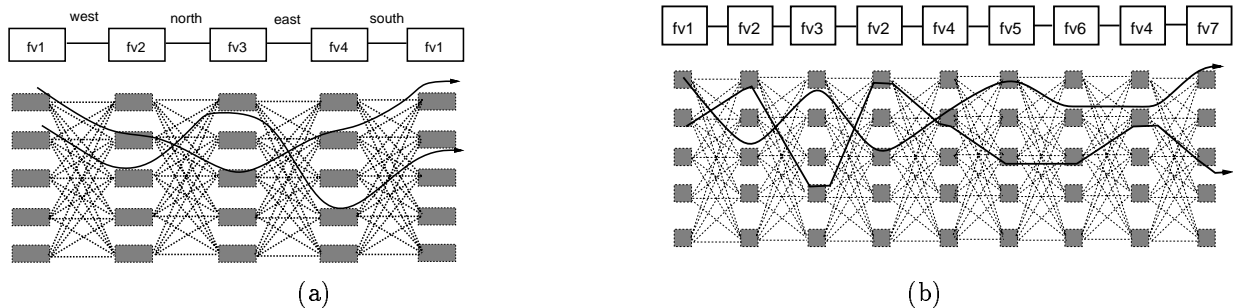


Figure 6. Examples of the retrieval of composite objects in the trellis.

The translation result of Fig. 5(b) is shown in Fig. 6(b), in which two of the objects are visited twice. Duplication of the object stage is necessary in both cases in order to preserve the pairwise relationships that is needed in object retrieval, to be discussed later.

There is additional complication for handling these situations. This is due to the fact that the best- K survivors are always searched following a single thread. Using Fig. 6(a) as an example, the best K paths into the last stage, which is a replication of the first stage, might not be consistent with the original selection at the beginning stage for object component fv1.

The necessary modification of the SPROC algorithm is that in step 2, a self-consistency check is performed when evaluating all the paths entering into a node, and eliminate those paths which are inconsistent with those chosen earlier.

5. EXAMPLES

We illustrate an example of the execution of the SPROC search method. In this case, the best-first path to the final sub-goal is initially explored with no candidates surviving. In this case, the content-based sub-goals are loosened, i.e., the next set of query blocks (next best L matches) are retrieved, in order to generate more candidate composites. In this case the system back-tracks and starts the sequential program with the new candidates.

We illustrate an example sequential program table, which consists of seven stages. Stages 1, 5, 6, 7 correspond to logical sub-goals, while stages 3, 4, 5 correspond to fuzzy or content-based sub-goals. The similarity scores $S_{i,m}$ that are computed for the i^{th} object in the m^{th} stage are given below, where x denotes truth value of 1.

Composite	1	2	3	4	5	6	7
$a_0 b_0 c_0 d_0$							
$a_0 b_0 c_0 d_1$	x	0.5	0.4	0.1	x	x	0
\vdots							
$a_i b_j c_k d_l$	x	0.5	0.1				
$a_i b_j c_k d_{l+1}$	x	0.4	0.7	0.2	x	x	x ← Answer
$a_i b_j c_k d_{l+2}$	x	0.1					
\vdots							
$a_{N-1} \dots$	x	0.4	0.3	0.2	0		

As the sub-goals are evaluated, candidate composites are eliminated by the logical rules and are scored by the fuzzy rules. The scores are used to prioritize evaluations of candidate composites in the sequential program. The

candidate with the highest similarity score is examined first. This process is repeated until the last sub-goal is reached. The surviving candidate with the highest similarity score is the best match. This process guarantees that the global best match is found.

6. SUMMARY AND DISCUSSION

In this paper, we propose a new sequential query processing algorithm for evaluating content-based composite object queries. The composite objects consist of spatial and temporal arrangements of simple objects. The simple objects are defined in terms of spatial, temporal, feature and semantic attributes. The query method defines a process for executing a best-first search for the matches to the query, while providing a flexible framework for broadening the search space as required. The query method guarantees that there are no false dismissals of the candidate composite objects.

ACKNOWLEDGEMENTS

This work was funded in part by NASA/CAN contract no. NCC5-101.

REFERENCES

1. S.-K. Chang, Q. Y. Shi, and C. Y. Yan, "Iconic indexing by 2-D strings," *IEEE Trans. Pattern Anal. Machine Intell.* **9**, pp. 413 – 428, May 1987.
2. V. N. Gudivada and V. V. Raghavan, "Design and evaluation of algorithms for image retrieval by spatial similarity," *ACM Trans. on Information Systems* **13**, April 1995.
3. E. Oomoto and K. Tanaka, "OVID: Design and implementation of a video-object database system," August, vol. 5 1993.
4. J. R. Bach, C. Fuller, A. Gupta, A. Hampapur, B. Horowitz, R. Humphrey, R. C. Jain, and C. Shu, "Virage image search engine: an open framework for image management," in *In Storage and Retrieval Storage & Retrieval for Still Image and Video Databases IV*, vol. Proc. SPIE 2670, pp. 76 – 87, IS&T/SPIE, 1996.
5. A. Pentland, R. W. Picard, and S. Sclaroff, "Photobook: Tools for content-based manipulation of image databases," in *Proceedings of the SPIE Storage and Retrieval Image and Video Databases II*, February 1994.
6. M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker, "Query by image and video content: The QBIC system," *IEEE Computer* **28**, pp. 23 – 32, September 1995.
7. C.-S. Li, V. Castelli, and L. Bergman, "Progressive content-based retrieval from distributed image/video databases," in *Proceeding of the International Symposium of Circuit and System*, IEEE, 1997.
8. J. R. Smith and S.-F. Chang, "Integrated spatial and feature image query," *ACM Multimedia Systems journal*, 1997. To appear.
9. A. Soffer and H. Samet, "Retrieval by content in symbolic-image databases," in *Symposium on Electronic Imaging: Science and Technology – Storage & Retrieval for Image and Video Databases IV*, vol. 2670, pp. 144 – 155, IS&T/SPIE, 1996.
10. S.-F. Chang, W. Chen, H. Meng, H. Sundaram, and D. Zhong, "VideoQ: An automated content based video search system using visual cues," in *Proc. ACM Multimedia '97*, ACM, November 1997.
11. S.-S. Chen, "Content-based indexing of spatial objects in digital libraries," **7**, pp. 16 – 27, March 1996.
12. E. G. M. Petrakis and C. Faloutsos, "Similarity searching in large image databases," Tech. Rep. 3388, Department of Computer Science, University of Maryland, 1995.
13. N. Stroble, C.-S. Li, and V. Castelli, "MMAP: Modified maximum a posteriori algorithm for image segmentation in large image/video databases," in *Proc. ICIP-97*, IEEE, 1997.
14. J. D. Ullman, *Principles of Database and Knowledge-Base Systems; volume II: The New Technologies*, Computer Science Press, Rockville, MD, 1989.