Constrained Querying of Multimedia Databases: Issues and Approaches

Apostol (Paul) Natsev[a*†] John R. Smith[b] Yuan-Chi Chang[b] Chung-Sheng Li[b]
Jeffrey S. Vitter[a]

[a]Department of Computer Science, Duke University, P.O. Box 90129, Durham, NC 27708.
Email: {natsev, jsv}@cs.duke.edu.
[b]IBM T. J. Watson Research Center, 30 Saw Mill River Road, Hawthorne, NY 10532.
Email: {jsmith, yuanchi, csli}@us.ibm.com.

ABSTRACT

This paper investigates the problem of high-level querying of multimedia data by imposing arbitrary domain-specific constraints among multimedia objects. We argue that the current structured query model, and the query-by-content model, are insufficient for many important applications, and we propose an alternative query framework that unifies and extends the previous two models. The proposed framework is based on the querying-by-concept paradigm, where the query is expressed simply in terms of concepts, regardless of the complexity of the underlying multimedia search engines. The query-by-concept paradigm was previously illustrated by the CAMEL system. The present paper builds upon and extends that work by adding arbitrary constraints and multiple levels of hierarchy in the concept representation model.

We consider queries simply as descriptions of virtual data sets, and that allows us to use the same unifying concept representation for query specification, as well as for data annotation purposes. We also identify some key issues and challenges presented by the new framework, and we outline possible approaches for overcoming them. In particular, we study the problems of concept representation, extraction, refinement, storage, and matching.

Keywords: Constraints, concepts, CAMEL, content-based query, images, MPEG-7, multimedia

1. INTRODUCTION

The advances in computing power over the last few years have made a considerable amount of multimedia data available on the web and in domain-specific applications.[1] The result has been an increasing emphasis on the requirements for searching and describing such large repositories of multimedia data.[2] One of the first realizations about this new field was the fact that the traditional database model for querying of structured data does not generalize well to multimedia domains of unstructured data. One reason was that in databases, the results were always well-defined and unordered sets (i.e., each item was either in or out of the result set), while in multimedia queries, the results were fuzzy and typically ordered by their degree of similarity to the query. Another difference was in the way the data was queried. While the relational database model treated all items as attribute sets and consequently all queries were on the attributes, it was not apparent how to convert a sample image query into an attribute query in a meaningful and simple fashion. This led to the widespread adoption of the query by example (or query by content) model for image search, where the user would specify a sample image and the system would return images that are similar in color, shape, texture features, etc.[3–9]

The above query model has some advantages, such as comparison and ranking of images based on objective visual features, rather than on subjective image annotations, for example; and automated indexing of the image data, as opposed to labor consuming manual image annotation. However, this model has its drawbacks as well. For one, in some applications it is difficult to find an appropriate query sample from the same domain, or it may be difficult to provide it to the query engine. For example, if the user is looking for images on the Internet that are similar to a specific image, there is no way to upload the query image to the search engine. In most cases, he/she would have to do an extra text search step in the image annotations (file name, URL, surrounding text context, etc.), in order to find suitable image candidates to start the content-based search. However, even then, they may not find a suitable query image, and this reduction of the image search problem to a text search one is only a

*CONTACT AUTHOR.
†This work was done while the author was visiting IBM T. J. Watson Research Center.

workaround rather than a permanent fix for the problem. In other domains, there may not even be available annotations, or it has to be done manually, which makes the approach difficult to scale.

The more important drawback of the query-by-content model, however, is the fact that there is no intuitive way of describing multimedia data, and therefore there is a large gap between the user's perception of the data and the way it is actually stored and manipulated. In the text domain this is not a problem because the user perceives documents as a collection of words and this is typically how the documents are represented. A search based on keywords therefore makes perfect sense, both from the user's perspective, as well as from the system design point of view. However, in the image domain, for example, the physical image representation is very different from the mental model of the user. Therefore, the system does not really allow the user to clearly express what he/she is looking for. That ambiguity in the query specification reduces usability and naturally translates into poor retrieval effectiveness, leading to frustration on the user's part.

In light of the above, we have argued in favor of a different, *concept-based query model*, where we introduce an extra layer between the user's perception and the system's internal representation, captured in the form of semantic concepts.[10–12] In that model, the user would specify a high-level concept for each query, while the system would translate said concept into low-level features and would perform an enhanced content-based query. On the one hand, the model allows the user to reuse a rich set of predefined semantic concepts without having to define or express them explicitly. On the other hand, it provides a flexible mechanism for combining such concepts into even higher level composite concepts via inter-concept relationships (or constraints), thus enabling the user to express powerful queries with an interface close to his mental model.

The current paper introduces a generalized query framework in the spirit of concept-based querying and proposes a rich representation for the concepts. In addition to improving usability, the proposed framework enhances the query power due to the support of complex constraints. Realizing that the concept representation is the most critical component of any concept-based framework, we try to formulate the most general concept representation that can be supported efficiently by the system. We therefore focus our attention mainly on the problem of supporting arbitrary attributes and constraints in the definitions of concepts. The paradigm of concept-based querying itself is illustrated by the CAMEL system,[10] for example. The present paper builds upon that work by adding constraints among the concepts, and focusing on the constraint specification part for arbitrary domains.

The rest of the paper is organized as follows. In the remainder of this section, we try to explain and motivate the problem of constrained querying, describe an overview of the proposed framework, and list relevant work, as well as our contributions. In the next section, we give a more detailed explanation of the framework architecture and the different modules. We then talk about the concept representation we propose, as well as methods for matching such representations. In the following section, we discuss some of the issues that are relevant to the proposed framework and we present some possible approaches for dealing with those issues. Finally, we conclude with directions for future work and summary of contributions.

1.1. Problem Formulation and Motivation

Consider the following application scenario for oil exploration. A company is interested in using image analysis techniques for locating potential oil reservoirs. The data consists of rock structure images taken at different depths from various drill samples. The hypothesis is that a particular composition of certain rock formations is a good indication for potential oil reservoirs. The reasoning is that when a relatively soft layer of earth is located above and near an impermeable layer of rock, there are good conditions for oil formation. To that purpose, the company would like to find occurrences of both rock types, where the first one is on top of the second one. This example is illustrated in Figure 1.

The above scenario is an example of combining the results of two fuzzy searches using constraints that are both crisp (rock A is *above* rock B) and fuzzy (rock A is *near* rock B). We call such queries fuzzy joins with crisp and fuzzy constraints. According to the type of search results to be joined (crisp or fuzzy) and the type of constraints used for the join (again crisp or fuzzy), we can have several join combinations. Of those, the current database systems typically support only the crisp joins with crisp constraints. When it comes to introducing any fuzziness, such database systems are very inefficient at best, or completely unusable, at worst. The multimedia search systems, on the other hand, support fuzziness in the searches themselves but do not really allow any joins of the search results, or support them to a very limited and domain-specific extent.

Another feature that has very limited support in both traditional and multimedia databases is the definition of nested (or recursive) views of fuzzy result sets. For example, building on the above application scenario, we can actually define the concept of *DELTA_LOBE* as a sequence of *SANDSTONE* on top of *SHALE* on top of *SILTSTONE*. The concepts of *SANDSTONE*, *SHALE*, and *SILTSTONE*, can be recursively defined by specifying samples of rock textures that fall in the corresponding class. Using the previously defined views for *SANDSTONE*, *SHALE*, and *SILTSTONE*, the user might want to define *DELTA_LOBE* view using the following SQL statement:

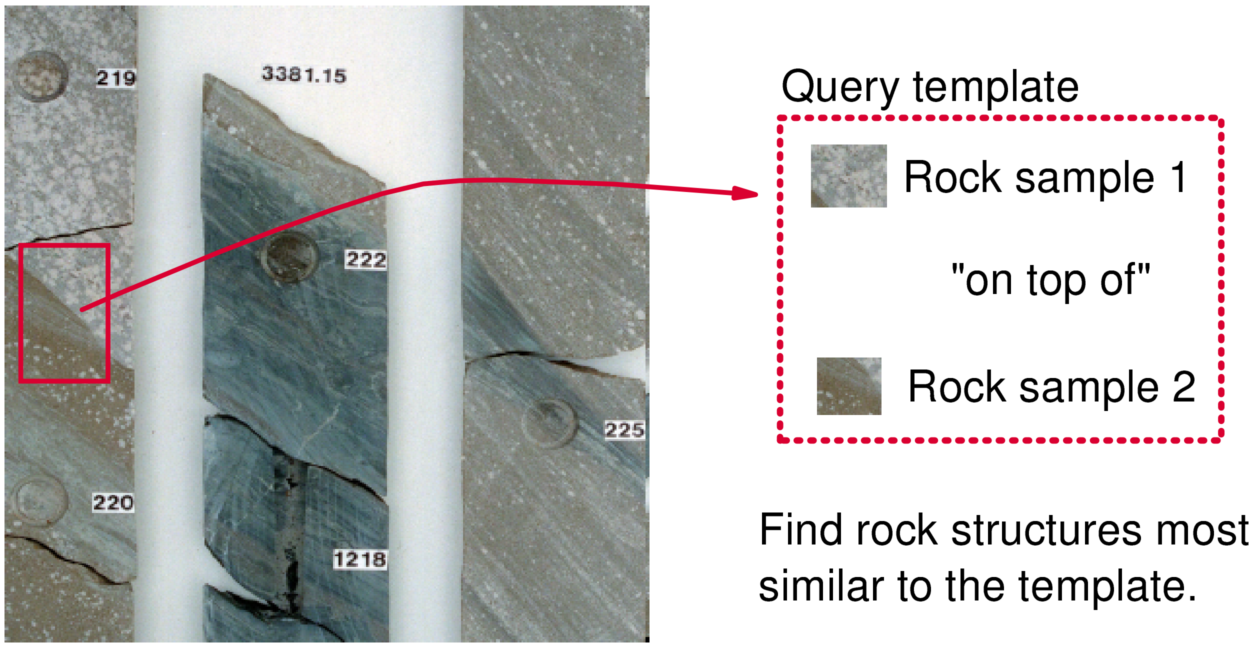

Figure 1. Example of a fuzzy join

CREATE VIEW *DELTA_LOBE* **AS**
SELECT $*$
FROM *SANDSTONE SD, SHALE SH, SILTSTONE SL*
WHERE ABOVE(*SD.DEPTH, SH.DEPTH*) = 1 **AND**
ABOVE(*SH.DEPTH, SL.DEPTH*) = 1 **AND**
NEAR (*SD.DEPTH, SH.DEPTH*) = 1 **AND**
NEAR (*SH.DEPTH, SL.DEPTH*) = 1

Even though conceptually there is nothing new in this definition, in practice it is very hard to support such nested fuzzy views efficiently. The reason is that due to the fuzziness, getting even a single candidate from the top view may involve scanning all candidates of the child views, which would be very time consuming. The reason is that a naive algorithm for determining the best match from the top view would require examining all possible combinations of matches from the children views in order to compute their scores.

On the other hand, there are numerous application scenarios that, like the one above, can benefit greatly from fuzzy searches, fuzzy joins, and fuzzy nested views. In addition to searching, such technology can be used for filtering, annotation, classification and inferencing purposes, among others. For example, selecting a proper helicopter landing space for a forest fire may involve a satellite image search with constraints on landing space size and flatness, proximity to water sources and to the fire itself. Or, the decision to buy a house in a certain area may include criteria about proximity to schools or hospitals, termite populations, as well as demographic factors. Ideally, all of these criteria should be incorporated into a single query where the system will perform separate queries to different sub-systems and will integrate the results according to the specified criteria. The range of application domains that can benefit from a single unified view that incorporates both fuzzy and crisp queries, as well as their joins and nested views, include, among others:

- Entertainment videos (e.g., movies, news and sports feeds, etc.)
- Aerial, satellite, seismic, or medical imagery
- Time series and stock marked data
- E-commerce for personalized virtual product catalogs
- Music shopping

1.2. Proposed Approach and Contributions

In this paper we address the above problems by proposing a new framework that unifies searches, joins, and nested views, allowing any of them to be crisp or fuzzy. Our framework combines and generalizes both of the previous query models for structured and non-structured data. It is based on the query-by-concept paradigm, where concepts are essentially generalized views and are represented by a fuzzy hierarchical graph data structure.

The specific details of the concept representation are discussed in Section 3 and the design is general enough that it allows arbitrary user-defined attributes, constraints, and matching algorithms to be plugged in the framework. The same concept representation is used for describing both queries and data so there is a unified query language and data annotation language. The internal concept representation can be mapped to and from SQL and XML. In fact we propose to use the emerging MPEG-7 meta-data standard (based on XML) as the interface to the concept representation. By adopting MPEG-7 as the concept definition language, we automatically inherit tools for expressing, manipulating, interpreting, and querying MPEG-7 descriptions. For example, the MPEG-7 Visual Annotation Tool[13] can be used as a visual query interface to the proposed framework.

In addition to proposing the unified query framework, we also investigate the key issues and challenges that arise in connection to the framework and the concept representation. Such issues include concept extraction, or automated concept learning, storage, compression, and matching of concepts, as well as selectivity estimation of such general queries.

1.3. Related Work

In this section, we survey some prior art and its relations to the proposed query framework model and its components. Since the primary target for our proposed framework is image querying, we first describe the major existing image query systems, along with some of their advantages and disadvantages. Consistent with our approach of concept-based querying, we then consider different knowledge representation schemes that could be used for concept representation purposes. Finally, we describe some work on designing efficient algorithms for joining fuzzy result sets.

1.3.1. Multimedia querying

The problem of searching multimedia data has received considerable attention in the last few years. The early image query systems, such as IBM's QBIC,[14,15,3] MIT's Photobook,[16,5] the Virage system,[4] etc., were invented in the early to mid 90s. Those systems typically took an image or a sketch as input, computed some visual features from it (e.g., color histograms, texture, shape features), and searched one or more indexes to return images with similar features. Alternatively, the user could specify values for these features and appropriate weights reflecting their relative importance. Being the first to use content-based search, those systems were a big improvement over the method of manually annotating images and doing a text search to find relevant ones.

Researchers then focused on studying other types of features, such as wavelets,[6–8,17,9] or localizing the features to sub-image regions.[8,9,18] By segmenting the images into their regions, or objects, and comparing images at the object level, such systems got closer to the mental model of the user of how image similarity is established. They were also able to exploit not only visual feature similarity but also the relationships among the image objects, such as their spatial arrangements, for example. However, the types of relationships, or constraints, were limited by the indexing methods used. The approach used in these systems was that the system should compute the relevant constraints among the objects transparently to the user, and therefore the user had little control over the types of relationships used. In this paper, we adopt the opposite approach that the user should have full control and be able to specify arbitrary inter-object constraints.

Recently, we have also begun advocating a new query-by-concept paradigm, which separates the user query interface from the inner workings of the underlying query engine(s).[10–12] The new intermediate level makes the mechanics of the actual search transparent to the user, without limiting the user's power in specifying relevant attributes and constraints. The system provides a uniform and natural API for specifying query concepts, which are then internally translated to the specific search engines' specifications. This modularized design provides a simplified interface and improved usability, while preserving, or even enhancing, the power of content-based search. Related works in this context are the CAMEL image query system[10] and the multimedia thesaurus system MediaNet.[11,12] This paper is another effort in this direction.

1.3.2. Knowledge representation

One of the critical components of a concept-based query system is the concept representation. On the one hand, if it is too simple, the whole framework would be too limited and useless for many applications. On the other hand, an overly complex representation would not be supported natively by many search systems, which will make the entire query process very inefficient. In search of the right balance, we hereby consider some previous knowledge representation schemes.

Attributed Relational Graphs (or ARGs) are one form of representation used previously to capture inter-object constraints by Petrakis and Faloutsos.[19] The nodes in ARGs correspond to objects, while the edges represent their relationships. ARGs are very similar to semantic networks, which were used in Artificial Intelligence and computer vision.[20,21] Both of these representations are very good candidates for our purposes, however, they lack an appropriate hierarchical structure necessary for support of nested fuzzy views, and they do not have a suitable mechanism for specifying representation parameters that can be used for learning or extraction purposes, or to tune a representation to a particular data set, for example.

Knowledge representations that include such parameters in the form of weights or conditional probabilities include neural networks[22] and belief networks (or Bayesian networks). Both of them have nodes representing particular states, and weights or probabilities corresponding to the transition between these states. Neural networks were not entirely suitable for our purposes because their intermediate states do not correspond to any intuitive concepts but are there simply for auxiliary purposes. In Bayesian networks, on the other hand, all states are well-defined and have a clear interpretation or meaning that can be translated into a concept. However, the parameters on the state transitions actually correspond to conditional probabilities, while for our purposes, we only need weights to express relative importance of the inner nodes or concepts. We have therefore used a mixture of the above representations that uses a hierarchical fuzzy graph data structure and is presented in detail in Section 3.

1.3.3. Fuzzy join methods

There are few problems related to constrained querying and fuzzy joins that have been studied before. Even though to the best of our knowledge, there are no previous publications focusing exclusively on the support for domain-independent constraints, every database query system or a search engine supports filtering by constraints to a certain extent. Typically, the support extends to scalar predicates that can be evaluated for each database tuple independently of other tuples, as well as some form of aggregate predicates, where the value of the predicate for a given item depends on the other items in the database as well (e.g., nearest neighbor or top-k queries). Indexing support is usually limited for such queries and the execution becomes even more inefficient when several constraints are combined.

Spatial and attributional constraints, for example, have been integrated previously by Petrakis and Faloutsos in,[19] where the authors proposed a method for searching medical images according to their attributed relational graphs (ARGs). The method relies on the assumption that certain objects are contained in all images from a given domain (e.g., heart, lungs, etc.) in addition to a variable number of objects that are unexpected (e.g, tumor). The specific structural relationships among such objects are stored in ARGs and are used for computing image similarity. The above assumption does not generalize or scale well, however, and the indexing method is fairly ineffective if the number of unlabeled objects is large (which is typically the case for most domains). Other methods for encoding spatial relationships are the 2-D strings, which capture the relative position of objects with respect to the other objects in the set.[23] Smith and Chang developed a query processing system for spatial and feature-based image queries,[8] where the retrieval was based on specifying absolute or relative position constraints on the objects in the image. However, the proposed pipeline and parallel processing methods do not guarantee the lack of false dismissals.

The problem of supporting Boolean combinations on multiple ranked independent streams was considered first by Fagin for the case of top-k queries.[24,25] Ortega et. al.[26] studied a similar problem of combining multiple similarity scores for the same document in a weighted fashion so that the overall score is a combination of how well the document matched the query with respect to both text and image parts. In their MARS database system, they defined a query tree whose nodes represented intermediate matches derived from the matches at the children nodes,[27] and evaluated it from the bottom up, arriving at the final similarity score at the root. Both approaches, however, considered only Boolean conjunctions and disjunctions as the join constraints, and only a single level of a join hierarchy. The MARS approach did use a multi-level query tree but the different levels were derived simply by breaking up a single join level of multiple streams into a hierarchy of pairwise joins. Neither approach therefore considered nested fuzzy views or fuzzy joins with arbitrary constraints.

The SPROC algorithm[28] was the first algorithm, to the best of our knowledge, that addressed the problem of joining fuzzy result sets by means of specifying arbitrary fuzzy constraints among them (see Figure 1 for example). It allows the user to express the query as a set of sub-queries (nodes in a graph), along with fuzzy constraints among them (edges between the nodes). The problem is solved by looking for a maximal cost path, computed with a Vitterbi-like algorithm. Based on certain assumptions, the algorithm allowed both pruning of some edges, as well as some nodes. The assumptions were that no nodes participate in more than one constraint and that there were no cycles. This method was the most general of the described methods but was still designed for a single hierarchy level.

Our framework is similar to these ideas but generalizes them to arbitrary domains, allows fuzzy matching and scoring, and very importantly, introduces concept hierarchies that can be reused. Previously, we proposed the use of primitive concepts for querying of images, where the concepts were defined by simple aggregation of wavelet features extracted from sample images.[10] The current paper extends that work by adding arbitrary constraints and complex hierarchies of semantic concepts.

2. FRAMEWORK ARCHITECTURE

The design of the proposed framework is illustrated in Figure 2. It consists of four library modules that collectively are used to define concepts. Once defined, concepts can be used in a variety of applications, such as querying, filtering, annotation, classification, etc. Thus, the concepts form an intermediate layer between the user's application and the actual search engines. The system uses a set of APIs to exploit not only data from heterogeneous sources but also different matching (or join) algorithms and arbitrary constraints, including user-defined ones.

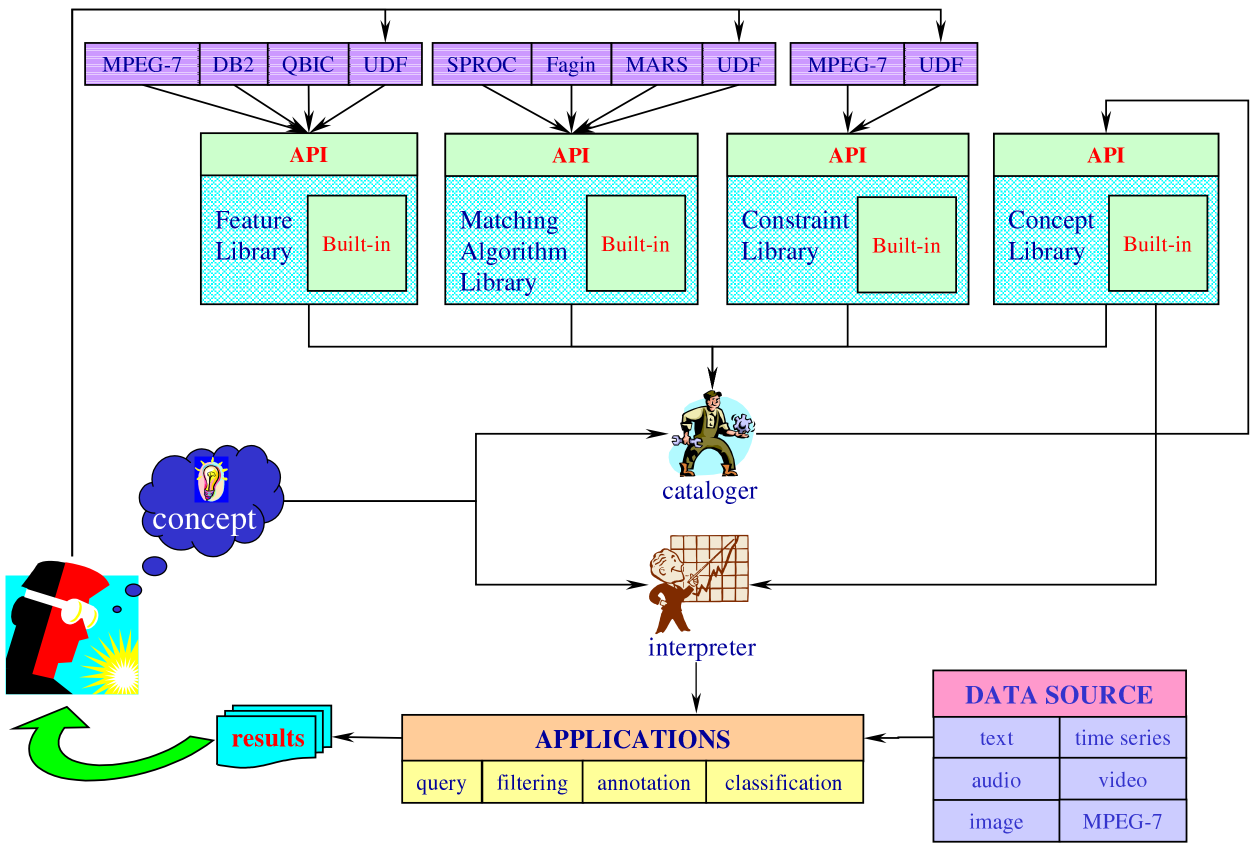

Figure 2. Architecture of the proposed framework

The feature library contains a set of features, such as color histograms, textures, shape, motion, etc. These features can be thought of as concept attributes and can come from various sources (traditional databases, multimedia search engines, MPEG-7 annotated data, etc.). The types of features correspond to MPEG-7 Descriptors and Description Schemes, or can be user defined through the API. In a way, the feature library reflects the capabilities of the underlying search engines. For example, if the system uses QBIC to extract a particular type of texture feature, then that feature would be present in the library so that concepts can reflect searches on that feature. Treating all features in a unified manner provides more power to the system because concepts can use the capabilities of multiple back-end search engines transparently to the user.

The constraint library is simply a collection of constraints, or relationships, defined on a single feature, or attribute. Examples include spatial or temporal constraints, feature similarity constraints, etc. The library has an API so that users can define their own constraints by specifying the attribute they apply to, as well as the number of arguments (or concepts) that they relate.

The matching algorithm library consists of a set of join algorithms that have the same interface. They all take an ordered set of matches for a set of children concepts, and compute an order set of matches for the parent concept defined through the children ones. In the process, all algorithms make sure that any constraints defined in the parent concept are met when combining the results from the child concepts. Examples of such join algorithms include Fagin's algorithm,[24,25] the MARS query tree algorithm,[27] and the SPROC algorithm.[28] While the first two do not really work with any constraints but simply provide Boolean conjunction and disjunction, the latter allows arbitrary fuzzy constraints. Other algorithms can be specified as long as they conform to the established interface.

Finally, the concept library contains a list of pre-defined concepts, either built into the system, shipped as concept library plug-in modules, or previously defined by users. The process of defining new concepts is performed by the Concept Cataloger module, and consists of the following steps: 1) select a set of concept features from the feature library; 2) select a set of child concepts from the concept library; and 3) select a set of constraints on the child concepts. Any of the above sets can be empty. The combination of the specified child concepts, along with the relationships among them, form a new concept with the specified attributes. That concept is then inserted in the concept library for later reuse. Given the set of pre-defined concepts, an application can then use a concept-based interface, where any input concepts are looked up into the concept library by the Concept Interpreter, and translated into the corresponding features, sub-concepts, and constraints. Those components are then used to form queries to the underlying search systems, and the results are aggregated back using the specified join algorithms, and presented to the user. The Concept Cataloger, Concept Interpreter, and the Concept Library have essentially the same functionality as the corresponding modules in the CAMEL system. However, due to the more complex concept representation model, here they are slightly more sophisticated and make use of the added Feature Library, Constraint Library, and Matching Algorithm Library modules.

3. CONCEPT REPRESENTATION

In this section, we formalize the notion of concepts and present the particular concept representation that we propose. We use a recursive definition, where each concept is defined as a triple $\langle \mathcal{C}, \mathcal{A}, \mathcal{R} \rangle$, with $\mathcal{C}$ denoting a set of children concepts (or objects, such as *car*), $\mathcal{A}$ denoting a set of attributes (or features, such as *color, shape, texture,* etc.), and $\mathcal{R}$ corresponding to a set of relationships (or constraints, such as *left-of* or *near*) among elements of $\mathcal{C}$. When $\mathcal{C}$, and therefore $\mathcal{R}$, are the empty sets, we call the resulting concepts *primitive*, and otherwise we call them *composite*. Intuitively, primitive concepts represent basic multimedia entities, such as image regions (defined by spatial location and shape, or by homogeneity constraint on a color or texture attribute), video segments (specified by a time interval in the video sequence), etc. Composite objects (or higher-level semantic concepts) are then derived from lower-level ones by imposing inter-object constraints on them.

The concept representation we propose for capturing the above definition is illustrated in Figure 3. It is a hierarchical fuzzy graph data structure that resembles a tree of graphs. Nodes in the representation correspond to concepts, and contain attributes as well as constraints. There are two types of edges in the data structure: aggregation edges, representing parent-child relationships, and association edges, representing inter-sibling constraints. Note that for simplicity, the association edges only represent binary relationships although the actual representation allows n-ary constraints in general. The aggregation edges also have associated weights, reflecting the relative importance of the child concepts in the parent's definition. We could add weights to the association edges as well but for now we will treat all constraints as being equally important.

Figure 3 depicts a sports video domain example of scoring a goal in a soccer game. We have defined the concept as a sequence of three events: a player kicking the ball, the goal-keeper not catching the ball, and finally the ball entering the goal. The latter events are in turn concepts defined recursively by means of the primitive concepts *player, ball, goal, goalie*. The relationships that make sense in this scenarios are a temporal ordering of the intermediate-level events, as well as several spatial, scale and motion constraints on the primitive concepts (i.e., ball *moving away* from player, ball *inside* the net, etc.). Overall, there is a time constraint on the total length of the sequence (the three events need to occur in a relatively short period of time for them to be significant). We could also express the last constraint as a sequence of proximity constraints on the time stamps of the three events. Finally, the weights on the edges represent the relative importance of the concepts with respect to their siblings. We note that in the definition of the top constraint, none of the first two child concepts are absolutely mandatory. For example, a player may head the ball into the net rather than kicking it, or the goalie may not even be near the ball at the time of a kick so that there would be no attempt to catch the ball. The only required event is that the ball enters the net, and it therefore has the highest weight of the three child events. The other two are appropriately weighed so that the farther away an event is from the critical scoring event, the less weight it has. On the other hand, in the recursive definition at the leaf level, all primitive concepts are necessary for the definition of the parent ones and are therefore weighted as equally important.

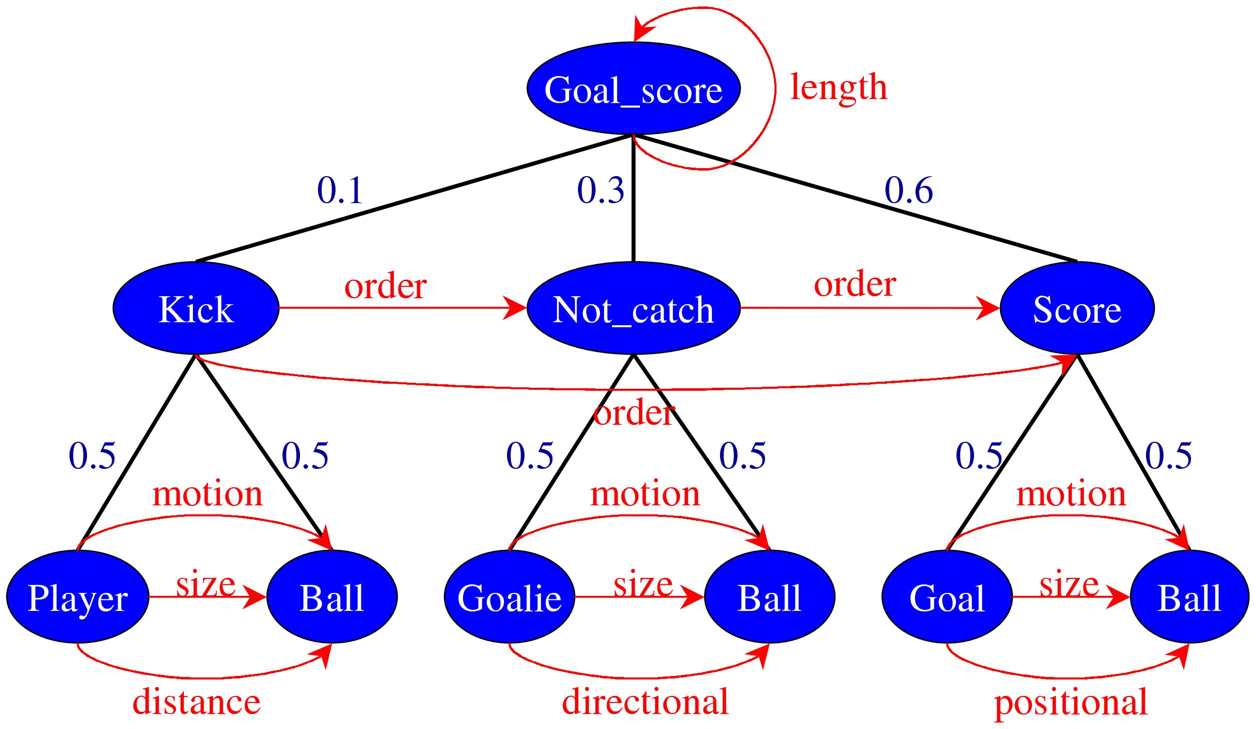

Figure 3. Example of a concept representation from the sports video domain.

4. RESEARCH ISSUES

In this section, we investigate several research issues that arise in connection with the proposed framework—including the study of a constraint taxonomy, representing concepts by constrained compositions, extracting and learning concept definitions, and supporting constrained queries efficiently.

4.1. Constraint Taxonomy

The first aspect in evaluating the usefulness of the proposed framework and its concept representation includes studying the class of semantic concepts that can be expressed as a combination of other concepts along with a set of constraints defined on them. In order to facilitate the process of mapping semantic notions into primitive or composite concepts, we study the types of constraints that can be imposed. We therefore consider a taxonomy of such constraints from a variety of applications and domains. For example, classifying constraints by type, we differentiate between topological constraints (such as inclusion/exclusion, union/intersection, negation, etc), directional (e.g., before/after, left/right, above/below), and metric (e.g., close/far, 0.1 mi away, 5 minutes before, etc.). According to precision, constraints can be classified as fuzzy (or generic), such as soon, close, etc., and crisp (or exact), such as “within 1 hour”, “5 meters away,” etc. Also, we can group constraints by dimensionality, and distinguish between 1-dimensional (e.g., left of, before, smaller than, close to, etc.), 2-dimensional (e.g., 30 degrees North-West of, in the top-left quadrant, etc.), or general multi-dimensional constraints (e.g., hyper box intersection, within certain distance in a k-dimensional space, etc.). Note that for a fixed dimensionality, the types of constraints are fundamentally the same, regardless of the domain they are applied to. Thus, the set of canonical 1-dimensional constraints are described, for example, by Allen’s relationships which map to different domain-specific constraints—for example, the “before/after” temporal relationship is topologically equivalent to the “left-of/right-of” spatial relationship. This simplifies the support of arbitrary constraints by pre-defining a certain set of canonical relationships for each dimension that the user can then map into specific domains. Finally, according to their domains, we can classify constraints into Boolean, spatial, scale, temporal, time series constraints, etc., with applications in the image, video, audio, stock data domains, etc. For example, the constraint that object A is 2 mi North West of object B is a Boolean combination of a 2-dimensional crisp metric constraint and a 2-dimensional fuzzy directional constraint. Having such a taxonomy in mind, it may be easier to determine if a particular concept can be decomposed into a set of child concepts and their relationships.

4.2. Concept Extraction and Refinement

Another issue related to the proposed framework is learning the constraints needed to define a given concept and extracting the parameters of these constraints from sample data sets. So far, we have proposed a particular concept representation but we have not said how it is to be constructed. One approach is to manually build the set of primitive concepts using a domain expert, and then ask the user to form higher-level concepts according to their specific query needs. While this may be sufficient for query purposes, it is inadequate for annotation and filtering purposes, however. In that case, we propose to use supervised learning by examples, that is, to annotate a training data set and to extract the features from the training set that would generalize well for the given concept. A simple way to extract the generalized features is to aggregate them by taking the average, minimum, maximum, etc., depending on the feature type. In the CAMEL system,[10] for example, we used clustering to identify clusters of sample images that have common features and we used the centroids as cluster representatives.

A more sophisticated approach would be to use reinforcement learning techniques in order to automatically extract the right weights and features. Gradient descent techniques, simulated annealing, and back-propagation neural networks are all examples of such learning methods.[29,22,30] The problem, however, is difficult, and becomes even harder when we add inter-object constraints. In that case, we need to identify not only the right object features and their weights, but also the types of constraints between the primitive objects and their parameters. Extracting the structure of the network is a very important problem in neural networks and Bayesian networks but unfortunately, there are no universal solutions. In Bayesian networks, the structure and the initial probabilities are typically given by a domain expert, and in neural networks, the structure is extracted implicitly through the weight manipulation and weight decay in the learning process. There are also some methods for pruning nodes with small weights, and splitting nodes with heavy load, so that the network structure is manipulated explicitly as well.[31–33]

A related approach for concept extraction is using relevance feedback methods, where the user gradually refines the concept by providing feedback to the system about the query results. The above neural network learning techniques can in fact naturally be incorporated to provide non-linear relevance feedback. A more thorough investigation of these learning methods and their application to concept extraction and refinement is a major direction for future work.

4.3. Concept Translation

Regardless of the extraction method, once the concepts are defined (manually or automatically), they need to be formally expressed and stored for reuse. We need to be able to translate the internal concept representation into a well-specified definition and vice versa. We therefore study syntax languages for describing the concept semantics, as well as storage and retrieval issues. We also investigate the use of such languages for query purposes. Our approach is to view the query as a description of a virtual data set (namely, the data that the user is looking for), and therefore to express the query with the same language that we use to describe concepts and constraints. In particular, we consider two translation mechanisms for converting concepts into SQL statements and into MPEG-7 XML descriptions.

An illustration of specific concept and it's SQL representation was given in Section 1, and a totally different example was in Section 3 (Figure 3). The mapping to SQL is achieved by converting concepts into views, concept attributes into abstract data types (ADTs), and constraints into user-defined functions (UDF) acting as predicates. The function used to calculate the overall concept similarity from the child concepts' similarities is specified in the **ORDER BY** clause, and the functions used for aggregating children attributes into attributes for the parent concept are specified in the **SELECT** clause. The weights can be specified either with a separate **WEIGHTS** clause or can be built into the UDF invocations without changing the SQL syntax. Figure 4 illustrates how the example from Figure 3 can be expressed in an SQL statement.

The second translation mechanism enables use of MPEG-7 XML descriptions to express concepts, constraints, and queries, and to store them in libraries of predefined concepts and constraints. MPEG-7 is an emerging standard for multimedia meta data description driven by searching and browsing requirements of multimedia applications.[2,1] It defines a variety of XML schemas that can be used to capture semantic information about the data source.[34] It can therefore serve as a description language for primitive as well as composite concept definition purposes. Of particular interest are the Collection DS and the Graph DS, and their correspondence to the concept representation. Due to space considerations, we will not describe the above Description Schemes and their mapping to/form concepts. The interested reader is referred to[34] for more information on the MPEG-7 Description Schemes. One application of the XML concept translation mechanism is the fact that the concept definitions can then be automatically stored, queried, or compressed as XML data. In addition, MPEG-7 authoring tools such as the Visual Annotation Tool,[13] can be used to write and manipulate concept definitions with a simple and easy GUI interface. Since, in our framework, defining a concept is equivalent to specifying a query, or annotating a document, the above tool can automatically be used as a query or annotation interface.

```
CREATE VIEW GOAL_SCORE AS
SELECT    KICK.ID,
          EVAL(KICK.SIM_SCORE, NOT_CATCH.SIM_SCORE, SCORE.SIM_SCORE),
          MERGE(KICK.KEYWORDS, NOT_CATCH.KEYWORDS, SCORE.KEYWORDS),
          AGGREGATE_1(KICK.ATTR_1, NOT_CATCH.ATTR_1, SCORE.ATTR_1),
          . . .,
          AGGREGATE_N(KICK.ATTR_N, NOT_CATCH.ATTR_N, SCORE.ATTR_N)
FROM      KICK, NOT_CATCH, SCORE
WEIGHTS   0.1, 0.3, 0.6
WHERE     BEFORE(KICK.ID, NOT_CATCH.ID) = 1 AND
          BEFORE(KICK.ID, SCORE.ID) = 1 AND
          BEFORE(NOT_CATCH.ID, SCORE.ID) = 1 AND
          KICK.ID = NOT_CATCH.ID AND
          KICK.ID = SCORE.ID AND
          NOT_CATCH.ID = SCORE.ID AND
          LENGTH(KICK.ID) < 10 sec
ORDER BY  EVAL(KICK.SIM_SCORE, NOT_CATCH.SIM_SCORE, SCORE.SIM_SCORE)
```

Figure 4. SQL statement describing the *GOAL_SCORE* concept from Figure 3.

4.4. Concept Matching

The above discussion leads finally to the problem of supporting such constrained queries efficiently. We already mentioned three methods for joining fuzzy result sets, namely Fagin's algorithm,[24,25] the MARS query tree approach,[27] and the SPROC algorithm.[28] The proposed framework is designed in such a way that arbitrary join algorithms can be plugged in it for concept matching purposes. Therefore, the first two methods above can be used when the join is only a Boolean conjunction or disjunction, while the third method can be used for joins with fuzzy constraints.

We are currently working on an efficient join algorithm for the case of crisp constraints only. Our approach uses a heuristic algorithm that guarantees correctness of the answer but does not guarantee polynomial bounds on the execution time. Alternatively, we can guarantee polynomial time but not the correctness of the answer. In practice, we expect the best algorithm to be truly heuristic in that it will not guarantee either correctness or time bounds but will provide the best trade-off between the two. Note that the matching problem is NP-hard (follows by reduction from sub-graph isomorphism), and therefore we cannot guarantee both correctness and polynomial time bounds.

In addition to general join methods, there may be domain-specific constraints that use specialized join algorithms. For example, the WALRUS system[9] can be considered as a particular type of constrained querying, where the constraint enforced that the matches of query regions and candidate image regions maximize the overall matched area. An additional constraint could be added that the matching is perfect in that each region matches no more than one region from the other image and vice versa. The WALRUS system provided an efficient algorithm for computing the best matches under those constraints, and thus can be plugged into the proposed framework for that specific constraint type.

4.5. Selectivity Estimation

Another issue of importance to the query execution efficiency is the problem of estimating selectivity statistics for such arbitrarily complex queries. Since the queries are defined at run time, it is not feasible to maintain any particular kind of statistics that will give the query selectivity information directly. However, one possible solution is to propagate selectivity information from the child concepts to the parent concepts so that at run-time such information can be computed dynamically for arbitrary queries. This approach requires maintenance of selectivity statistics only for the primitive concepts that are statically defined in the system. For example, suppose that we have selectivity histogram information for the built-in concepts. When defining composite concepts on top of the built-in ones, we can ignore the constraints and derive the distribution statistics for the parent concept using a convolution of the histograms from the children methods. An efficient algorithm for computing the convolution of two histograms with N bins each will run in time $O(N \log N)$. Similarly, computing the parent histogram from k children can be done in time $O(kN \log N)$. This approach will approximate the distribution at the parent node as a mixture of independent distributions from the child nodes, and it may be sufficient for selectivity estimation purposes.

5. CONCLUSION

In this paper we introduced a framework for querying of multimedia databases by specifying complex constraints among high-level semantic concepts. The proposed framework improves usability and enhances query power by enabling the user to specify a rich set of constraints on the underlying concepts. The proposed design was general and modularized so that arbitrary user-defined attributes, constraints, or matching algorithms can be plugged in. We motivated the problem by presenting a set of application scenarios that are not supported by current database systems but can be enabled by the proposed framework. We presented a rich concept representation, and we touched upon the problem of concept extraction and automatic constraint learning from training data sets. We also illustrated a method for expressing the concept definitions using SQL and XML (MPEG-7 schema definitions).

REFERENCES

1. "MPEG-7 applications document v.8.1." ISO/IEC JTC1/SC29/WG11/M4839, MPEG99, July 1999.
2. "MPEG-7 requirements document v.10." ISO/IEC JTC1/SC29/WG11/M4839N2996, MPEG99, Oct. 1999.
3. M. Flickner, H. Sawhney, W. Niblack, J. Ashley, B. Dom, Q. Huang, 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**(9), pp. 23–32, 1995.
4. A. Gupta and R. Jain, "Visual information retrieval," *Communications of the ACM* **40**(5), pp. 69–79, 1997.
5. A. Pentland, R. W. Picard, and S. Sclaroff, "Photobook: Content-based manipulation of image databases," in *SPIE Storage and Retrieval Image and Video Databases II*, (San Jose), 1995.
6. C. E. Jacobs, A. Finkelstein, and D. H. Salesin, "Fast multiresolution image querying," in *Proc. of SIGGRAPH 95*, Annual Conference Series, pp. 277–286, August 1995. Available at http://www.cs.washington.edu/research/projects/grail2/www/pub/abstracts.html.
7. J. Z. Wang, G. Wiederhold, O. Firschein, and S. X. Wei, "Content-based image indexing and searching using Daubechies' wavelets," *Intl. Journal of Digital Libraries (IJODL)* **1**(4), pp. 311–328, 1998. Available at http://www-db.stanford.edu/∼zwang/project/imsearch/IJODL97/.
8. J. R. Smith and S.-F. Chang, "Integrated spatial and feature image query," *Multimedia Systems* **7**(2), pp. 129–140, 1999.
9. A. Natsev, R. Rastogi, and K. Shim, "WALRUS: A similarity retrieval algorithm for image databases," in *Proc. 1999 ACM SIGMOD International Conference on Management of Data*, pp. 395–406, (Philadelphia, PA), May 1999.
10. A. Natsev, A. Chadha, B. Soetarman, and J. S. Vitter, "CAMEL: Concept Annotated iMagE Libraries," in *Storage and Retrieval for Image and Video Databases*, SPIE, (San Jose, CA), Jan. 2001.
11. A. B. Benitez, J. R. Smith, and S.-F. Chang, "MediaNet: A multimedia information network for knowledge representation," in *Conference on Internet Multimedia Management Systems*, vol. 4210, IST/SPIE'00, (Boston, MA), Nov. 6–7 2000.
12. A. B. Benitez and J. R. Smith, "New frontiers for intelligent content-based retrieval," in *Storage and Retrieval for Image and Video Databases*, SPIE, (San Jose, CA), Jan. 2001.
13. "MPEG-7 Visual Annotation Tool." Available at http://www.alphaworks.ibm.com/tech/mpeg-7.
14. C. Faloutsos *et al.*, "Efficient and effective querying by image content," *Journal of Intelligent Information Systems* **3**, pp. 231–262, 1994.
15. W. Niblack *et al.*, "The QBIC project: Query image by content using color, texture and shape," in *Storage and Retrieval for Image and Video Databases*, pp. 173–187, SPIE, (San Jose), 1993.
16. R. W. Picard and T. Kabir, "Finding similar patterns in large image databases," in *IEEE ICASSP*, vol. V, pp. 161–164, (Minneapolis), 1993.
17. J. R. Smith and S.-F. Chang, "Querying by color regions using the VisualSEEk content-based visual query system," in *Intelligent Multimedia Information Retrieval*, T. M. Maybury, ed., IJCAI, 1997.
18. C. Carson, M. Thomas, S. Belongie, J. M. Hellerstein, and J. Malik, "Blobworld: A system for region-based image indexing and retrieval," in *Third International Conference on Visual Information Systems*, June 1999.
19. E. Petrakis and C. Faloutsos, "Similarity searching in medical image databases," *IEEE Transactions on Knowledge and Data Engineering* **9**, pp. 435–447, May/June 1997.
20. M. R. Quillian, "Semantic memory," in *Semantic Information Processing*, M. Minsky, ed., MIT Press, Cambridge, MA, 1968.
21. M. W. Firebaugh, *Artificial Intelligence: A Knowledge-Based Approach*, Boyd & Fraser Publishing Company, Boston, MA, 1988.

22. D. Rumelhart, D. Hinton, and R. Williams, *Parallel Distributed Processing, Explorations in the Microstructure of Cognition, Vol. 1: Foundations*, MIT Press, 1986.
23. S.-K. Chang, Q.-Y. Shi, and C.-W. Yan, "Iconic indexing by 2-d strings," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **9**, pp. 413–428, May 1987.
24. R. Fagin, "Fuzzy queries in multimedia database systems," in *Proc. of the 1998 ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, 1998.
25. R. Fagin, "Combining fuzzy information from multiple systems," *Journal of Computer and System Sciences* **58**, pp. 83–99, 1999. An extended abstract of this paper appears in {*Proc. Fifteenth ACM Symp. on Principles of Database Systems (PODS '96)*}*, Montreal, 1996, pp. 216–226.*
26. *M. Ortega, K. Porkaew, and S. Mehrotra, "Information retrieval over multimedia documents," Tech. Rep. TR-MARS-99-11, University of California, Irvine, CA, 1999.*
27. *M. Ortega, Y. Rui, K. Chakrabarti, K. Porkaew, S. Mehrotra, and T. S. Huang, "Supporting ranked Boolean similarity queries in MARS,"* IEEE Trans. on Knowledge and Data Engineering **10***, Nov.–Dec. 1998.*
28. *C. S. Li, J. R. Smith, V. Castelli, and L. Bergman, "Sequential processing for content-based retrieval of composite objects," in* Storage and Retrieval of Image and Video Databases, VI*, SPIE, 1998.*
29. *R. Schalcoff,* Pattern Recognition*, John Wiley & Sons, Inc., 1992.*
30. *Y. Chauvin and D. E. Rumelhart,* Backpropagation: Theory, Architectures, and Applications*, Lawrence Erlbaum Associates, Inc., 1995.*
31. *H. H. Thodberg, "Improving generalization of neural networks through pruning,"* International Journal of Neural Systems **1***(4), pp. 317–326, 1991.*
32. *A. S. Weigend, D. E. Rumelhart, and B. A. Huberman, "Generalization by weight-elimination with application to forecasting,"* Advances in Neural Information Processing **3***, pp. 875–882, 1991.*
33. *A. S. Weigend, B. A. Huberman, and D. E. Rumelhart, "Predicting the future: A connectionist approach,"* International Journal of Neural Systems **1***(3), pp. 193–209, 1990.*
34. *A. B. Benitez, S. Paek, S.-F. Chang, A. Haung, A. Puri, C.-S. Li, J. R. Smith, L. D. Bergman, and C. Judice, "Object-based multimedia description schemes and applications for MPEG-7,"* Image Communications Journal *, Summer 2000.*