An Object-Oriented Model for the Semantic Interpretation of Multimedia Data

James Griffioen, Raj Yavatkar, Robert Adams
Department of Computer Science
University of Kentucky
Lexington, KY 40506
{griff,raj,adams}@dcs.uky.edu

INTRODUCTION

As multimedia information becomes more mainstream, the need to go beyond the simple viewing of that information increases. In order to be a useful tool, multimedia data needs to be annotated and stored so that it can be queried like other information. Specifically, a general data management model is required that provides content-based annotation and retrieval of multimedia data.

Unfortunately, because of its diverse nature and the vast amount of information embedded within a multimedia document, there are currently no uniform methods of identifying semantic information within multimedia documents. Some indexing methods have been proposed, for example, using textual strings, or graphical icons, however, these indexing schemes suffer from two major drawbacks. First, they require a human operator do most of the work. This requirement is both imprecise and time-consuming. Second, modern indexing methods don't allow multimedia data to be updated as new semantic information becomes available — once the data is stored in the database, rarely is it updated with new annotations. For example, once a "person" has been identified within an image, further processing is necessary in order to identify a specific person. Under current multimedia indexing schemes, no method is provided to apply processing on data already stored in the database. Basically, processing is done only on new data as it arrives into the database.

What is needed is a general data management model that can use currently available processing techniques to analyze and categorize multimedia data based on semantic content. This new model should provide means of automatically (and manually) extracting and identifying the semantic content of multimedia using state of the art processing routines. The data model should also allow currently annotated data to be updated as new semantic information or new processing techniques become available.

We have developed a data model called MOODS (Modeling Object-Oriented Data Semantics) that provides such a framework. MOODS combines database operations with a powerful processing engine to allow multimedia to be automatically, and semi-automatically processed for extracting and storing semantic content. Once the data has been annotated, MOODS allows content-based retrievals to be made, possibly invoking other processing routines to further analyze the data in the database. This "lazy-evaluation" of semantic content permits old information in the database to take advantage of new processing routines and new semantic information.

MOODS OVERVIEW

MOODS is an extended object-oriented data model. MOODS objects contain both data and functionality, like ordinary objects, and also contain semantic information. This semantic information represents the current state of knowledge about the semantic content of an object.

The semantic information within an object serves two major purposes. First, it is used to permit content-based searching of multimedia objects. Second, the information restricts the operations on an object to only those that "logically" apply. By having processing routines specify the type of semantic information they can be applied to, MOODS can match the semantic information in an object with the semantic data requirements of an operation. Only those operations that "match" the data within an object can be applied. For example, not allowing a face recognition function to be applied to a picture of a horse.

The identification of a semantic feature is typically done through the application (either automatic or manual) of a series of functions. The new semantic information is represented by updating the semantic data within the object. Therefore, over the lifetime of an object, the semantic data within the object changes. However, once the semantic information in an object is updated, the functionality of an object should be changed to allow new functions to be applied, and to remove functionality that is no longer applicable. For example, once a "dog" has been identified, we should be able to apply a function to determine the type of dog.

Once the desired semantic information has been extracted from the object, the object's data and semantic content are entered in the database for later retrieval.

The functionality of an object is defined in terms of *logical operations*, which MOODS implements via *function groups*. A function group is a collection of related functions (e.g., edge detection routines) that, when applied to an object, automatically selects a function(s) appropriate to the semantic content of the object. Function groups permit users to apply abstract operations without regard to the logical applicability of a particular function.

Semantic information is specified by a set of text strings, where each string represents some specific semantic meaning, e.g., "dog", "house", "car". Both function groups and objects specify semantic content with these labels. In order to provide a more flexible method of specifying semantic content, a hierarchy of labels can be created. The hierarchy can be thought of as an inheritance (isa) hierarchy, where each node inherits the semantic meaning of its parents. For example, "collie" may be a child node of "dog", meaning that a collie "is a" dog.

The semantic hierarchy allows generic functions to be developed. For example, a function specifying it requires an "image", rather than "dog_image" or "house_image", etc. Since each child node inherits from its parents, all image types can be used with the function.

In our current prototype, we use Prolog to specify the semantic hierarchy, since it allows MOODS to easily query the hierarchy to see if a function should be applicable to an object. However, we are currently investigating a graphical method of generating the hierarchy, in which case, we would be able to interface into any other logic system to manage the hierarchy information.

The Prolog rules can also be used to guide the application of processing to multimedia documents already in the database. This typically occurs during a database query. The user may ask for a semantic object which cannot be found. However, a rule may exist (or can be created) that describes what the user is looking for, in terms of the semantic labels already stored in the database. For example, `President :- Person, VIP, HeadOfState`. This means that if an object is labeled with the semantic labels "Person", "VIP", and "HeadOfState", then the object should also be labeled with "President".

One area we are currently researching is the application of processing functions to objects currently in the database. Query-time processing can be accomplished by giving the name of a processing routine to apply, along with the semantic labels. Expanding the example from above, `President :- Person, VIP, HeadOfState, in_united_states()`. Now, besides being labeled with "Person", "VIP", and "HeadOfState", the function `in_united_states()` is invoked on the object. If the function returns a correct result, then the object can be labeled with "President".

DEMONSTRATION

The demonstration illustrates the major features of the MOODS model based on a prototype developed on a Unix workstation. We have developed an interactive interface to MOODS that allows users to apply image processing functions to image objects in order to extract semantic content.

Each node in the interface represents one MOODS object. Clicking the mouse on an object presents the user with a choice of functionality to apply to the object. Once a particular function is chosen, and function parameters are entered, the function is applied to the object.

MOODS has immutable objects. Applying a function to an object doesn't change the object, rather it creates a new object containing the results of processing. If the user is unhappy with the results of the processing, the immutable object paradigm allows the user to reapply the function (perhaps with different parameters) in order to achieve the desired results.

The MOODS demonstration presents a simple prototype system. The purpose of the prototype is to extract letters and ligatures from a manuscript page of Beowulf based on predefined samples of certain letters. The application uses a correlation algorithm to try to find the occurrences of a letter or a ligature within the page. This type of processing is useful in the paleographic analysis of documents.

Once the given letters are located, MOODS updates its database with the location of the letter and enlarges the located letter for fine-grained human analysis. When processing is complete, the user can view the results of the search through an external viewer that allows the user to select the located letters in order to view the enlarged regions.

SUMMARY

MOODS provides users with the capability of extracting the semantic content from a variety of media types. The demo we have presented shows only one such type — images. But clearly MOODS can be extended with the appropriate functionality to extract information from audio, graphs, maps, etc.

For more complex media, for example video, we are currently investigating ways to support partitioning video into segments, and of choosing "representative" frames from each segment. We plan to provide generic support for this kind of "pre-processing" so that other complex media types can be handled similarly.