MOODS: A Multimedia Information Modeling System*

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

Rajiv Mehrotra
Dept. of Mathematics and Computer Science
University of Missouri-St. Louis
St. Louis, MO 63121.
rajiv@arch.umsl.edu

Abstract

Unlike alphanumeric data, multimedia data contains embedded information *which must be extracted and interpreted before it can be used or queried. Our goal is to explore new data models that are capable of representing and processing multimedia data. We propose a new object-oriented data modeling approach called MOODS (*Modeling Object-Oriented Data Semantics) *that integrates the multimedia data processing component of an information management system with the data modeling technique. MOODS supports three new abstractions that distinguish it from ordinary object-oriented data modeling techniques:* dynamic data semantics*,* function groups*, and* composition*. This paper describes the MOODS approach and shows how the approach can be applied to a pictorial information system.*

1 Introduction

Historically, a primary role of computers has been to manage data, in particular, alpha-numeric data consisting of numbers and characters. However, recent hardware and software advances have created a wide variety of new data types generally referred to as multimedia data that have almost no similarities to alpha-numeric data. Video cameras, audio speakers and microphones, high-resolution graphics displays, and other computer peripherals have become inexpensive and widely available. The multimedia data created or displayed by these devices differs drastically from conventional alpha-numeric data and requires completely new methods for managing this data.

In particular, we find the following aspects of multimedia data challenging and important from a database management perspective:

Embedded Information: Multimedia data, unlike alpha-numeric data, contains almost limitless amounts of *embedded information*. That is, the structure of multimedia data and the data's contents in its raw form is relatively meaningless. Instead, the information of importance to the user or application is embedded in the data and the data must be processed to extract the important information. For example, a picture may need to be processed to identify a tree, house, or some other semantic object that is in the picture. Consequently a multimedia information management system must provide mechanisms that allow users and applications to access the embedded information. Moreover, the amount of embedded information is almost limitless which means the information management system must be capable of efficiently handling potential large amounts of information.

Domain-dependent Views: A unique characteristic of multimedia data, not found in alpha-numeric data, is that the important (embedded) information that must be extracted depends on the application domain or the user/application. Consider a picture of a river flowing through a valley. A civil engineer might be interested in the construction of the bridge over the river, while an environmental engineer might be interested in the color of the water and the amount of trash lining the edges. Another example might be a satellite image or an aerial photograph. A defense application may identify various regions (roads, bridges, specific structures) in the photograph as target sites and associate target specific identification information with each region. However, a geologist might concentrate on identi-

*This work was supported in part by the NASA Langley Research Center under grant NAG-1-1276 and by a NSF Research Initiation award no. NCR-9111323.

fying and classifying land and water masses according to her specific interest. In both examples, the application domain, be it civil engineering, environmental engineering, defense, or geology, determines which embedded information is important and which is not.

In other words, each application defines its own *view* or interpretation of what is important and meaningful in an image. Moreover, defining a view typically involves applying a view-dependent multistage transformation to extract the desired embedded information.

Evolution of Views: A related problem to the domain/user view problem is the fact that views often evolve or change over time. That is, the embedded information that was important in the past is either no longer important, or the information is now viewed in a new way. For example, recall our picture of a river scene. Assume we had extracted and identified a boat from the picture, but were unable to determine the type of boat in the picture. Later, new image processing/enhancements techniques become available that allow us to determine the boat's type. In this case, the advent of new technologies and mechanisms resulted in a revised view of the embedded information.

Heterogeneous Components: Another important characteristic of multimedia data modeling arises from the fact that multimedia data often consists of multiple components. For example, a picture or video is often associated with registration information identifying the location the picture was taken, the date the picture was taken, the photographer, etc., which are an integral part of the data. Moreover, like registration information, the embedded information extracted from the data also must be maintained as an integral part of the data. In some sense multi-component multimedia data resembles hypermedia. However, in this case, the "links" can now point into the multimedia data from itself.

We are investigating a new model for managing multimedia data called *MOODS*. MOODS takes a new approach to the design of information management systems by incorporating the ability to directly model and manipulate multimedia data objects to get at (extract) the embedded information contained in the data[1]. The MOODS data modeling paradigm builds on and extends standard object-oriented approaches to provide the flexibility and semantic processing capabilities required by a multimedia information management system.

This paper describes the MOODS multimedia data modeling system, shows how such a design can be applied to a pictorial information system, and describes a prototype system with an example application.

2 The MOODS Multimedia Data Model

MOODS supports three new abstractions that distinguish it from ordinary object oriented data modeling: *dynamic data semantics*, *composition* and *function groups*.

2.1 Dynamic Data Semantics

Data semantics represent a significant conceptual deviation from the standard object oriented model. In MOODS, each object consists of three components: data structures, methods (functions), and a *data semantic* specification. The data semantics portion of the object describes the current semantic meaning of the data currently stored in the data structure component of the object. Over the lifetime of an object, the semantic meaning of the object will vary depending on the current contents of the data structures. The data semantic component allows us to record the current semantics of the data and to adjust (change the binding of) the set of methods associated with the data. This ability allows us to correctly model the data based on the semantic meaning of the data rather than the structure of the data as is the case in an object oriented model.

Multimedia data such as image data often undergoes a sequence of processing stages before reaching its desired form. Each stage has its own semantic meaning and applicable functions. To model these semantic changes, MOODS allows the methods associated with an object to change both the data and the semantic meaning of the data. Note that changing the semantic meaning of the data effectively changes the class to which the object instance belongs. In MOODS, all methods or functions associated with a particular class have the property that when invoked, they transform an instance of one class into an instance of another class (possibly the same class). These transformations between semantic classes closely model the sequence of processing stages a data item traverses over its lifetime.

MOODS also deviates from the standard object oriented model in terms of the way in which methods operate on the data within an object. MOODS supports *immutable objects*. When a user invokes a method on an

[1]In the MOODS system, multimedia data refers to any data that contains embedded information. Thus, multimedia data types handles by MOODS may include images or pictures, graphs, charts, tables, video, audio, and some new and special-purpose forms such as molecular graphical/structural models or CAD/CAM models.

object, the method does not modify the current object but instead produces a completely new object. Saving past versions of nontextual data is useful for several reasons. An image typically goes through a series of transformations that extract information from the image or compute new information based on the image. Thus, the state of an object must contain not only the current data and associated semantics, but also the sequence of operations that were applied to the original object to bring it to the current state. Such a history is especially useful in an interactive programming system where a user may wish to test a variety of alternatives for processing the visual information.

A desirable property of nontextual data modeling is the ability to backtrack to an earlier version of the data to restart processing at that point. Backtracking is particularly useful in an interactive data modeling system. Unfortunately, in practice, few methods have an inverse operation requiring us to store previous versions of the object. Also, past versions of data may contain information in an unprocessed state that may be of value to a researcher (e.g., a radiologist may need the original x-ray to confirm conclusions reached from a processed version of the x-ray).

Thus, MOODS allows a user to take an image through a sequence of transformations and associate specific semantics with each stage of transformation. Given an application domain, users can specify a tree consisting of semantic stages and transitions that the image may legally traverse for that domain. At run time, users manipulate an image according to the transitions specified in the tree, possibly backtracking or repeating a transition as often as necessary. In the following, we describe how MOODS allows dynamically creating new objects from existing objects.

2.2 Composition

An information modeling system must have the ability to create new objects by combining two or more existing objects together. For example, two independent pictures of the same scene may be merged together to produce additional information about the scene (e.g., the depth of objects in the scene). Alternatively, a single facial image might be processed to extract the individual facial features, modify some of the features, and then merge the modified facial features back together to create an object that can be easily searched for certain combinations of facial feature characteristics.

Moods provides a mechanism by which a user can dynamically create a new object from existing objects. The new object's type and semantic meaning may correspond to a predefined class or may be a completely new, dynamically created class.

Logically, the Composition operation can be broken into three steps: merge the data from the objects being composed, merge the functions from the objects being composed, and add new abstract functions that operate on the merged data. Composition is similar in many respects to the notion of inheritance in standard object-oriented programming. However, unlike inheritance, composition is the runtime combination of the current data found in the objects being combined.

2.3 Function Groups

Another unique feature of MOODS is that it allows the operations on an object to be defined in terms of abstract methods called *function groups*. Each function group represents an abstract or logical function and actually consists of a set of related functions. Function groups are globally defined and can be incorporated into any class. MOODS objects are defined in terms of function groups rather than specific functions. That is, a class definition specifies the list of abstract operations that apply to the class, not particular functions.

Typically a function group contains a set of semantically or functionally equivalent methods. The set of methods within a function group need not accept the same inputs nor produce the same outputs, however, they should all provide the same or similar logical operation. For example, consider an edge detector function group that represents the abstract operation *edge detection*. The group actually consists of a suite of edge detection methods based on different edge detection algorithms. Even though each one of the edge detection routines perform edge detection, they may differ in terms of their accuracy in detecting edges, execution times, or space requirements. As new edge detector methods become available, they are dynamically added to the edge detector function group and made available to existing and new objects.

Use of function groups raises an important question about resolving ambiguities when more than one function can be selected at run time. To uniquely identify a particular function from a function group, the system needs four pieces of information: function name, types of any implicit inputs, types of implicit outputs, and the data types of the explicit input and output parameters. In many cases, the current semantics of the data provides sufficient information for the system to automatically select a specific function at runtime [3]. However, if an ambiguity arises, the user must select a specific function from the function group. If the system is interactive, this selection may occur at runtime via prompting. Alternatively, a specific default function can be specified at compile time, load time, or runtime.

3 The MOODS Prototype

To evaluate our design and new semantic data model, we have developed a new language called *MOODS*. The MOODS language is an extension of C++ with support for semantic data groups, function groups, composition, casting, and a visual runtime interface for processing semantic data. The MOODS compiler is implemented as a preprocessor whose output is then passed through the standard GNU C++ compiler.

Creating a new application typically requires only creating a set of specifications (SPEC_FILE) that define the data and semantic processing stages the data may traverse over its lifetime. The MOODS preprocessor takes as input a SPEC_FILE and a library of function groups that contains definitions and implementations of available function groups. The resulting C++ code is compiled and then linked into a MOODS executable.

MOODS includes a Tcl/Tk (X-window) user interface that helps users push semantic data through the various processing stages at runtime. The user interface displays the stages the data has already traversed and allows the user to interactively direct further processing. The user selects the next processing step using a pop-up menu with the resulting processing stage being displayed on the screen. Backtracking can be achieved easily by clicking on a previous stage to invoke its pop-up menu of applicable functions. Our experience shows that users are rarely satisfied with their first choice of parameters at any give processing stage. Consequently, backtracking has proven to be an crucial capability. Backtracking allows a user to interactively and iteratively refine an image until the desired information is visible/available.

3.1 An Example Application

To demonstrate the MOODS data model, we have implemented the image processing component of an image database system. However, due to space constraints, we will limit our example to a rather basic application of the model. A more detailed description of the prototype along with a more realistic and complex example can be found in [4]. Despite its simplicity, the basic application presented below is sufficient to demonstrate the salient features of the MOODS model and embedded information management/processing.

Consider an image database that contains black and white images of block figures. Figure 1 illustrates an example image taken from the database. To effectively search/query the embedded information contained in these images, users must first be able to identify the semantic components of a block image (e.g., *arms*, *legs*, or *head*). Once the semantic information has been extracted, a user can search and retrieve images using semantic searches.

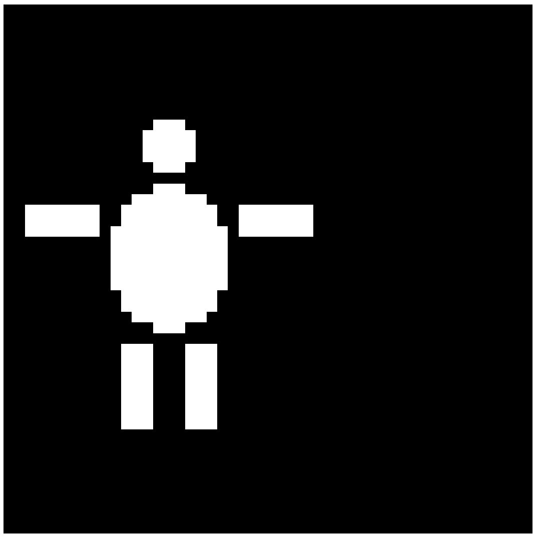

Figure 1: An example block figure image.

Assume three function groups are available for processing image data:

Erosion: a set of functions which erode an image using either disk erosion, horizontal line erosion, or vertical line erosion.

Dilation: a set of functions which dilate an image using either disk dilation, horizontal line dilation, or vertical line dilation.

Difference: a function to find the differences between two images.

Using this set of functions, users can manipulate images to extract the desired information, ultimately reaching a stage where the necessary embedded information has been clearly identified.

However, as the image passes from stage to stage, the semantic information conveyed by the image continues to change. At each step of the transformation process, the image has some semantic meaning and only certain functions should be applicable. Given block images and the above set of functions, we can define the following semantic stages and transitions that an image may legally traverse:

New Block Image: a new image in raw format.

Eroded Head/Body: extracted head and body after disk erosion

Head/Body: head and body after disk dilation

Eroded Body: extracted body after vertical line erosion

Body: body after vertical line dilation

Head: head after differencing

Arms/Legs: arms and legs after differencing

Eroded Legs: extracted legs after vertical line erosion

Legs: legs after vertical line dilation

Arms: arms after differencing

Figure 2 illustrates these processing steps. The semantics of each processing stage dictate the set of operations available and the semantic data object that results from applying an operation.

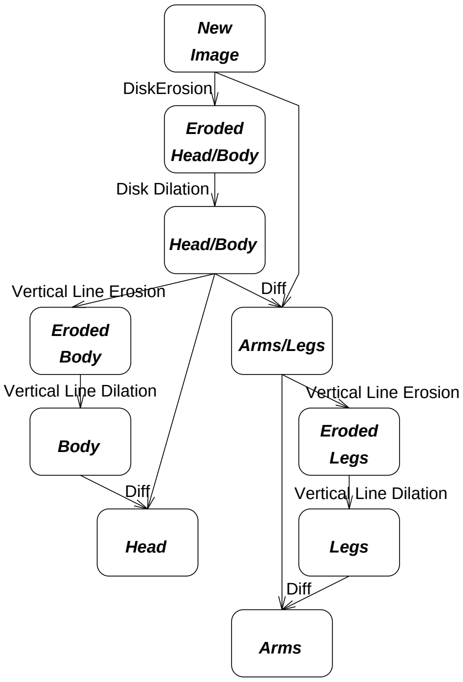

Figure 2: The processing stages a block figure image may traverse.

Once the above set of processing stages has been defined, users can manipulate the images either manually or automatically to reach a stage where the desired embedded semantic information has been extracted and identified. For example, given the image shown in Figure 1, a user may process the image to identify the arms and the legs (see Figures 3 and 4), which can then be queried and searched for using semantic queries.

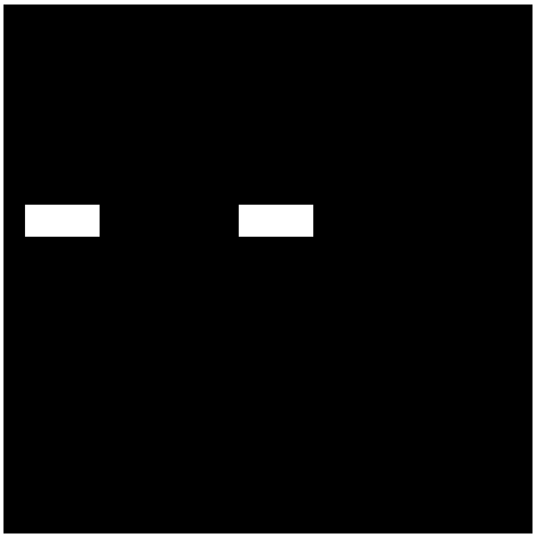

Figure 3: The portion of the image identified as arms by the semantic state "arms" in Figure 2.

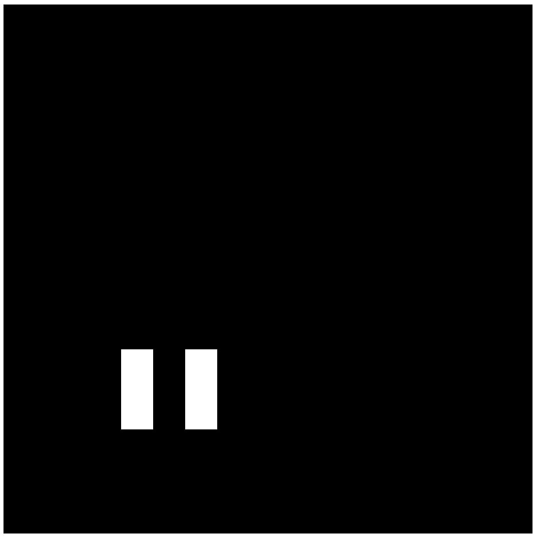

Figure 4: The portion of the image identified as legs by the semantic state "legs" in Figure 2.

Despite the simplicity of this example, it demonstrates several features of the MOODS data model. First, the semantics of the data at each stage continuously tracks the current meaning of the information that has been identified. The newly identified semantic information can then be queried and searched. The semantics at each stage also restrict the allowable operations at that stage, preventing users from applying inappropriate operations. Second, extracting the arms and legs requires computing the difference between the results of two distinct processing stages which illustrates the power and necessity of the combine operation. Third, both backtracking and interactive refinement allow users to follow multiple processing paths out of a single stage and also allows the user to repeat operations which fail to produce the desired results (e.g., the wrong parameters were selected for an operation such as dilation or erosion). Fourth, MOODS history and path saving abilities allow users to define and then automatically replay a processing path (e.g., to extract the arms and legs) on new images.

4 Summary

Management, modeling, and searching non-alphanumeric data (such as images, videos, graphs, charts, audios, etc.) has emerged as an important research issue and will continue to play a major role in large information databases as such media forms become increasingly commonplace.

In this paper, we have proposed a new semantic object-oriented data model that integrates the processing of non-alphanumeric data into the standard database abstraction. In particular, we have shown how such a model could be applied to visual data such as image data. The model not only allows users to query a database of visual information but allows a user to direct an image through a series of transformations which change the semantic meaning of the image and extract application specific information from the image. At each step in the transformation, the model defines a new set of applicable functions based on the current semantic meaning of the image. We have also shown how abstract operations can be applied dynamically to existing and future data. This ability allows the model to incorporate new techniques and algorithms as they become available and, thus, effectively revitalizes existing data by providing new views of the information inherent in the data. We have built a prototype system which demonstrates the effectiveness of the model.

References

[1] S. K. Chang, J. Reuss, and B. H. McCormik, An Integrated Relational Database System for Pictures, *Proc. IEEE Workshop on Picture Data Description and Management*, Chicago, IL, April 1977, pp 49-60.

[2] N. S. Chang and K. S. Fu, A Relational Database System for Images, in *Pictorial Information Systems*, S. K. Chang and K. S. Fu, Editors, North-Holland Publishing Company, Amsterdam, 1980, pp. 405-414.

[3] J. Griffioen, R. Mehrotra, and R. Yavatkar, An Object-Oriented Model for Image Information Representation, *Technical Report No. 232-93*, Department of Computer Science, University of Kentucky, April 1993

[4] James Griffioen, Rajiv Mehrotra, and Rajendra Yavatkar. A Semantic Data Model for Embedded Image Information. In *IS&T/SPIE Symposium on High-Speed Networking and Multimedia Computing 1995*, February 1995. To appear.

[5] W. I. Grosky and R. Mehrotra, Image Database Management, *Advance in Computers*, Vol. 34, M. C. Yovits, Editor, Academic Press, 1992, pp. 237-291.

[6] A. Gupta, T. E. Weymouth, and R. Jain, An Extended Object-Oriented Data Model for Large Image Data Bases, *Proc. 2nd Symposium on Design and Implementation of Large Spatial Databases*, Zurich, Switzerland, August 1991.

[7] A. Gupta, T. E. Weymouth, and R. Jain, Semantic Queries with Pictures: The VIMSYS Model, *Proc. 7th International Conference on Very Large Databases*, Barcelona, Spain, August 1991, pp. 69-79.

[8] H. V. Jagadish and L. O'Gorman, An Object Model for Image Recognition, *IEEE Computer*, vol. 22, No. 12, 1989, pp. 33-41.

[9] T. Kunii, S. Weyl, and J. M. Tenenbaum, A Relational Database Schema for Describing Complex Pictures with Color and Texture, *Proc. 2nd International Joint Conference on Pattern Recognition*, Copenhagen, Denmark, August 1984, pp. 310-316.

[10] W. Kim, Object-Oriented Databases: Definition and Research Directions, *IEEE Transactions on Knowledge and Data Engineering*, vol. 2, No. 3, September 1990, pp. 327-341.

[11] R. Mehrotra and W. I. Grosky, REMINDS: A Relational Model-Based Integrated Image and Text Database Management System, *Proc. Workshop on Computer Architecture for Pattern Analysis and Image Database Management*, Miami Beach, FL, November 1985, pp. 348-354.

[12] B. Stroustrup, What is "Object-Oriented Programming"?, *IEEE Software*, May 1988.