Automatic and Dynamic Identification of Metadata in Multimedia

James Griffioen, Raj Yavatkar, Robert Adams
University of Kentucky
{griff,raj,adams}@dcs.uky.edu

Abstract

Multimedia data (video, audio, images) are now in widespread use and require radically new techniques for indexing and searching. Multimedia data consists of two components: the raw data itself and semantic information (metadata) contained in the data. Unfortunately, the metadata needed for indexing is not directly accessible, and must first be identified. Because of the voluminous amount of multimedia information, automated techniques must be used for semantic identification. Fortunately, an expanding array of multimedia processing routines exist that can be used to automatically identify semantic content. However, because multimedia data contains an almost infinite amount of domain-dependent semantic information, we cannot identify all embedded information ahead of time. We introduce a new paradigm called MOODS, that combines an enhanced object-oriented data model, a powerful multimedia database, and a dynamic semantic information extraction engine. MOODS provides a framework in which processing operations can be automatically, dynamically, and intelligently applied to the subsets of the data that are most likely to contain the metadata sought by a user. We have used the MOODS framework to construct tools for a variety of application domains including a paleographic tool for managing, manipulating, and searching for the semantic letterforms of ancient manuscripts.

1 Introduction

Multimedia data (video, audio, images) are now in widespread use in many areas of scientific research, such as medicine, geology, climatology, and cartography. However, to be a useful medium, one must be able to index and search multimedia data much like typical alphanumeric information. Unlike alphanumeric data typically represented by ASCII values, the content (pixel values or compressed pixel values) that make up multimedia data don't convey much interesting information in themselves. Consequently, we would like to index multimedia information on what the pixels represent, i.e. the *semantic content* of the data rather than indexing the pixel values themselves. For example, searching alpha-numeric data for the string 'abc' is straightforward. However, searching for a car in an image or the sound of a lion in an audio clip requires the ability to access the semantic content. We call this semantic content "metadata". Each image, audio, or video clip contains enormous amounts of potentially useful metadata.

Most multimedia data contains two types of metadata that can be indexed. First, multimedia is typically entered along with registration information. Example registration information includes when the image was taken, where the video was shot, who recorded the audio clip, camera settings, etc. Second, multimedia contains semantic information that must be assigned manually (human examination of the picture) or automatically (visual processing). In short, metadata in MOODS is any information in addition to the raw data stream. The remainder of this paper focuses on the latter metadata component (extracted or assigned semantic information), although the former can be treated in a similar manner.

Manual identification of semantic features is unacceptable for several reasons. First, the amount of multimedia data is large and growing quickly. Human operators cannot label data at even a fraction of the rate at which it is being generated. Second, there is an enormous amount of metadata contained within each multimedia object, and no operator can manually extract all the important semantic features that are of potential interest to all application domains. Third, identifying the semantic features in the data is typically not sufficient. Users want to reason about the content and each user may reason differently. For example, identifying a bone in an x-ray may not be sufficient, a doctor will want to classify the bone as fractured or broken.

Because manual identification of semantic content is both time consuming and inexact, we need automated methods for semantic data extraction. Fortunately, an ever expanding array of multimedia processing routines exist that can be used to automatically identify semantic content [1, 2, 3, 4, 5, 6].

However, because multimedia data contains enormous amounts of semantic information, the system cannot possibly identify all meaningful semantic information ahead of time. Therefore, the system must provide dynamic (runtime) and automatic processing. Depending on the domain, some semantic content will be more important than others, therefore the system must postpone the identification of less important semantic content in favor of semantic information that is more crucial. All other processing should be done on an "as-needed" basis, postponing operations until query time.

Multimedia processing operations only provide information on semantic features found in multimedia data. They do not allow reasoning about the semantic content. For example, image processing techniques can locate people in an image. However, determining if a person is a politician requires information outside the domain of image processing. Thus, we need another mechanism that allows the system to deduce semantic information based on the metadata returned from processing and other information sources.

2 The MOODS Management System

This paper introduces a new data modeling paradigm, called *MOODS*, for managing multimedia data and identifying semantic content. MOODS is framework for managing the semantic content (metadata) of multimedia data. It integrates an enhanced object-oriented data model, a multimedia database, and a dynamic semantic information extraction engine. The purpose of MOODS is to provide a framework for developing applications that allow users to identify, classify, store and retrieve the semantic metadata embedded within multimedia data.

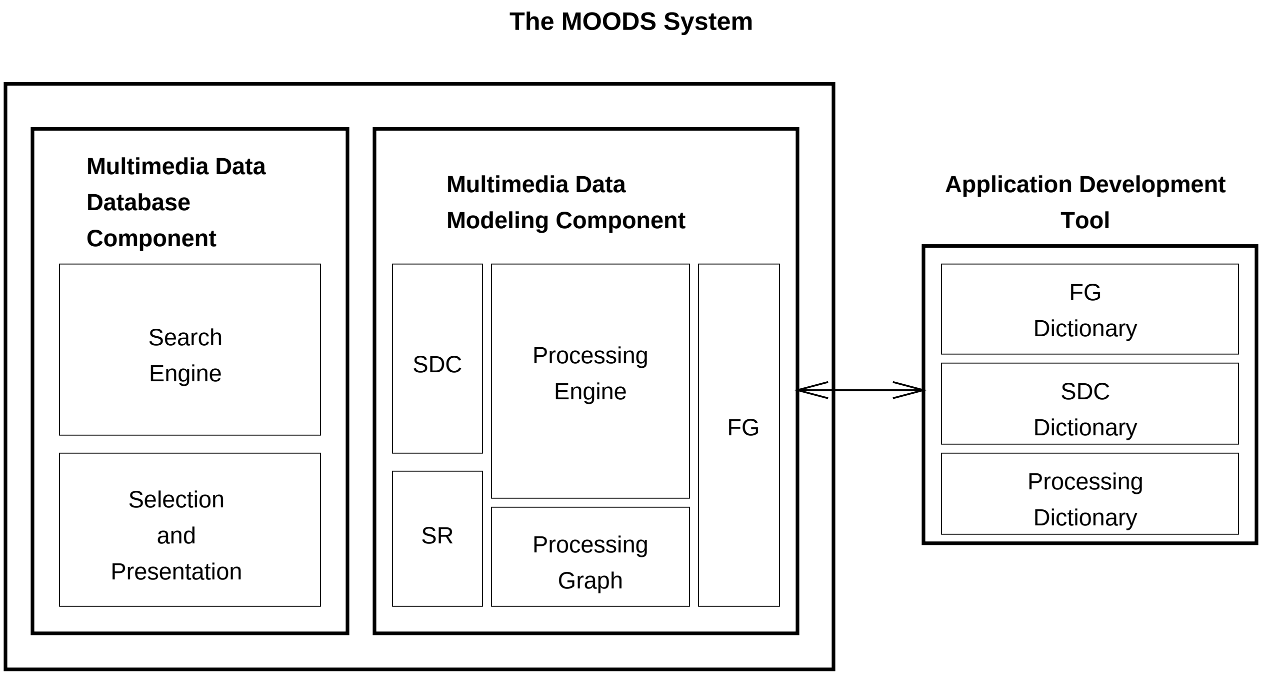

Figure 1: The components of the MOODS multimedia information management system.

MOODS takes a unique approach to the management of semantic metadata. In particular, MOODS provides a processing engine that supports *semantic objects*. In addition, the semantic processing engine is tightly coupled with a database (i.e., the database understands semantic objects). Figure 1 illustrates the system architecture. The *database* component stores information about semantic objects and supports queries, while the *data modeling* component contains a processing engine that can extract semantic metadata. The processing engine also contains semantic inference rules (SR) that express high-level semantic concepts. In addition, an *application development tool* aids in the construction of modeling systems and facilitates reuse of functions.

When new objects are added to the system, the processing engine automatically preprocesses the data to identify a limited amount of semantic content. The objects, along with their semantic metadata are then stored in the database. When a user issues a query for semantic content, MOODS returns the database objects matching the user's request. However, if no matching objects are found, the database calls on the processing engine to dynamically search for data that may contain the desired semantic information. Using the processing rules and semantic inference rules, the processing engine will locate data potentially containing the desired information and process it further to extract the information if it is present.

Combining processing routines with semantic inference rules into an object-oriented data modeling framework provides a powerful facility for identifying embedded semantic metadata. The framework facilitates automatic processing required to identify semantic content. Furthermore, delayed processing allows MOODS to dynamically identifying a wide range of semantic content without having to initially process new objects for all possible content.

3 Semantic Objects and Processing Graphs

MOODS proposes an enhancement to the standard object-oriented programming model by introducing the concept of *semantic objects*. Like standard object-oriented models, MOODS objects contain both data and methods to manipulate the data. However, MOODS objects differ from standard objects by including a semantic definition. The semantic component consists of a set of semantic descriptions identifying semantic concepts and entities contained in, or represented by, the object. The semantic description captures the current knowledge (known facts) about the object, i.e., the semantic features that have been identified in the object.

Users invoke methods on semantic objects to identify semantic features in the object. Assuming the feature is found, the semantic description is updated to reflect the new content. As the semantic content changes, the list of applicable methods may also need to be changed because some methods may no longer make sense and new methods may be needed. Consequently, MOODS users view semantic objects differently than standard objects. As the user manipulates an object, the object's "appearance" (semantic meaning and available methods) changes.

Identifying a semantic feature typically requires a series of processing operations. For example, before a car can be identified in a picture, we must first identify all the edges, construct connected regions, identify shapes, and then classify the shapes. Note that the ability to classify shapes was not needed (made no sense) in the early stages, but was later very important as the semantic content began to evolve.

Semantic objects model this type of transformation process very nicely. As the object is processed, new information is obtained. At each stage of the transformation, old methods that are no longer appropriate are removed, while new methods needed to extract additional information are added. At each stage the semantic description changes to express the information known about the object.

Given the ability to represent the transformation process, the MOODS system allows expert users in a particular application domain to define the legal semantic transformations a semantic object may undergo during its lifetime. At each stage of a transformation the expert defines the meaningful methods that can be applied given the current semantic meaning of the object. Once a semantic processing model has been defined, novice users can interactively manipulate their objects to extract the semantic features they find interesting.

Many application domains require the same set of processing operations in the initial stages of a transformation. Consider images. Most application domains typically need noise removal, edge detection, segmentation, shape identification, connected region analysis, and other image processing routines in the initial stages of analysis. Because many semantic objects require such functionality in the early stages of the transformation, MOODS supports the concept of *function groups*. A function group (FG) represents a logical operation and consists of a set of functions capable of performing the operation using different algorithms or operating on different data. Consequently, semantic objects can be defined in terms of logical operations, i.e., the logical operations that make sense given the current semantic meaning. At runtime, a specific function is automatically chosen depending on the data to which the function group is applied. This allows users to write functions independent of the objects in which they will eventually be used, and facilitates reuse of functions in many different objects.

4 Automatic Processing

Because manual identification of all potentially interesting semantic content is not practical, automatic techniques to extract and identify semantic metadata must be provided. As noted earlier, automatic techniques typically involve applying a sequence of operations to extract the desired information. The particular processing sequences required depends on the semantic metadata desired, the characteristics of the input data, and the precision desired by the user. For example, identification of a car in an image may require a different processing sequence than the sequence used to identify a flower. At each stage of the sequence, functions must be applied with certain parameters. For instance, a contrast enhancement function takes a contrast map as a parameter. The best map to use depends on the data being enhanced. (i.e., the characteristics of the input data). Consequently, the parameters used in the functions that comprise a sequence may need to change for different input data. Finally, each user has their own expectation levels for the metadata processed/retrieved by the system. By modifying the parameters of the functions comprising the transformation sequence, a user can tailor the resulting semantic metadata to their liking.

To support the type of flexibility in the processing described above, MOODS divides up the task of defining a processing sequence into three parts: definition of a *processing graph*, definition of *processing paths* and *interactive refinement*. A processing graph defines the legal operations and possible function sequence that can be used to extract various semantic features. An expert in a particular application domain defines the graph by specifying the semantic states an object would have to go through to reach semantic stages of interest in the expert's domain.

Figure 2 illustrates a simple processing tree for a medical application that identifies components of the heart. Each node represents a semantic state and each edge represents a logical operation. Starting with a plain image read from disk, the user is able to perform image processing operations to remove image noise, to modify the image contrast, etc. Each of these operations will transform the object to an EnhancedImage object. Once the image has been enhanced to the user's satisfaction, edge boundaries can be detected, then the connected regions can be found. From the region information the aorta or the ventricals can be identified, and so on.

Given a processing graph that defines the processing operations required to extract all semantic information of interest to a particular application domain, semantic identification can be achieved by applying the appropriate processing operations necessary to extract the desired semantic features. Applying functions to extract the desired metadata results in the definition of a *processing path*. Processing paths may be defined by experts or novices. While applying functions, a user must specify the function parameters that work best for the data being processed. Users may need to define two or more processing paths that traverse the same semantic states but use different parameters along the path. One path may work well

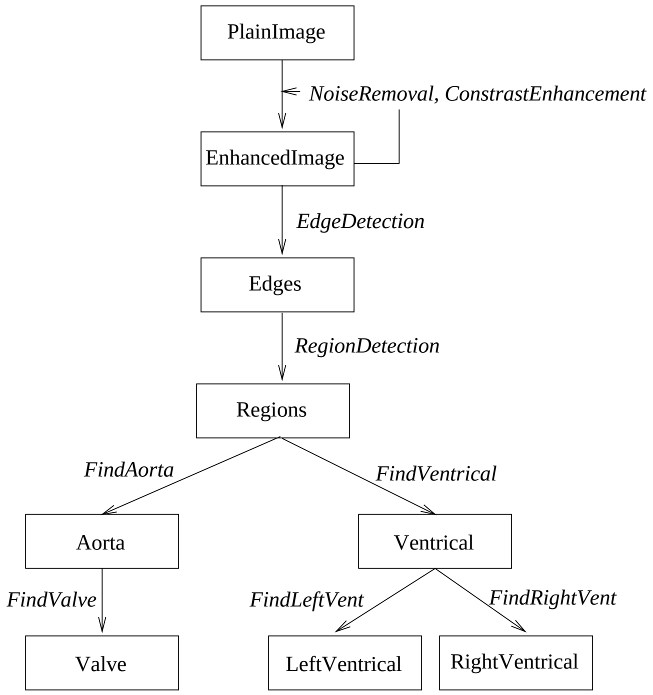

Figure 2: An example processing graph.

for fuzzy images while the other path (through the same states) works well for bright images.

All the processing operations from the initial state in the processing graph to the final state (the object state in which the semantic feature has been identified) can be saved and automatically replayed to extract semantic information from similar data. Consequently, new data can be processed quickly and automatically using previously defined processing paths.

Although an automatic processing path has been defined, the results may not suit all users. Consequently, users need to be able to modify the parameters along a path to their liking. However, in many cases, only the most recently applied functions need to be reapplied (i.e., the specific functions that are focusing in on the semantic feature of interest).

Backing up to reapply functions requires that the old object data be accessible. MOODS supports immutable objects for this purpose. Each time an object method is invoked, instead of actually changing the object's data, MOODS sends the data created by a method to a new object. In essence, methods act as data filters, taking data from one object, transforming it, and sending the results to another object. This new object is also updated with the newly identified semantic content, along with a new set of methods. Once the user finishes refining the function parameters, old objects can be deleted.

5 Dynamic Processing

Each multimedia data item contains an enormous amount of metadata. Consider an image or video taken of the crowd and the playing field at the SuperBowl. Such an image contains a staggering amount of information. However, not all of the potential semantic information will be useful to every user or every application domain. A sports enthusiast will be interested in the names of the teams that are playing and the score, while a corporate sponsor will be interested in the location and visibility of the sign containing the company's logo. Consequently, it is unlikely that all information of interest to all potential users can be predicted and identified. Furthermore, given the high costs of identifying semantic information both in terms of computation time and storage space, only a limited amount of embedded metadata can be identified.

The problem then is one of selectively identifying only the most crucial semantic information. Identification of all other semantic information should be postponed until the data is actually needed, i.e., query time. Postponing all non-critical processing until query time allows us to introduce new objects into the system relatively painlessly. New objects are automatically preprocessed to identify the semantic content that will be useful to a majority of the users. Later, when queries are made for metadata that has not yet been identified, MOODS automatically applies the processing operations from a processing path that can identify the desired semantic content. This type of dynamic processing allows us to postpone a large amount of time consuming and potentially useless processing.

Recall that automatic identification of semantic information requires a series of operations (i.e., a processing path). MOODS divides each automatic processing path into two components, the *pre-processing component* and the *search-processing component*. The division between the pre-processing and search-processing components is defined when the processing path is created, and typically represents a trade-off between semantic content that is generally useful to most users (pre-processing), and specialized content that is rarely needed, or is very time consuming to identify (search-processing).

To limit the initial processing costs, new objects are automatically processed only along the pre-processing component. The data along with any semantic content identified during the pre-processing stages is then stored in the database.

The semantic states along the search-processing component represent semantic information that can potentially be identified from data currently in the pre-processing states. The search-processing path is invoked when a user searches for semantic metadata corresponding to a node in the search-processing graph. MOODS searches for a semantic state in the search-processing component that matches the user's query. A matching node in the graph means that MOODS is capable of identifying the requested semantic content.

Because all data in the pre-processing state cannot reach all states on the search-processing component of the path, MOODS automatically applies the processing operations represented by the sequence of edges between the closest state of an object in the pre-processing graph, and the state in which the queried content can be identified. For example, Figure 3 shows that objects will be automatically pre-processed to nodes N_1 and N_2. If the user requests objects containing semantic information M, the processing operations between N_2 and M will be applied to the object state in N_2. If the operations are successful, the object will be labeled with M. MOODS then returns those objects in which the embedded information has been correctly identified.

This model of dynamic processing has several advantages. Postponing rarely needed operations limits processing done at data entry time. This means that MOODS doesn't waste time identifying useless metadata, and doesn't waste space storing and indexing metadata

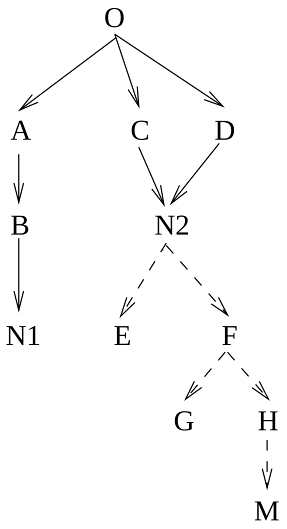

Figure 3: A sample processing graph. Solid lines represent the pre-processing component. The dashed lines are the search-processing component. Objects are automatically processed through the pre-processing stages (to nodes N1 and N2). Further processing is postponed until required.

that won't be needed. Furthermore, performing preprocessing helps limit the data that has to be processed at runtime. When users make queries, MOODS does not start randomly searching all objects in the database. Rather, MOODS only processes data in the pre-processing states that is known to lead to the correct node in the search-processing graph. MOODS does not try to process data that has been shown not to contain the information that the user requested (i.e., we know that it will never reach the desired node in the processing graph).

6 Identifying High-Level Semantic Concepts

Although processing can identify whether or not an image contains a certain person (e.g., Bill Clinton), processing alone cannot express high-level conceptual ideas or classifications such as "the identified person is a politician", nor can processing alone deduce, conclude or reason about the information found in the data. The problem is that processing operations only give facts about the data, but cannot draw conclusions or derive information about higher-level semantic concepts. However, these high-level semantic features are often important to users, and so we need a way of identifying them.

MOODS allows users to define a knowledge base of *inference rules* to help identify semantic content that currently cannot be identified through processing alone. An inference rule specifies the high-level semantic content that can be inferred to exist, given the fact that certain semantic content has already been identified in an object.

Inference rules consist of propositional logic statements, and are of the form: $high\_level\_semantic \leftarrow semantic\_expression$, where $high\_level\_semantic$ is the content that we want to identify, and *semantic_expression* specifies the conditions that must hold in order to make the inference. The semantic expression is given in terms of semantic labels that have already been identified.

The simplest rule is an alias such as *canine* $\leftrightarrow$ *dog*. Aliases imply equivalence between semantic labels. For example, if we have an object with the label "dog", we can conclude the object contains a "canine", and visa versa. More complex rules can be written that take advantage of the power of the boolean and comparative expression. For example, the rule *dog* $\leftarrow$ *four_legs && tail && hairy && height* $>$ *1* says that if an object has the semantic labels *four_legs*, *tail*, and *hairy*, and the height of the object is greater than 1, then we can conclude that the object contains a dog.

Semantic descriptions in the expression can also refer to other inference rules. Continuing with the example, four_legs might itself be a rule defined by four legs connected to the same body. Ultimately the processing must be able to identify regions of a figure and determine their connectedness.

7 Database Queries

The creation of an intelligent multimedia management system necessarily requires a more powerful database engine. Most significantly, the database must be tightly-coupled with the processing engine. The database must know about semantic objects, semantic states, processing paths, etc.

Not only does the database have a set of known information (semantic information that has already been identified), but the database also has the potential to learn about more (new) semantic information. New information can be extracted, assuming the user is willing to pay the cost of processing.

The coupling of the database and processing engine also means that the database can accept queries for high-level conceptual information (using the semantic inference rules) as well as searches for semantic features in the data. This model of data discovery is significantly different than conventional databases.

The expression of semantic information (i.e., defining a processing path) consumes no CPU time and little storage space. Thus, people will be inclined to define large amounts of semantic information which can potentially be searched. In general, the amount of known semantic information will be significantly less than the amount of searchable semantic information. This implies that as the system matures, the amount of searchable information will continue to grow. This is in stark contrast to conventional databases in which all information is specified a priori.

When users issue queries, MOODS first looks at the known facts currently found in the database. From this set of facts, MOODS derives more information using the semantic inference rules. If any of the facts match the user's query, the results are returned.

However, if no facts match the query, or the user is not happy with the results, object processing will be required. MOODS starts by first looking for a semantic state in the search-processing component of the processing path that matches the query. If a matching state is found, objects are processed further by invoking the sequence of processing operations specified in the path.

If still no results match the user's query, MOODS starts following the semantic inference rules on the new set of facts generated from processing. Note that following an inference rule may point to other inference rules, or may refer to facts yet to be found. This effectively starts a new search for a different fact that might cause the desired information to be found.

8 Prototype System and Example Application

To evaluate our design and semantic data model, we have developed a new language which contains support for semantic objects, logical function groups, and processing graphs. We have incorporated this into a GUI based on X-Windows Tcl/Tk coupled with a database for search and retrieval. We call the entire system MOODS and have used MOODS to implement a paleographic tool for ancient manuscripts, in particular, the Beowulf manuscript. Identification of *letterforms* allows paleographers to capture differences in writing styles, and gives clues to the authorship of a folio.

The MOODS language is an extension of C++ with support for semantic data groups, logical function groups, dynamic object composition, and dynamic function invocation. The MOODS compiler is implemented as a preprocessor whose output is then passed through a standard C++ compiler.

Creating a new application typically requires creating a set of specifications (SPEC_FILE) that define the data to be operated on and the semantic processing stages the data may traverse over its lifetime. Each semantic state defines the logical operations that make sense in that state. The MOODS preprocessor takes a SPEC_FILE and a library of processing functions as input and produces C++ code that will allow users to interactively control the semantic identification process. The resulting C++ code is compiled and then linked with a GUI library to produce 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 it's pop-up menus 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.

Finally, MOODS supports a conventional database for multimedia data storage and retrieval. We have used both the Illustra system [7] and our own simple database storage system for this purpose. Many other databases could be used as well.

Figure 4 shows a segment of a page from the Beowulf manuscript. The goal is to identify and highlight various letterforms that appear. For example, identifying "æ" or "Œ".

The prototype uses image correlation routines, along with a sample letterform in order to match the sample with the letters in the manuscript. The correlation routine parameters allow the user to vary the sensitivity of the match — in essence, allowing the user to manipulate how "close" the letter must be to the sample. Once the letterforms have been

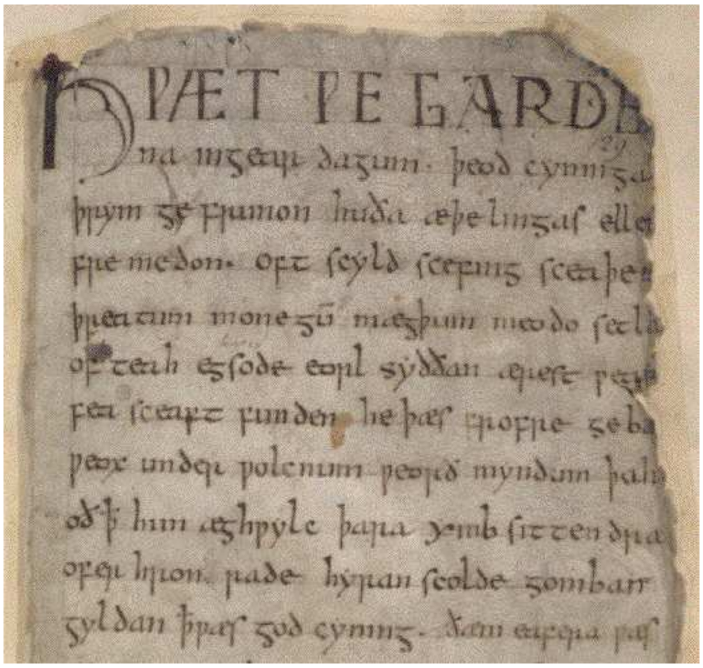

Figure 4: A portion of a manuscript page from Beowulf.

identified, they are extracted from the page and scaled for fine-tuned analysis.

The processing steps used to identify the letterforms is given in Figure 5 and consists of the following operations:

read: read a manuscript page from the disk

correlate: match a sample letterform against the letters of the page

expand/refine: extract the matching letterforms from the manuscript page, and scales them

Each link in the graph identifies a logical operation the can be applied at that semantic stage. The organization and semantic stages are defined by the expert of a domain just once. Later, novice users can use the graph to manipulate their own data, selecting the appropriate parameters for each logical function to identify the letterforms in the manuscript they are investigating.

Figure 6 shows the manuscript page after a search of Æforms. The results are then added back into the database for retrieval by other paleographers. If another paleographer is unhappy with the results, they can return to the processing engine to refine the data themselves. In this case, not all æletterforms are identified correctly via the automatic processing. Such differences may indicate things like multi-authorship which other scholars will certainly try to analyze further.

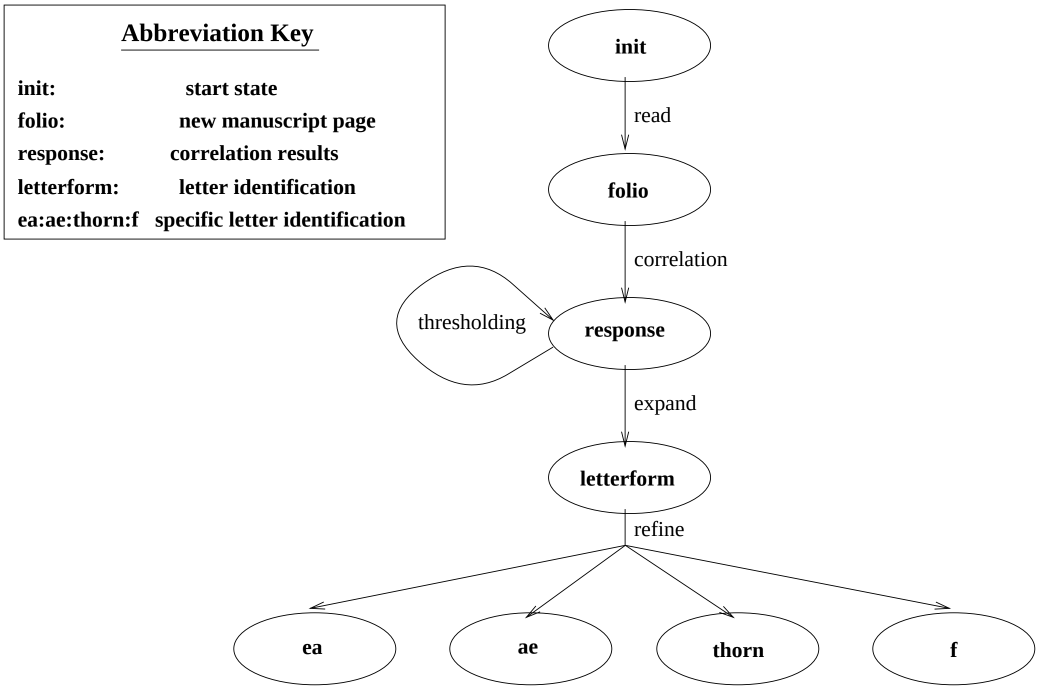

Figure 5: The static processing graph used to identify letterforms.

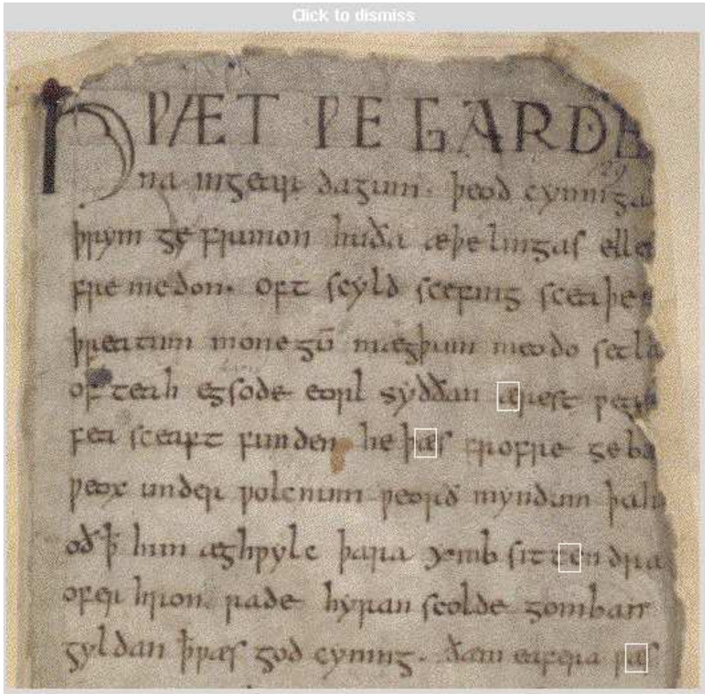

Figure 6: The manuscript page after Æ's are identified.

9 Related Work

Content based retrieval is an area of much current research. Unlink most of the work being done, MOODS is a framework for managing and modeling the semantic content. Most of the current systems propose techniques for extracting or identifying certain specific types of semantic content. Most of these techniques could be incorporated into the function groups of MOODS to enhance the types of semantics that can be extracted.

Several systems have proposed techniques for identifying or extracting specific semantic information. Srihari [1] showed that indexing is possible for images containing embedded caption information. Flickner et. al. [2] have developed a system for querying images based on image similarity measures, while Wu et. al [3] showed that similar image processing techniques can be used successfully for identification of faces.

Yoshitaka et. al. [8] demonstrated how a knowledge base can aid in video and image indexing. Similarly, Oomoto and Tanaka [9] illustrated how logically relating semantic descriptions can be used to formulate high-level concepts.

Finally, Smoliar and Zhang [4] showed how to analyze the semantic content of video clips using image processing routines, while Davis [10] illustrated a method of graphically querying semantic content in video information.

10 Conclusion

In conclusion, combining a multimedia database, object-oriented data model, and a dynamic processing engine, allows MOODS to identify and extract a wide spectrum of semantic information, from the most trivial, to the complex. Using multimedia processing operations and saving processing steps and parameter values in a path allows MOODS to automatically identify semantic content. Consequently, as research continues in the area of multimedia processing, the usefulness of MOODS will continue to grow. However, because of the amount of metadata that can potentially be identified, dynamically processing object at query time permits a wide range of users to access the data, without requiring them to sit through processing that they will never need. Finally, because users want to query data based on high-level semantic concepts, providing inference rules allows users to identify semantic concepts that cannot be identified through normal processing techniques alone. We have constructed a prototype framework and used the framework to implement data modeling systems for several application domains including music note analysis and paleography demonstrating the effectiveness of the approach.

References

[1] Rohini Srihari. Automatic Indexing and Content-Based Retrieval of Captioned Images. *IEEE Computer*, September 1995.

[2] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yankar. Query by Image and Video Content: The QBIC System. *IEEE Computer*, September 1995.

[3] J. Wu, Y. Ang, P. Lam, S. Moorthy, and A. Narasimhalu. Facial Image Retrieval, Identification and Inference System. In *Proceedings of ACM Multimedia 1993*, pages 47 – 55, 1993.

[4] S. Smoliar and H. Zhang. Content-Based Video Indexing and Retrieval. *IEEE Multimedia*, Summer 1994.

[5] A. Hampapur, R. Jain, and T. Weymouth. Digital Video Segmentation. In *ACM Multimedia 94*, 1994.

[6] Department of Electrical and Computer Engineering, University of New Mexico. *Khoros Systems User's Manual*, 1994.

[7] Illustra, Inc. *Illustra Multimedia Information System User's Guide*, 1994.

[8] A. Yoshitaka, S. Kishida, M. Hirakawa, and T. Ichikawa. Knowledge-Assisted Content-Based Retrieval for Multimedia Databases. *IEEE Multimedia*, Winter 1994.

[9] E. Oomoto and K. Tanaka. OVID: Design and Implementation of a Video-Object Database System. *IEEE Transactions on Knowledge and Data Engineering*, August 1993.

[10] M. Davis. Media Streams: Representing Video for Retreival and Repurposing. In *ACM Multimedia 94*, 1994.