Selection and Dissemination of Digital Video via the Virtual Video Browser[1]

T.D.C. Little, G. Ahanger, H.-J. Chen, R.J. Folz, J.F. Gibbon, A. Krishnamurthy, P. Lumba, M. Ramanathan, and D. Venkatesh

Multimedia Communications Laboratory
Department of Electrical, Computer and Systems Engineering
Boston University, Boston, Massachusetts 02215, USA
(617) 353-9877, (617) 353-6440 fax
tdcl@bu.edu

MCL Technical Report 01-20-1995

Abstract–In this paper we describe the Virtual Video Browser (VVB) software application designed to allow the interactive browsing and content-based query of a video database and to facilitate the subsequent playout of selected titles. The VVB is a manifestation of our mechanisms for the location, identification, and delivery of digital audio and video in a distributed system which can be extended to several application domains including multimedia-based home entertainment, catalog shopping, and distance learning.

The VVB employs a two phase retrieval process to serve its users. In the query phase, user queries are sent to a metadata server for processing. In the subsequent playout phase, a connection is established between the client workstation and a video server for the delivery of video data. The VVB incorporates a simple query interface that lets users specify their preferences to the system and retrieve the appropriate video. The application is designed to work in a distributed environment where video sequences are stored in different databases interconnected via a network. It has been shown to be a viable target application useful for investigating research problems related to building interactive multimedia systems.

Keywords: Video indexing, Internet-based services, metadata management, client-server architectures, video-on-demand.

---

[1]In *Journal of Multimedia Tools and Applications*, Vol. 1, No. 2, June 1995, pp. 149-172. Portions of this work were presented at the Intl. Workshop on Services in Distributed and Networked Environments (SDNE), Prague, Czech Republic, June, 1994, and at the 1st ACM Intl. Conf. on Multimedia, Anaheim CA, August 1993. This work is supported in part by the National Science Foundation under Grant No. IRI-9211165

1 Introduction

Multimedia computer applications are becoming one of the most dominant areas of development in the computer and information delivery arenas. One manifestation of this view is the development of systems supporting video-on-demand (VOD) [15, 25]. In a VOD system, video data are provided to a user community on a personalized basis. The home entertainment arena represents only a modest subset of the many possible interactive multimedia applications. Services including games, movies, home shopping, banking, health care, electronic newspapers/magazines, and classified advertisements represent some of the plethora of applications that can take advantage of the new technology.

In this context, we envision a VOD system[2] that allows the viewing preferences of each individual to be tailored and adapted to the available programming. Such a system would filter both stored and "live" information and offer these selections to the viewer instead of the many possible choices. Other features would permit more fine-grained information filtering. For example, specific topics of interest might be identified within a set of newscasts or documentaries. The realization of this vision requires identifying, and solving a diverse set of problems:

- A VOD system must possess the ability to capture and index large data sets for subsequent browsing and retrieval.
- Providing interesting, useful, and diverse means of accessing multimedia information from a very large realm of data necessitates mechanisms that allow searching for specific items in unstructured data.
- A VOD system must provide mechanisms to access enormous amounts of information, especially as derived from video data. This necessitates the development of techniques for efficient storage and retrieval.
- Construction and management of a high degree of interconnectivity and communication bandwidth among the different components of the VOD system.

Fig. 1 illustrates one view of a VOD system specific to the distribution of home entertainment [17]. This model consists of a local database connected via a high speed backbone

[2]In the remainder of this paper, we use VOD to describe any *interactive* multimedia system that encompasses video delivery.

network to archives that store multimedia information [11]. The local databases cache information locally (on a very large scale) and make them available to their users on-demand [29].

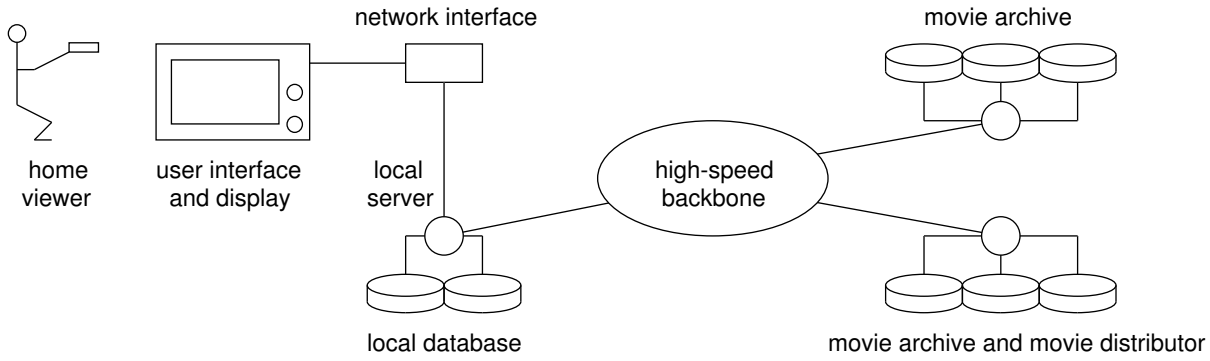

Figure 1: A Distributed Database Scenario for VOD

The advantage of this system architecture is its scalability. It is impractical to expect a single site to store all data and provide interactive services to thousands of users. This is because the support of a single VOD session[3] requires substantial storage and communication bandwidth [1, 26]. Local storage on the end user's computer is usually not feasible due to the cost. Moreover, the distribution of data and use of multiple access points increases the system's overall I/O capacity to support interactive sessions. In addition to the database systems that allow users to access information, there is a need to build a communication infrastructure that interconnects the different system components. The communication system must transfer large amounts of data in real time and provide guarantees on the delivery Quality-of-Service (QOS). This requires designing efficient algorithms to manage the flow of data within the network.

To access the large information space and support complex queries, it is essential to build schemes that model the content of unstructured data types such as audio and video [27]. These schemes must capture timing information necessary to support audio and video data delivery, playout, and queries based on temporal ordering. Research related to this effort includes modeling unstructured data for content-based retrieval [24, 35, 36, 37, 38], modeling the temporal component of multimedia data [12, 19], and the modeling of application-specific multimedia data [9, 22, 23, 28, 30, 31]. Approaches that tailor the output to the user are discussed by Lippman and Bender [14], Bender et al. [3], and by Loeb [22, 23].

[3]A session is described as a single, truly interactive connection from the system to the user.

Our objective in this work has been to demonstrate the utility of our models for time-dependent multimedia data [19] in constructing a VOD system. In this process we have sought to develop interesting access approaches for the retrieval of video information. The Virtual Video Browser (VVB) represents a testbed application designed to help us understand the end-to-end workings of a VOD system. Similar experimental testbeds and deployments of VOD are described by Rowe and Smith [28], Federighi and Rowe [10], Loeb et al. [23] and Keller and Effelsberg [13].

To achieve the aforementioned objectives, the VVB application characterizes motion video and its attributes to the level of *scenes* by using a data-specific schema. This information is stored in a relational database for subsequent interactive retrieval. Content-based retrieval is achieved using a domain-specific model used to characterize the specific video type (newscast, classroom, or movie). This database functionality is built on top of a metadata system model that supports location, identification, and delivery of multimedia data. This system is modeled on the client-server architecture in which a server coordinates different client requests. The VVB can support *temporal access control* operations based on data structures describing digital movies, classroom lectures, and newscasts and uses several tools that aid in the capture and indexing of video for content.

The remainder of the paper is organized as follows. In Section 2 we discuss the issues involved in building video databases and describe the basis for the construction of the VVB prototype. In Section 3 we describe the operation of the VVB and its implementation details. In Section 4 we discuss the lessons learned during the construction of the prototype and future directions. Section 5 concludes the paper.

2 VOD Databases

The support of interactive services requires the provision of mechanisms that support location, retrieval, and user temporal-access-control (TAC) functions. A VOD database server must also support time-dependent data delivery to many individuals over long-lived connections. To support complex user queries and browsing, the multimedia information system must first model, then store data in structures suitable for data playout. It must also provide retrieval mechanisms that deliver multimedia data-types to the consumer in real-time.

In this section we describe domain-specific conceptual models for video data and consider temporal and physical models for supporting the retrieval of time-dependent audio and

video data in content-based access. We then discuss issues of database management and implementation from the perspective of an end-to-end system.

2.1 Domain Specific Modeling of Video Data

Prior to the construction of any database application, one needs appropriate information models characteristic of the application domain. For a video browser application, the required models are application-related and media-related. Here we focus on the media-related models for video data. Application-related information models for the movie database are described in Section 3.

A video stream is typically modeled as a finite-length sequence of synchronized audio and still images.[4] These segments, if recorded contiguously in time, are called *shots* [9]. To provide useful access functionality, shots can be associated with information content. In addition, context can also be associated with shots based on relationships among video components. Embedding information about the content and context into the information models allows users to subsequently query for content and receive desired information.

To this basic shot component, attributes such as content, perspective, and context can be assigned, and later used to formulate a specific view on a collection of video. Aguierre Smith and Davenport [30, 31] use a technique dubbed *stratification* for aggregating collections of shots by contextual descriptions called *strata*. These strata provide access to frames over a temporal span rather than to individual frames or shot endpoints. This approach is applicable to classroom video recordings in which a query extracts a collection of shots associated with a particular topic of interest. However, it remains a time-consuming process to identify and assign context and content to the video components.

2.2 Information Extraction from Unstructured Data

In a conventional DBMS (e.g., using a relational model), access to data depends on distinct attributes for well-defined data developed for a specific application. For unstructured data such as audio, video, or graphics, similar attributes can be defined. However, the information contained in the data can be diverse and difficult to characterize within a single application [38].

[4]This characterization is a trivial instance of a timing relationship between media from the much larger set of relations possible among general multimedia objects.

The problem of extracting content and context from unstructured data can be approached by segmentation and feature extraction [2, 32]. Content of unstructured data such as imagery or sound is easily identified by human observation; however, few attributes lead to machine identification. For still and moving images, it is possible to apply segmentation to yield features such as shape, texture, color, relationships among objects, and events [4]. This process is very difficult to apply to complex scenes; however, it can be simplified with *site models* which characterize images *a priori* and allow detection of scene changes.

In our prototype system we currently use manual feature extraction of well-defined attributes which fit into a fixed data schema. The process of scene identification is semi-automated by detection of scene transitions [6]. A similar approach has been applied by Swanberg et al. [36, 37] and MacNeil [24]. In the former work, application domain-specific models are used to capture extracted information and to support content-based retrieval. In Steven's work in the ALT project at CMU [35], knowledge of content, structure, and use of the content is embedded in video objects. This approach is applied to the generation of video sequences using a rule-base and leads to the selection of seamless concatenation of video sequences from many small video shots.

2.3 Time-Dependent Data Support for Audio and Video

In addition to inter-media timing between audio and video in the aforementioned shots, there is a requirement to manage the timing among collections of shots (e.g., as *scenes*). These relationships can be sequential, as in the conventional video delivery model; parallel, with multiple concurrent playout; or arbitrarily complex combinations of the former. Moreover, the relationships can be conditional as indicated by a user's preferences and as explored via browsing. Both concurrency and and alternate sequence selection (browsing) are representable using the timed Petri net (TPN) and the hypertext paradigm [21, 34]. We illustrate an example in Fig. 2. In this example a collection of scenes is represented as places in the Petri net model. At the end of each scene, via user selection, a limited set of subsequent scenes can be viewed. Timing within the scenes can be managed by using an independent representation.

Few representation techniques specify appropriate data structures that support subsequent TAC operations on a database schema. Our approach is to use a mapping from a specification methodology to a database schema using the timed Petri net (TPN) and the relational database model [19, 21]. In this case, temporal intervals and relationships are de-

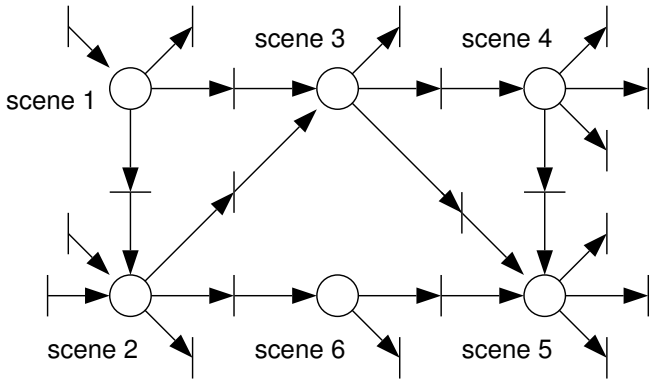

Figure 2: TPN for Representing Relationships Among Scenes of a Motion Picture

scribed by a timeline representation in an unstructured format, or by a TPN in a structured format. Using a TPN, temporal hierarchy can be imparted to the conceptual schema as sets of intervals bound to a single temporal relation. For a model as simple as a movie consisting of a sequence of scenes, we can directly create the conceptual temporal schema (Fig. 3). In a more general multimedia application this representation can be quite complex.

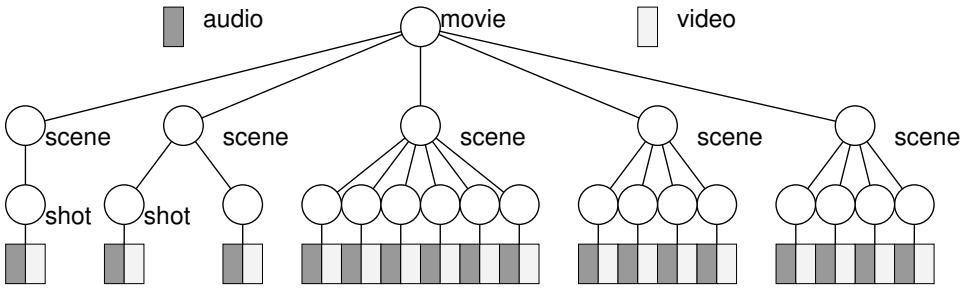

Figure 3: Temporal Schema for a Motion Picture

With this approach, the time-based representation is translated to a conceptual schema as a temporal hierarchy representing the semantics of the original specification approach. Leaf elements in this model typically represent base multimedia objects (audio, video, text, etc.), but are only audio and video (and subtitles) for the VOD application. Timing information is also captured with node attributes, allowing the assembly of component elements during playout. Most proposed models for multimedia data employ similar temporal-interval-based (TIB) representations (e.g., [8, 12]), including our model [19, 21].

Temporal intervals can be used to model multimedia presentation by letting each interval

represent the presentation or *playout* duration of some multimedia data element, such as a still image or an audio segment. The specification of timing for a set of intervals is achieved by indicating element durations for each data element and relative timing among elements.

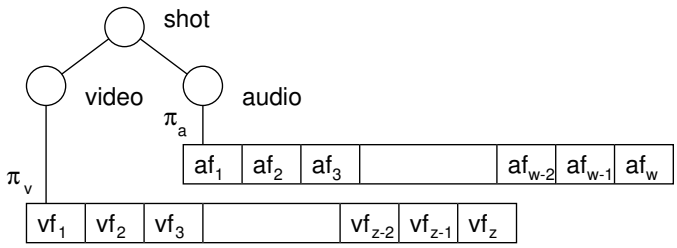

Figure 4: Blocking for Continuous Media

This TIB modeling scheme can represent an arbitrary grain of interval (e.g., individual video frames or entire movies). However, the practicality of representing each frame by its temporal parameters is limited for audio and video data. Therefore, we group logically related frames into sets called *shots*, that are not normally decomposed. This aggregation corresponds to Fig. 3, and yields a blocking of sequences of continuous media as shown in Fig. 4. Within a block, data are assumed to have a homogeneous temporal characteristic (i.e., period of playout).

2.4 Physical Data Storage

A mapping of the multimedia data to the physical storage system is necessary to facilitate subsequent access and retrieval. For time-dependent multimedia data this presents some interesting challenges. Problems arise due to the strict timing requirements for playout of time-dependent data. From our TIB model it is possible to derive a playout schedule appropriate for the available resources at the time of delivery. Converting the relative timing specification of the TIB model to an absolute one and scheduling this specification on the available resources achieves this purpose [20]. Furthermore, it is possible to derive an absolute schedule from the TIB representation in either the forward or reverse direction, or for only a fraction of the overall playout duration [19].

Besides intelligent data organization, suitable operating system behavior is necessary for multimedia data delivery. Timing relationships between audio and video are preserved during multimedia synchronization. Furthermore, in an application supporting audio and video playout such as our VVB prototype, nontraditional DBMS data access mechanisms

are necessary to support TAC functionality. These TAC operations include *reverse, fast-forward, fast-backward, midpoint suspension, midpoint resumption, random access, looping*, and *browsing*. These operations are feasible with existing technologies. However, when non-sequential storage, data compression, data distribution, multiple accesses on a server, and random communication delays are introduced, the provision of TAC capabilities can be very difficult. Examples include viewing a motion picture backwards or reversing an animation of a series of images, rapid viewing of a long sequence of images (fast-forward or fast-backward), and stopping and starting of a motion picture (midpoint suspension and resumption). Browsing and other operations involving random user interaction can be satisfied by data prefetching, caching and other approaches (e.g., Dannenberg, et al. [7]).

2.5 Metadata Services for Supporting Interactive Multimedia Applications

Along with efficient schemes that capture content and timing information from continuous media are the requirements for mechanisms that enable browsing and data delivery. To simplify the query mechanism and increase interactivity, most database systems employ a "metadata" mechanism. Metadata are "data about data." In our context, this means that they contain concise information about the location and characteristics of the data to be retrieved (e.g., movie titles, where they are stored) [18]. The metadata approach decouples the data delivery process from the database management functions.

With a metadata system the user can quickly go over an information summary and select the data to be delivered without retrieving the actual data objects. This concept is particularly appealing with respect to stored audio and video which can be costly to deliver and can consume significant portions of a data server's I/O bandwidth.

The metadata concept is also ideal for interactive multimedia applications in which many objects can be involved in a single presentation. For example, a movie might involve the coordination of audio and video streams, a classroom video might involve the simultaneous presentation of audio, video, graphics and text. The metadata server maintains the relationships between these diverse objects and informs the client of their presence whenever a query is made. When resources are overloaded or not available, the metadata server finds alternate sources of information, if available, or informs the user of their non-availability. The metadata server also keeps track of changes to the system configuration which can include resource relocation, failure, and the announcement of new resources.

Using the metadata model provides us with a simple scheme that is quite powerful. As database entries are used to maintain the relationships between objects, the metadata can be easily extended to support different media formats by object redirection. This is particularly appealing when supporting heterogeneous terminal devices with different display and presentation devices.

Associated with the problem of locating the desired data for the user is the related problem of distributing data across the system. In a distributed information system accessed by hundreds of users, there exists a need for efficient schemes that distribute data to storage sites where they are most likely to be used [17]. However, a detailed discussion of this issue is beyond the scope of this paper. We now examine the application of the models described in Section 2 in building a prototype VOD system, the VVB.

3 The Virtual Video Browser

The basic objective in the development of the VVB prototype has been to illustrate the capabilities of our temporal modeling scheme and to gain experience with an interesting multimedia application that requires the convergence of many technologies. Secondary objectives were to design a software system with various desirable properties including ease of use and understanding, ease of maintainability, portability, and extensibility.

The VVB operates in association with a distributed multimedia database. Movies and scenes of movies can be identified and viewed through the VVB based on movie-specific attributes including actor names, director names, and scene characteristics. Similarly, courses can be specified based on subject, instructor, and lecture-unit attributes. Video content indexing is facilitated by summary, keyword, and transcript searching. Besides viewing scenes, the user can access related textual information including summaries and transcripts of scenes, shots, etc. The VVB incorporates a simple query interface that lets users specify their preferences to the system to retrieve the appropriate video. The user interface was developed via approximately eight iterations of fast prototyping and user evaluation.

3.1 VVB Operation

The VVB is characterized by five operational modes and associated interfaces. These are: (1) category selection menu, (2) virtual video shelves, (3) query input, (4) query output, and

(5) video playout interfaces, described as follows.

The *opening menu* (Fig. 10) of the VVB displays images representing each of the available categories. The user can choose a category by the click of a mouse button. Each genre is identified by a picture. For example, the "western" category is indicated by a desert scene with cacti. Once a category is selected, the *virtual shelf screen* appears, showing an iconic representation of shelves of a video rental store.

The *virtual shelf screen* (Fig. 12) is a virtual display of the available movie cassettes as patterned after a video rental store.[5] Each video on the virtual shelf screen is represented by a rectangular icon that resembles the shape of an actual VHS video box. To help distinguish videos of different categories on the video shelf screen, each video on the screen contains an icon which represents an available category. If several videos are to be displayed, the user can move between multiple shelves using the arrows on the bottom of the screen. Selection of a movie icon brings up the query output and video playout screens. When the query button is pressed, the *query input screen* appears.

The *query input screen* (Fig. 11) allows the user to create database queries in the identification of scenes for viewing or to identify a desired movie. This interface permits the user to make queries to the database on any content within the realm of courses or movies. For example, using the VVB the following database queries are possible:

- Find the movie in which Humphrey Bogart says "Play it again Sam."
- Find scenes with boat chases involving James Bond.
- Find all course segments dealing with homework assignments in Geometry 101 the past 3 years.
- Find all slides related to "Gaussian Distributions" in all courses.

In addition to the iconic graphical representation of the selected video, the VVB can display any textual data from the database by using available attributes. This functionality supports quick scanning of the database to find a video of interest. The *query output screen* (Fig. 13) is invoked by performing database queries using the database query input screen. Each time the "Apply" button is depressed on the database query screen, a query is issued to the database manager and a list of movies is returned.

[5]Loeb [23] mimics a jukebox in a similar manner for the purpose of achieving rapid user acceptance.

The set of input and output application-specific relations accessible for query are *movie, scene*, and *actor*.[6] Their attributes are summarized in Table 1. Similarly, for classroom videos, application-specific relations accessible for query are *course, instructor*, and *lecture-unit*. Their attributes are summarized in Table 2.

Table 1: Motion Picture Database Attributes

Movie (M)	Scene (S)	Actor (A)
title	keywords	name
director	setting	date of birth
producer	summary*	sex
year	transcript*	
category		
summary*		
rating		
character		

Table 2: Classroom Video Database Attributes

Course	Instructor	Lecture-Unit
title	name	title
number	department	number
summary	sex	year
syllabus		semester
		summary
		keywords

Once a video or scene is selected with the mouse from the virtual shelf or query output screens, the *video playout screen* appears. This screen provides the visual interface for displaying different parts of a movie and supports TAC buttons. A user can scan from scene to scene beginning with the initial selection.

3.2 VVB Mechanics

The VVB system architecture is designed to support many video-terminals, or clients, and video-databases, or servers, interconnected via a computer network (Fig. 5). No particular assumptions are made about the underlying network in its design. A metadata server

[6]To satisfy the requirement for query input simplicity, we sacrificed the ability to support multiple same-attribute queries, even though the underlying data models and DBMS can support them.

(possibly distributed) called the query-database (QDB) contains the information about the availability of videos and their locations within the VOD system. It maintains the availability of different resources in the system and acts as a *name-server* that maps movies to the video databases.

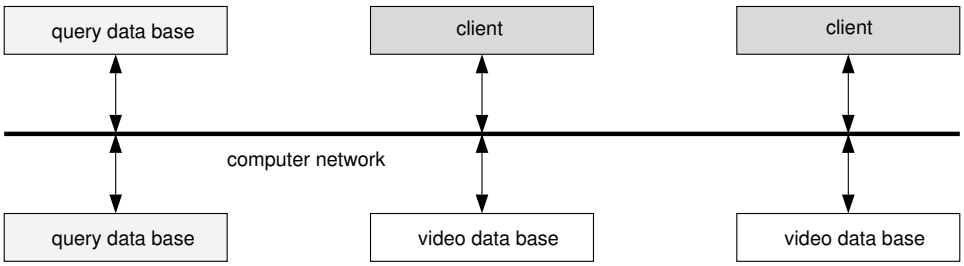

Figure 5: VVB Architectural Model

The video-databases (VDBs) contain the video data (movies) necessary for playout. The clients are provided with the necessary hardware/software to support the VVB user-interface and movie playout. For implementation purposes, it is assumed that while the QDB is located at a single site known to all network stations, the VDBs are distributed across many sites in the network. When the user wishes to view a video, a query is made first to the QDB for an interpretation of user browsing operations. The delivery of the video data to the user begins after the user has browsed the selections, issued a command to "play," and has been given permission to do so from the server. In the remaining subsections we examine the connection establishment and maintenance functions necessary to support these processes (Fig. 6).

3.2.1 Query Phase

The query phase involves the querying of the QDB to get all information required by the client to obtain a selected movie or scene. The query mechanism is implemented using a remote procedure call (RPC) between the client and the QDB.[7] A query phase is initiated when the user selects a category. The client system then queries the remote database (the QDB) for videos in the specified category. The user can get additional information on the selected videos including a summary, poster, or textual description of a scene by further querying the QDB for specific data. The QDB has the additional functionality of guiding the client to the VDB that contains the desired information. The QDB makes sure that the VDB being accessed has enough residual capacity to support the new connection before

[7]POSTGRES is the DBMS used in the VVB implementation of the QDB.

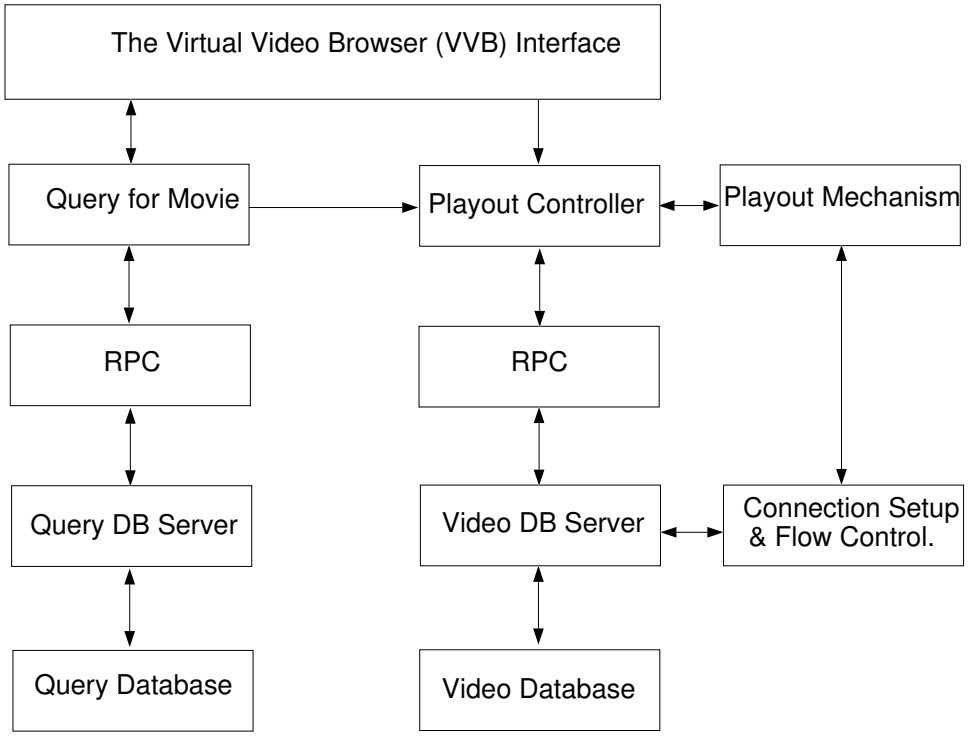

Figure 6: Mechanics of the VVB Operation

directing the client. The advantage of this approach is that the data location process becomes transparent to the user.

3.2.2 Connection Establishment

If the user decides to view a video sequence, the system enters the connection setup and playout phase. This phase is divided into two parts; the connection setup phase, and the connection management phase. During connection setup, the client checks with the VDB to see if it can support a new connection. If a connection can be set up, a negotiation phase is initiated during which QOS parameters are negotiated. If the negotiation proceeds to a successful conclusion, the connection setup and maintenance phases are initiated.

The client requests the establishment of a connection with the VDB specified in the query process. In the VVB, the delivery of video data to the client occurs over a network-connection that lasts the lifetime of the session. The advantage of this architecture is that it can support diskless clients that cannot cache the video data locally. Alternatively, the entire video file can be transferred and cached at the client workstation. However, there are

several disadvantages to this approach as described elsewhere [15].

The video server evaluates the system state to ensure sufficient resources are available for movie playout for the duration of the feature presentation. If the connection cannot be supported, the server sends back a "connection refused" message and the client must repeat the query to the QDB to find an alternate source of data.

Table 3: Typical Server Functions at Connection Setup

check_num_connections()	Check to see if a new connection can be supported.
retrieve_video_chars(video_name)	Retrieve video characteristics for QOS negotiation.
indicate_QOS_needs(max_rate, min_rate,...)	Indicate QOS Requirements.
increment_num_users()	Reserve resources.
start_QOS_negotiation(video_name, client_name)	Begin QOS negotiation.
setup_connection(video_name, client_name)	Spawn a process to setup dedicated UDP connections.

Table 4: Typical Client Actions During Connection Setup

query_video_database(database_name, video_name)	Query the VDB for the specified video.
examine_video_statistics(video_params)	Check to see if the connection can be sustained.
acknowledge_conn_acceptance()	Start the connection.
setup_playout_and_wait_for_connection()	Initialize the playout hardware and wait for the connection call from the remote server.

If the connection can be supported, the server then proceeds to examine the characteristics of the required video from the VDB. Depending on these characteristics, the server then starts a QOS negotiation procedure.[8] If the negotiation completes successfully, and mutually acceptable QOS parameters are agreed upon, the connection is established. Otherwise the connection is terminated and the resources are freed up. During the QOS negotiation phase, network and storage resources are reserved for the new connection and will be released only if the connection setup fails. This ensures that there are no deadlocks at the server.

Once the QOS parameters are established the server initiates a new process to handle the transfer of video data. A dedicated connection is set up between the client station and

[8]The QOS negotiation stage is not currently implemented on the VVB prototype.

the VDB. The functions that are required at the client and server machines for connection setup are summarized in Tables 3 and 4.

3.2.3 Connection Management

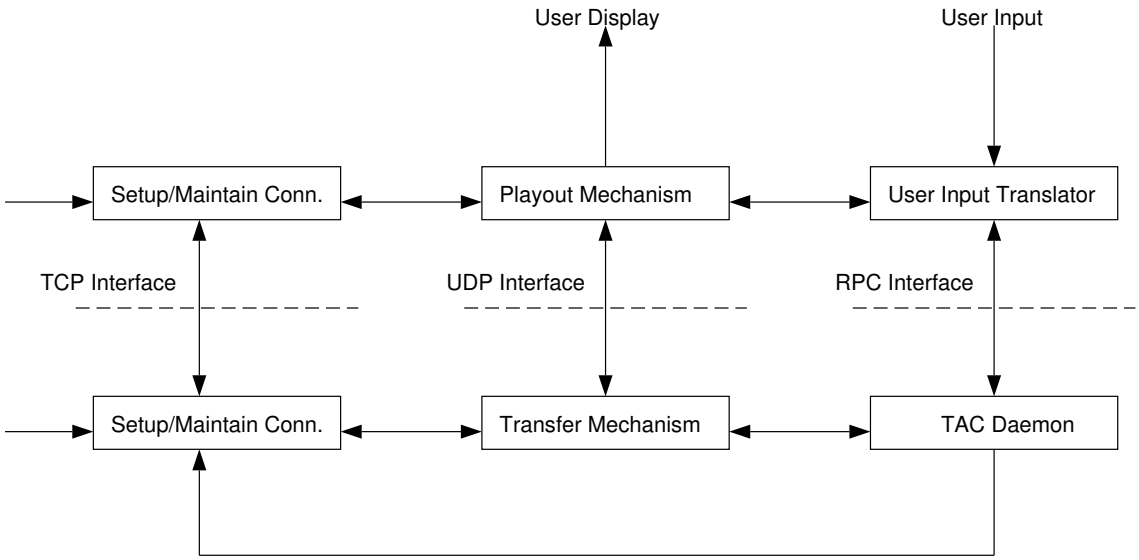

Figure 7: Connection Management for VVB

Connection management processes are responsible for ensuring that the QOS parameters negotiated at the time of connection setup are maintained during the lifetime of the connection. In addition, user inputs are translated and sent as control signals to both the server and the client for processing. Certain functions such as audio volume controls are dealt locally via the playout mechanism at the client. Other functions involve the delivery of the proper signal to the VDB server for appropriate action. The connection management flow is illustrated in Fig. 7. Some of the functions that are required for connection management and maintenance are summarized in Table 5.

In the VVB, the delivery of data between the client and the server takes place via a UDP connection. The choice of UDP was due to the need for supporting data delivery in real-time. Even though TCP is a more reliable protocol, it introduces additional delays due to retransmission which are not acceptable for real-time video communication. Furthermore, video data are tolerant to losses and we exploit this characteristic in our design. Flow control is facilitated by an out-of-band feedback circuit over a TCP connection. This is used

to ensure the timely delivery of video frames to the client. The system can adapt itself to load changes that can affect continuous media playout using this feedback mechanism.

Table 5: Connection Management Primitives for the VVB

playout_video()	Indicate to both the client and the server to ensure that frames are played out at the proper rate.
forward_video()	Indication to the server to playout every nth frame in lieu of consecutive frames.
reverse_video()	Analogous to forward_video().
pause_video()	Pause video stream. Directed toward the playout mechanism and the server.
change_volume()	Change volume at the client.

3.3 The VVB Database

The VVB database is implemented using the POSTGRES DBMS [33] and several data schemata. These schemata provide application-specific data access based on content as well as TAC functionality through our temporal models. A third data schema relates the application-specific data model to the temporal model. These schemata and their relationships are illustrated in Fig. 8 using a network representation.[9]

3.3.1 Database Schema

The application-specific schema defines the VVB system application database to which queries can be made. For movie-specific data, it contains six components, implemented as relations: movie, scene, actor, movie-scene, scene-actor, and movie-actor. The actor relation describes actors and actresses. The movie-scene relation defines the parent-child relation which identifies all movies and their scenes. The scene-actor relation defines the relation to identify which actors are in which scenes. The movie-actor relation defines the relation to identify which actors are in which movies. Example data in these fields are illustrated in Tables 5–8.

The temporal schema captures the relative timing relationships between components of the movies as described in Section 2.3. In order to relate the application-specific schema

[9]This structure is rendered as a relational model for the database management system (DBMS) in POSTGRES.

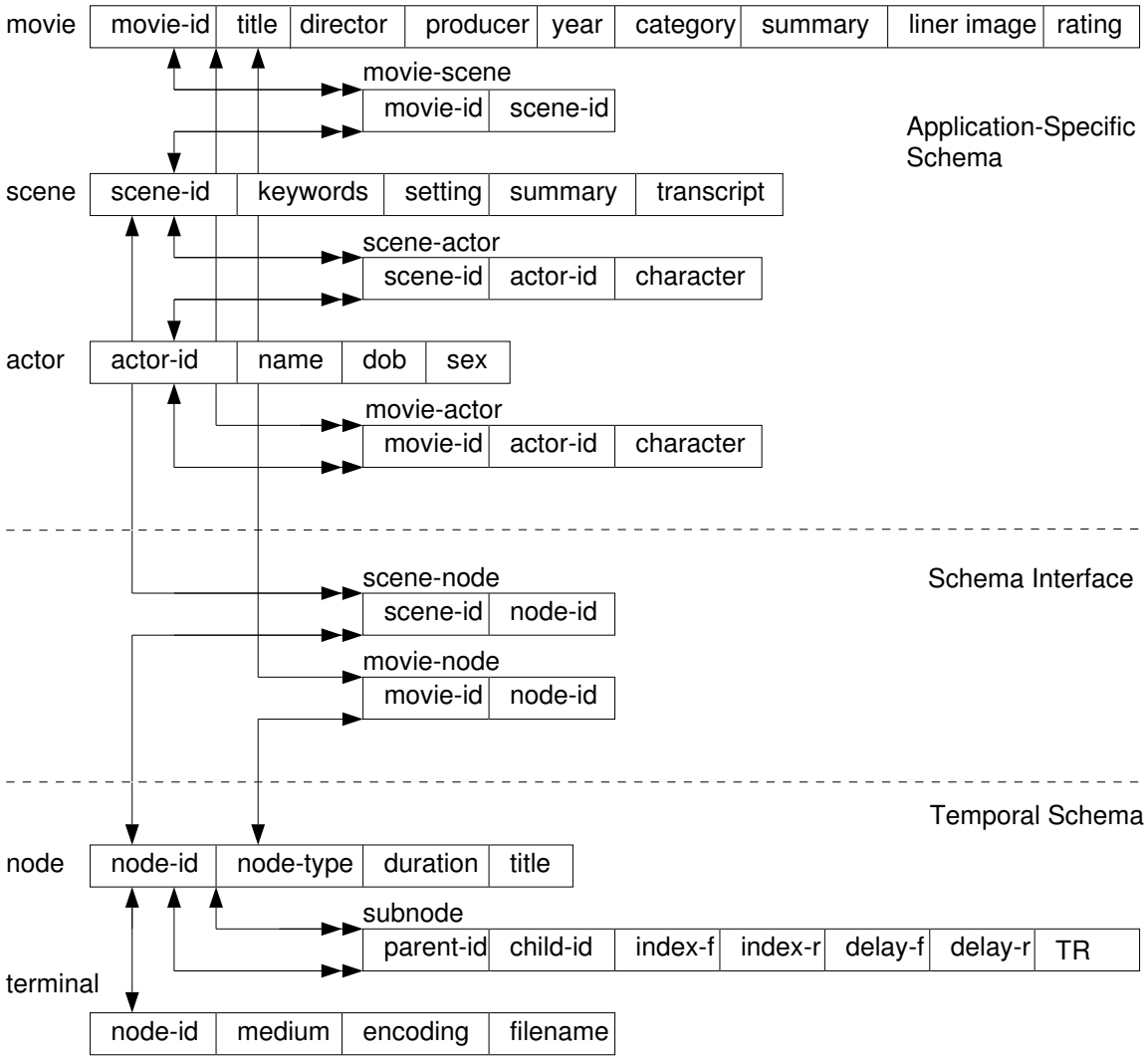

Figure 8: Database Schema for VVB System

Table 6: Example Data Fields: Actor

actor_id	name	date of birth	sex
001	Sean Connery	August 25, 1930	male
002	Ursula Andress	March 19, 1936	female

Table 7: Example Data Fields: Movie_Actor

movie_id	actor_id	character
001	001	James Bond
001	002	Honeychile Ryder

Table 8: Example Data Fields: Movie

movie_id	title	director	producer	year	category	summary	liner
001	Dr. No	Young	Broccoli	1962	action	sum1	img1

for a video database to the temporal schema we use the schema interface. For the VVB application, this component consists of two relations, scene-node and movie-node, as defined in Fig. 8.

3.3.2 Query Engine

The query engine for the VVB records the input selection from the query input interface in a data structure called DBQin. For each selected input attribute, a corresponding component is generated in the Postgres-Query-Language (PQUEL) query. Similarly, each selected output field results in a term in the PQUEL query. The make_query subprogram then formulates the PQUEL text string as:

retrieve <determined by output selections >
from < m in movie, s in scene, a in actor>
where <determined by input selections>

The text string is built by retrieving from tables the appropriate strings and adding the user input text as appropriate. This final PQUEL string is passed to the POSTGRES DBMS

Table 9: Example Data Fields: Scene

scene_id	keywords	setting	summary	transcript
001	license to kill	Crab Key Island	Bond swimming	Oh James
0010	skiing	Swiss chalet	Bond skiing	Shall we ski?...

where it is executed, resulting in the return of the selected output attributes.

3.4 Software Architecture

In order to meet our objectives of application flexibility, extensibility, portability, and maintainability, we have designed the VVB system software in a layered fashion. This simplifies the replacement of software components with changing technology. For example, we can easily replace the video compression/decompression subsystem to accommodate new algorithms and hardware improvements.

The software design of the VVB system is divided into three distinct components [18]. These components are the user interface, the database, and the imaging system. An application programming interface (API) has been created to provide functionality in each case. The actual VVB application is implemented on top of the APIs to satisfy the VVB requirements. A particular advantage with this design methodology is the ability to create new applications from existing code because most of the VVB functionality is implemented within APIs. Below is a diagram of the three different APIs and how they relate to the commercial off-the-shelf (COTS) products and the VVB application (Fig. 9).

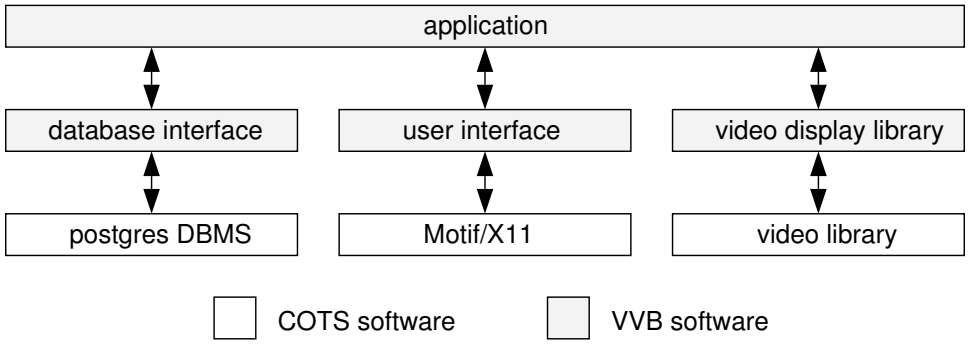

Figure 9: Software Architecture of the VVB System

The existing VVB system is implemented in C on a Unix platform and uses X11, Motif and POSTGRES. Digital video recording and playback are facilitated by XVideo hardware from Parallax Graphics Inc. Currently the VVB supports the delivery of M-JPEG data. We are scaling the application to support other encoding formats.

4 Discussion

The VVB was developed primarily as an application to demonstrate our research results from our earlier work in the modeling of time-dependent data. To this end it has satisfactorily achieved its purpose; we are able to specify and store temporal relationships and use them in media playout. However, the inherent sequential characteristics of conventional videos limit the application of our temporal models. Entertainment video is not a multimedia-rich environment as movies are homogeneous and sequential in nature. For this reason we are pursuing additional applications that require the full range of TAC functionality. Educational material (classroom videos with associated text and stills) presents us with a much more diverse data set to experiment with.

The VVB can also be evaluated from several other perspectives. In particular, these are (1) number of supported users, (2) database retrieval performance, (3) ease of use, (4) diversity of semantic database content, and (5) complexity of database maintenance.

In our current configuration, the number of supported users is limited by the availability of compatible video decompression hardware. With additional workstations served by the same video database, the available bandwidth of the communication medium (Ethernet in our testbed) and the I/O performance of the database server become the performance bottlenecks. Selection of number and performance of storage and communications components in a VOD scale-up is dependent on component costs and degree of supported VOD functionality. For example, our studies indicate the feasibility of supporting up to five true-VOD sessions (users) from a conventional hard disk storage device [5]. Performance limitations are primarily due to component bandwidth limitations and not the behavior of the VVB interface. The VVB has been successful in attaining our objective of ease of use. This quality, necessary for the VVB to gain user acceptance, was achieved through the development of a simple human-computer interface using fast prototyping and numerous iterations with user feedback.

In terms of diversity of supported semantic content, the VVB is limited by our choice of domain-specific attributes. In order to support a wide range of queries (e.g., "find a scene containing a dog"), attributes are required to be identified *a priori*, or supported dynamically. The former requires knowledge of a large set of potential queries. A dynamic approach would search for content in available data including in the transcripts, audio tracks and video segments. The VVB can find the scene with the dog assuming that "dog" is identified in the summary/transcript for the appropriate scene. However, a more interesting solution would

interpret a diverse set of input queries and apply them on the unstructured audio and video data as well as to an established semantic net containing pre-extracted information. This requires image segmentation and feature extraction which are currently not implemented in the VVB.

Database maintenance is also an issue that is closely related to the problem of dynamic attribute identification. As videos are created and added to the VVB database, video content must be extracted and indexed. This process is currently not automated. However, we are currently developing a semi-automated procedure for parsing movies using image and audio processing. As a basis, we have successfully used video frame size dynamics to identify scene transitions in an application called the Motion Picture Parser (MPP) [6].

5 Conclusion

In this paper we have overviewed the various technologies required to support a content-based retrieval from a multimedia information system encompassing video data. We have also presented a distributed database organization for multimedia information systems based on the use of metadata and resource servers. We have identified and illustrated the service primitives required for the management of continuous-media sessions in a distributed system.

The concepts presented are illustrated by example of the prototype VVB application. The VVB uses a client-server architecture to support browsing of a metadatabase of video databases and to set up video sessions to remote video servers. The system demonstrates our conceptual modeling scheme for time-dependent multimedia data and serves as a basis for a more general system supporting diverse multimedia data types.

The VVB uses a two phase process to serve its users. These are (1) a query phase during which user queries are sent to the metadata server and (2) a connection establishment and maintenance phase, during which sessions are set up between the video server and the client machine. The VVB is currently in use on a testbed of Unix workstations interconnected via Ethernet. The VVB interface has also been reimplemented to operate on the World Wide Web (WWW) using a forms-based interface. To be used as a viable alternative to applications such as courses-on-demand, the VVB needs to support a large number of users and a rich data archive. We are currently considering the effort required for such a scale-up. In the future, we plan to extend our system to support semantic models for information capture, automatic feature extraction/segmentation, and additional applications.

Acknowledgments

The development of the VVB prototype, including data collection, was made possible by the significant efforts of the following individuals: E. Deardorff, M. Herman, J.D. Marshall, A. Mol, G. Orlando, F.W. Reeve, and D.H. Schelleng, and R. Walzer. We also acknowledge the support of the National Science Foundation under Grant No. IRI-9211165.

References

[1] D.P. Anderson, and G. Homsy, "A Continuous Media I/O Server and Its Synchronization Mechanism," *Computer*, Vol. 24, No. 10, October 1991, pp. 51-57.

[2] F. Arman, A. Hsu, and M.-Y. Chiu, "Image Processing on Compressed Data for Large Video Databases," *Proc. ACM Multimedia*, Anaheim, CA, 1993, pp. 267-272.

[3] W. Bender, H. Lie, J. Orwant, L. Teodosio, and N. Abramson, "Newspace: Mass Media and Personal Computing," *Proc. Summer 1991 Usenix Conf.*, Nashville, Tennessee, June 1991, pp. 329-348.

[4] M.J. Carlotto, "Image Understanding Now and in the Future," *Advanced Imaging*, Vol. 7, No. 8, August 1992, p. 26.

[5] H.-J. Chen and T.D.C. Little, "Physical Storage Organizations for Time-Dependent Multimedia Data," *Proc. 4th Intl. Conf. on Foundations of Data Organization and Algorithms (FODO'93)*, Evanston, IL, October 1993.

[6] E. Deardorff, T.D.C. Little, J.D. Marshall, D. Venkatesh, and R. Walzer, "Video Scene Decomposition With the Motion Picture Parser," *Proc. of the IS&T/SPIE Conf.*, San Jose CA, February 1994.

[7] R.B. Dannenberg, T. Neuendorffer, J.M. Newcomer, D. Rubine, and D.B. Anderson, "Tactus: Toolkit-Level Support for Synchronized Interactive Multimedia," *ACM/Springer Multimedia Systems*, Vol. 1, No. 2, 1993, pp. 77-86.

[8] Committee Draft Intl. Standard, "Information Technology – Standard Music Description Language (SMDL)," ISO/IEC CD 10743, April 1, 1991.

[9] G. Davenport, T.G.A Smith, and N. Pincever, "Cinematic Primitives for Multimedia," *IEEE Computer Graphics & Applications*, July 1991, pp. 67-74.

[10] C. Federighi and L.A. Rowe, "A Distributed Hierarchical Storage Manager for a Video-on-Demand System," *Storage and Retrieval for Image and Video Databases II, IS&T/SPIE Symposium on Elec. Imaging Sci. & Tech.,* Vol. 2185, SPIE, Bellingham, Wash., 1994, pp. 185-197.

[11] A.D. Gelman, H. Kobrinski, L.S., Smoot, S.B., Weinstein, "A Store-and-Forward Architecture for Video-On-Demand Service," *Proc. ICC*, 1991, pp. 27.3.1-27.3.5.

[12] R.G. Herrtwich, "Time Capsules: An Abstraction for Access to Continuous-Media Data," *Proc. 11th Real-Time Systems Symp.*, Lake Buena Vista, FL, December 1990, pp. 11-20.

[13] R. Keller and W. Effelsberg, "MCAM: An Application Layer Protocol for Movie Control, Access, and Management," *Proc. ACM Multimedia 93*, August 1993, pp. 21-29.

[14] A. Lippman and W. Bender, "News and Movies in the 50 Megabit Living Room," *Globecom '87*, Tokyo, Japan, November 1987, pp. 1976-1981.

[15] T.D.C. Little and D. Venkatesh, "Prospects for Interactive Video-on-Demand," *IEEE Multimedia*, Vol. 1, No. 3, Fall 1994, pp. 14-24.

[16] T.D.C. Little and D. Venkatesh, "Client-Server Metadata Management for the Delivery of Movies in a Video-on-Demand System," *Proc. of the 1st Intl. Workshop on Services in Distributed and Networked Environments (SDNE)*, June 27-28, Prague, Czech Republic, 1994, pp. 11-18.

[17] T.D.C. Little and D. Venkatesh, "Probabilistic Assignment of Movies to Storage Devices in a Video-on-Demand System," To appear in *ACM/Springer Multimedia Systems*, 1995.

[18] T.D.C. Little, G. Ahanger, R.J. Folz, J.F. Gibbon, F.W. Reeve, D.H. Schelleng, and D. Venkatesh, "A Digital Video-on-Demand Service Supporting Content-Based Queries," *Proc. of the 1st ACM Intl. Conf. on Multimedia (ACM Multimedia'93)*, Anaheim CA, August 1993, pp. 427-436.

[19] T.D.C. Little and A. Ghafoor, "Interval-Based Temporal Models for Time-Dependent Multimedia Data," *IEEE Transactions on Data and Knowledge Engineering*, Vol. 5, No. 4, August 1993, pp. 551-563.

[20] T.D.C. Little and A. Ghafoor, "Scheduling of Bandwidth-Constrained Multimedia Traffic," *Computer Communications*, Vol. 15, No. 6, July/August, 1992, pp. 381-387.

[21] T.D.C. Little and A. Ghafoor, "Synchronization and Storage Models for Multimedia Objects," *IEEE J. on Selected Areas in Comm.*, April 1990, pp. 413-427.

[22] S. Loeb, "Delivering Interactive Multimedia Documents over Networks," *IEEE Communications*, Vol. 30, No. 5, May 1992, pp. 52-59.

[23] S. Loeb, R. Hill, and T. Brinck, "Lessons from LyricTime: A Prototype Multimedia System," *Proc. 4th IEEE ComSoc Intl. Workshop on Multimedia Communications (Multimedia '92)*, Monterey, CA, April 1992, pp. 106-113.

[24] R. MacNeil, "Generating Multimedia Presentations Automatically using TYRO, the Constraint, Case-Based Designer's Apprentice," *Proc. 1991 IEEE Workshop on Visual Languages*, Kobe, Japan, October 1991, pp. 74-79.

[25] S. Ramanathan and P.V. Rangan, "Architectures for Personalized Multimedia," *IEEE Multimedia*, Vol. 1, No. 1, Spring 1994, pp. 37-46.

[26] P.V. Rangan, H.M. Vin, and S. Ramanathan, "Designing an On-Demand Multimedia Service," *IEEE Communications Magazine*, Vol. 30, No. 7, July 1992, pp. 56-64.

[27] L.A. Rowe, J.S. Boreczky, and C.A. Eads, "Indexes for User Access to Large Video Databases," *Storage and Retrieval for Image and Video Databases II, IS&T/SPIE, Symp. on Elec. Imaging Sci. & Tech.*, Vol. 2185, SPIE, Bellingham, Wash., 1994, pp. 150-161.

[28] L.A. Rowe and B.C. Smith, "A Continuous Media Player," *Proc. 3rd Intl. Workshop on Network and Operating System Support for Digital Audio and Video*, La Jolla, CA, November 1992.

[29] W.D. Sincoskie, "System Architecture for a Large Scale Video On Demand Service," *Computer Networks and ISDN Systems* Vol. 22, 1991, pp. 155-162.

[30] T.G. Aguierre Smith and G. Davenport, "The Stratification System: A Design Environment for Random Access Video," *Proc. 3rd Intl. Workshop on Network and Operating System Support for Digital Audio and Video*, La Jolla, CA, November 1992.

[31] T.G. Aguierre Smith and N.C. Pincever, "Parsing Movies in Context," *Proc. Summer 1991 Usenix Conf.*, Nashville, Tennessee, June 1991, pp. 157-168.

[32] S.W. Smoliar and H.-J. Zhang, "Content Based Video Indexing and Retrieval," *IEEE Multimedia*, Vol. 1, No. 2, Summer 1994, pp. 62-72.

[33] M. Stonebraker and G. Kemnitz, "The POSTGRES Next Generation Database Management System," *Communications of the ACM*, Vol. 34, No. 10, October 1991, pp. 78-92.

[34] P.D. Stotts and R. Furuta, "Petri-Net-Based Hypertext: Document Structure with Browsing Semantics," *ACM Trans. on Office Automation Systems*, Vol. 7, No. 1, Jan. 1989, pp. 3-29.

[35] S.M. Stevens, "Embedding Knowledge in Continuous Time Media," *Proc. 2st Intl. Workshop on Network and Operating Support for Digital Audio and Video*, Heidelberg, Germany, November 1991.

[36] D. Swanberg C.F. Shu, and R. Jain, "Architecture of a Multimedia Information System For Content-Based Retrieval," *Proc. 3rd Intl. Workshop on Network and Operating System Support for Digital Audio and Video*, La Jolla, CA, November 1992.

[37] D. Swanberg, T. Weymouth, and R. Jain, "Domain Information Model: An Extended Data Model for Insertions and Query," *Proc. 1992 Workshop on Multimedia Information Systems*, Tempe, Arizona, February, 1992, pp. 39-51.

[38] T.M. Wittenburg and T.D.C. Little, "An Adaptive Document Management System for Shared Multimedia Data," *Proc. of the 1st IEEE Intl. Conf. on Multimedia Computing and Systems*, Boston, May 1994, pp. 245-254.

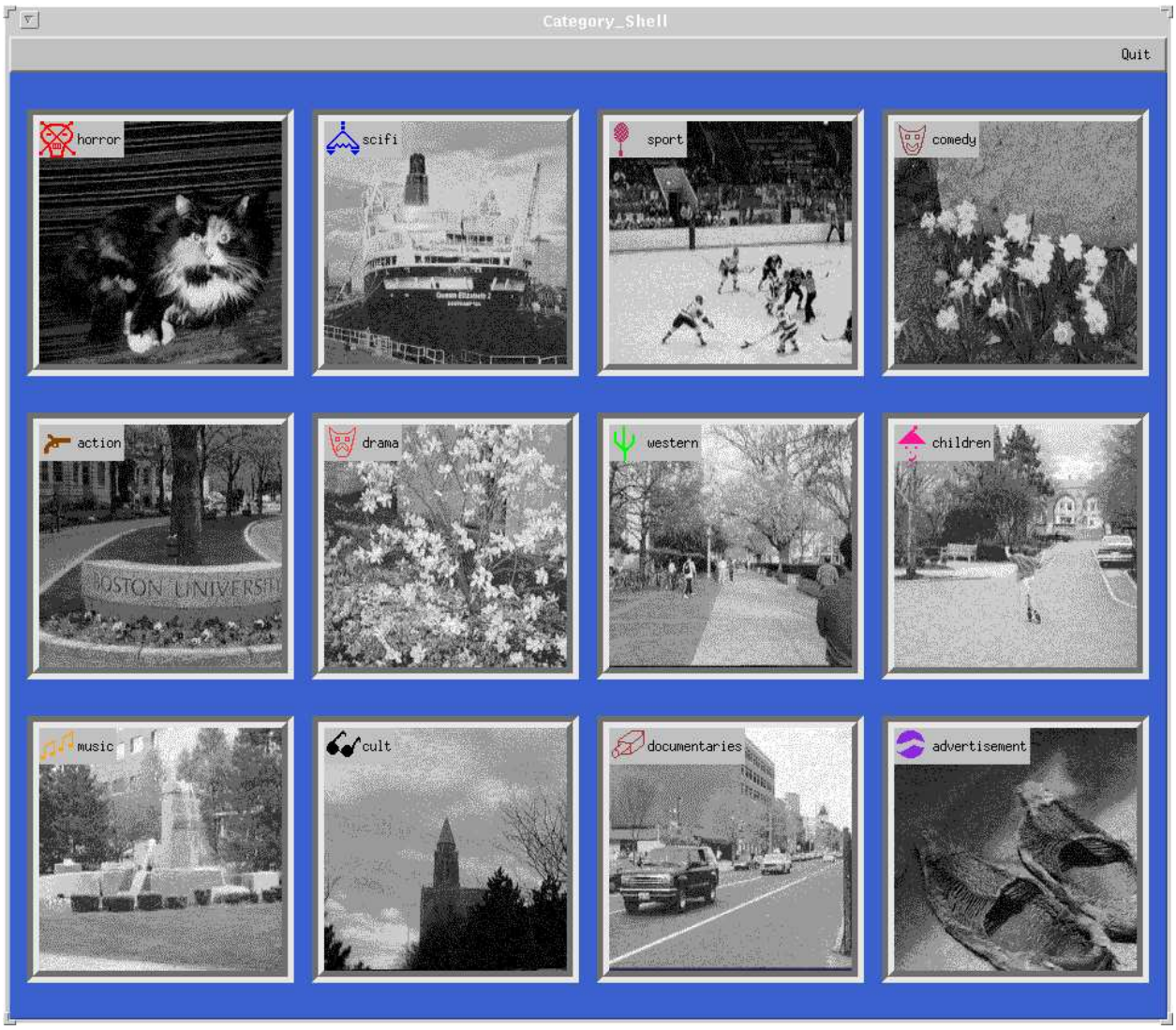

Figure 10: The VVB Input Screen

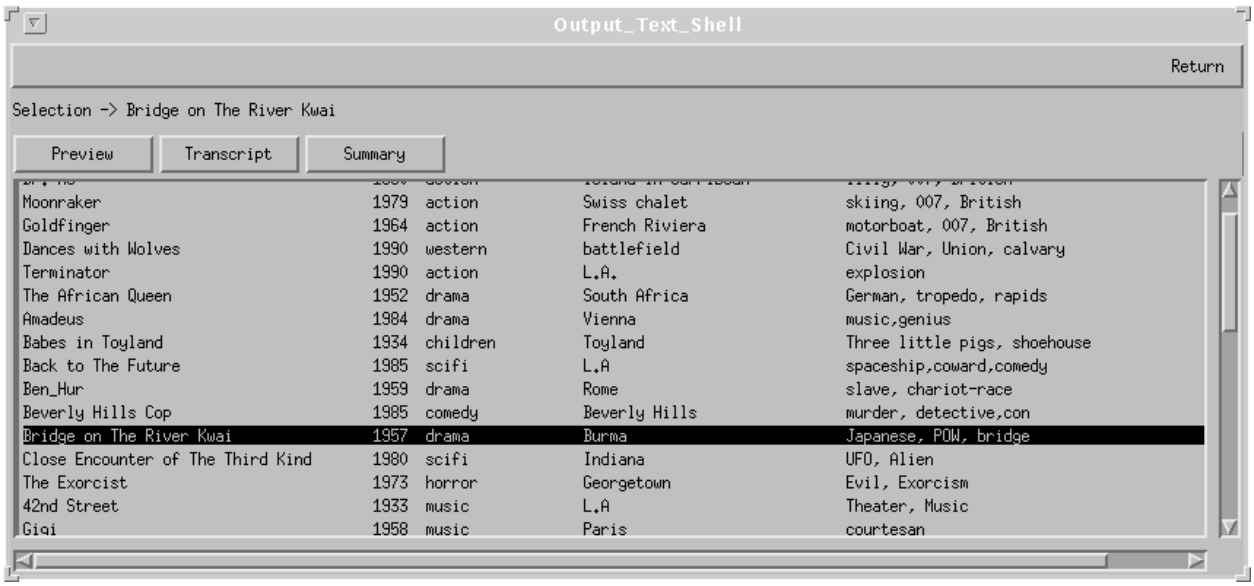

Figure 11: The VVB Query Screen

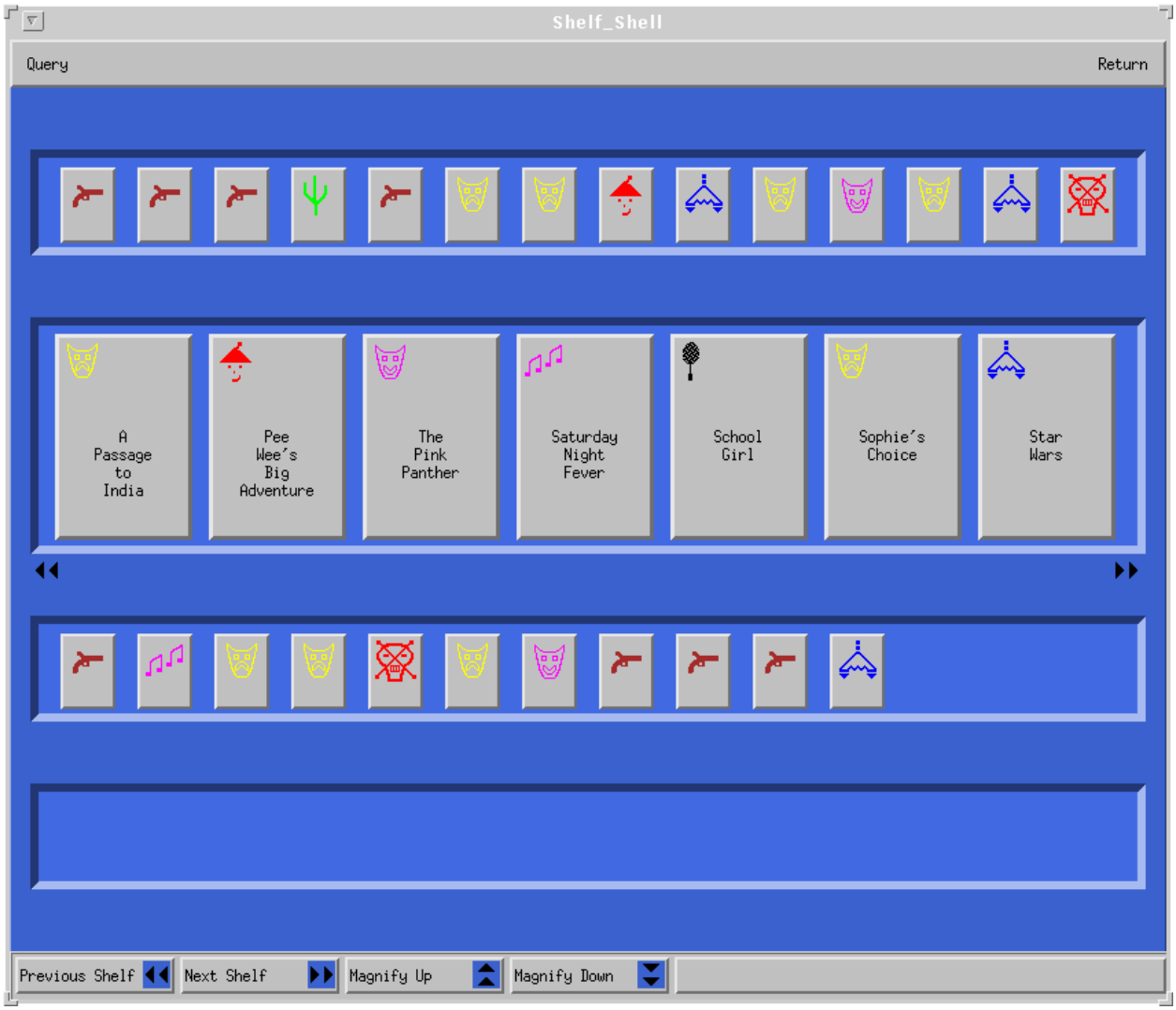

Figure 12: The VVB Shelf Screen

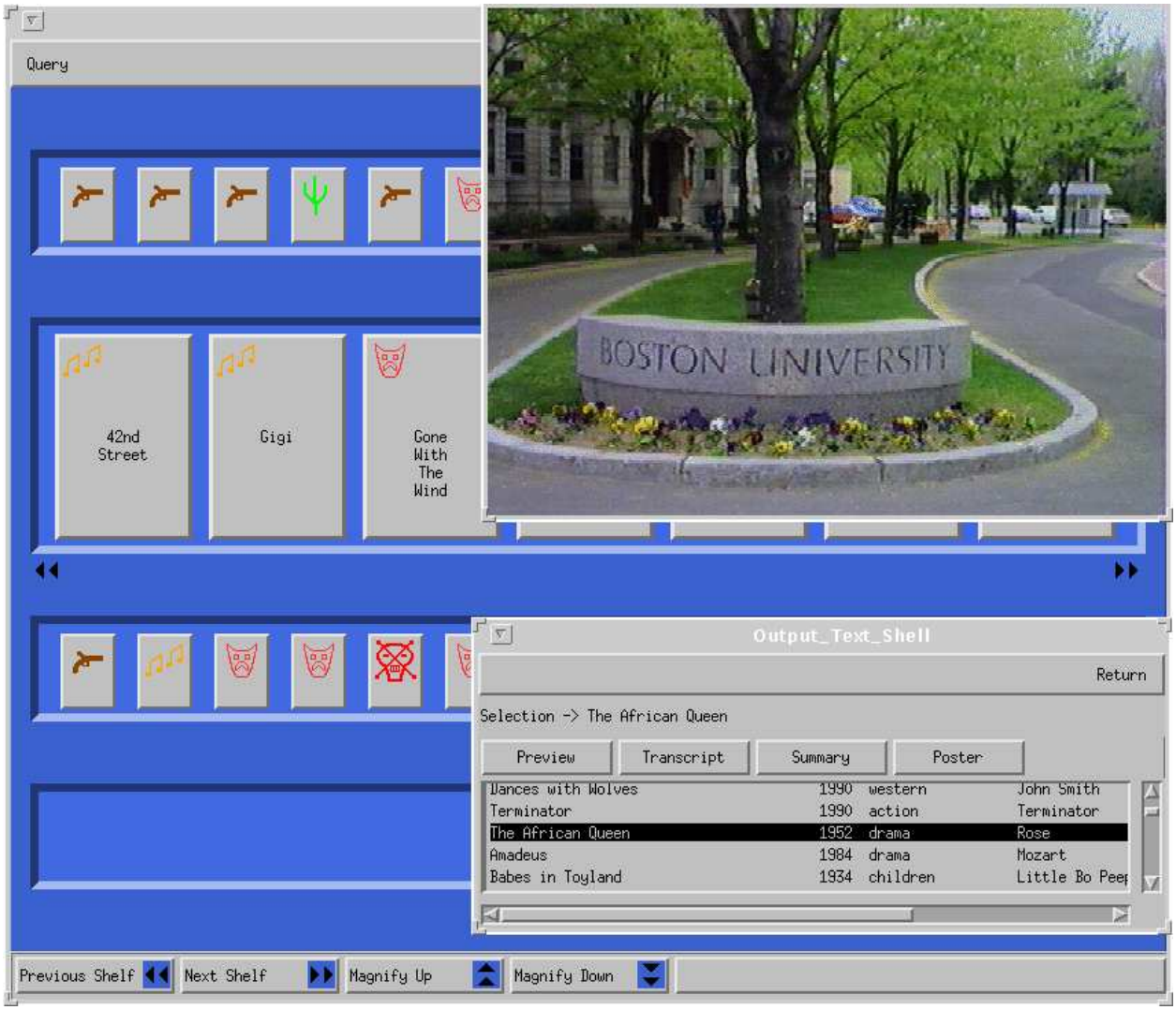

Figure 13: The VVB Playout Screen