Experiences with the Development of a Multimedia Information System for Medical Applications

Nipun Agarwal and Sang H. Son

Department of Computer Science
University of Virginia,
Charlottesville, VA 22903, U.S.A
E-mail : nipun@virginia.edu, son@virginia.edu
Phone: (804)982 2205 Fax: (804)982 2214

(Revised on Dec 15, 1993)

Abstract

The medical school of the University of Virginia is currently developing a comprehensive information system to be employed in the practice of clinical medicine. We have been actively participating in the project, primarily looking into the multimedia issues of the project. We have developed a prototype using QuickTime with Macintosh Quadra 950 as our hardware platform that meets most of the multimedia requirements of the project. We describe the requirements, design, implementation of the multimedia aspects of the information system.We have looked into certain issues like the compression ratio that would be acceptable for echocardiographic applications and the hardware platform which would be critical in the evaluation of the final implementation. We identify and discuss some of the real-time issues which we believe to be mandatory in the design of a multimedia information system and the challenges posed in the system design by the compression of media data.

This work was supported in part by the Health Informatics Enhancement Project of the University of Virginia, Center for Innovative Technology, and IBM.

1. Introduction

The role of computer is changing from a “computational” machine towards an “information processing” machine which supports the processing, communication and presentation of information. Information processing with “text” was complemented by graphics and images and is being enhanced to cover spreadsheets, formulas, audio and video. New storage technologies and architectures allow to store different media in a computer-controllable manner, making a wide variety of multimedia applications feasible [11].

A multimedia system is characterized by the computer controlled generation, manipulation, presentation, storage, and communication of both continuous and discrete media. The flexibility extended by the multimedia systems is hence providing the integration of different media into a single system. Such a system can unify the methods of information distribution, personalize information services through interactive access and individual information selection, enlarge the bandwidth of perception at the user interface, make information presentation more effective and provide flexible media processing and transformation [4].

The video medium though clearly a powerful force in our society, has a number of characteristics which prevent it from reaching its potential for expressive power, creativity and communications, particularly from the perspective of an individual; hence necessisating the need for the development of a multimedia environment. Vital among them are lack of accessible manipulation tools and technologies for an average user, an awkwardness of distribution for an individual and intractable access to a great archive of human knowledge. Interaction and exploration are not feasible, requiring the user to follow a linear path of thought [9].

Video is a medium that cannot be easily annotated, copying material with videotapes in the same or different formats can result in significant quality loss. Archiving is difficult as there is no way to search through a video database. In addition video distribution is “top-down” [12]. In this paper we describe the design of a system that fully exploits the above mentioned features of a multimedia system and serves as the basic infrastructure to support the incorporation and integration of continuous media with discrete media in a medical information system.

The medical school of the University of Virginia is currently involved in the development of a comprehensive information system to be employed in the practice of clinical medicine. We have been working closely with the cardiology department, which has been actively participating in the project.

The proposed model of the medical information system (MIS) is aimed at the automation and integration of the functions of the clinical echocardiography lab and at the same time

creating an information system that is accessible to any user of the medical community. The main components of the MIS are:

- A relational database with a friendly user interface tuned such that the interaction with it is intuitive to a physician.
- A multimedia environment supporting all the elements of clinical data (i.e. both continues and discrete media).
- An expert system capable of translating a clinical question to a report based on the data entered into the database.
- An interactive environment that would facilitate the beginners in echocardiography to learn about the techniques and their clinical role.
- Interface with analysis packages for analysis of clinical data.
- An administrative unit for examining the performance of the laboratory and generating automatic billing information.
- Networking to other information systems.

An abridged view of the relational database design is given in figure 1 and the type of users who can access the systems are depicted in figure 2. The demands of echo cardiography for multimedia support are sufficiently varied and complex such that our understanding and subsequent development of a prototype to fulfill their requirements should serve as a fair measure of our capability to design and incorporate multimedia capabilities in almost any other information system of reasonable magnitude.

A relational database for this information system has been developed at the echocardiography department on a VAX station 3500 (running VMS) using Ask/Ingress. The system is however restricted in its ability to control the continuous media data (audio, video) or graphical data resulting in very limited support of the desired multimedia functionality.

The rest of the paper is organized as follows. Section 2 discusses the multimedia requirements of the project, followed by a brief discussion of the hardware and software used for the development of the prototype in Section 3. Section 4 discusses the design and implementation of the prototype followed by a discussion in Section 5. Section 6 deals with the known limitations and finally we conclude in Section 7 with brief comments about the direction of future work.

2. Requirements

The multimedia support required by MIS is primarily governed from the need of maintaining the following objects in the multimedia database.

(1) CW Doppler and M-mode: each of which is a static monochrome image.

(2) Cine loops 2-D echo: These are 30 frames on average of monochrome image with no associated audio depicting one full cardiac cycle.

(3) Color Doppler: This is the only color data object in the database with the requirement of at least 4 full color frames per view.

(4) Contrast echo: This would consist of two static monochrome images.

(5) Voice annotations: These would be attached to any objects of types (1) - (4) and would be activated by clicking on a symbol displayed with the object. One or more distinct annotations can be attached with the object.

The multimedia support can hence be grouped into the following media types: monochrome video, color video and audio. The process of capturing the objects (1) - (4) which would be from a VCR is to be controlled using the computer interface. This includes the control for recording, play back, viewing through the previous frames stored in memory. Also, options must be provided to grab a sequence of frames from the tape, to select one *cycle* - which refers to an echo cardiac cycle - manually from the grabbed images and to display similar views in compound formats.

The target users of this multimedia system are both the physicians and the students. As would be evident from the discussion of the design and the interface, the turn around time, i.e., the time from capturing of the images by the technicians to the time these images would be presented to the physicians is dependent on the time taken to make a QuickTime movie, which is of the order a few seconds. The turn around time for the student's interface is however a little longer. This is very much in compliance with the needs of the medical community where the physicians are time constrained and the images captured by the technicians are required to be presented to the physicians for their analysis at the earliest; but the data may be presented to the students a little later.

3. Hardware and software platform of the system

The hardware platform we are currently using to develop the prototype of the multimedia MIS is the Macintosh Quadra 950 with 4Mb RAM and a hard disk of over 400 Mb. The software being used is QuickTime^{R} version 1.5 and Hypercard version 2.1. QuickTime provides the capability for combining audio, video, and graphics using Hypercard and has four main components namely system software, file formats, compressors and human interfacing. We provide a brief description of the capabilities of both Hypercard and QuickTime.

QuickTime is an extension of the Macintosh system software that enables the user to integrate time-based data into mainstream Mac applications [1]. The QuickTime architecture primarily comprises of basic components namely the Movie Toolbox and the Image compression manager. The movie toolbox allows the user to store, retrieve and manipulate time based data that is stored in QuickTime movies. The image compression manager comprises of a set of functions that compress and decompress images or sequence of graphic images which are device and driver independent. QuickTime performs most of the desired functions on the movie including displaying, copying and editing the movie. The process of capturing and displaying the movies using QuickTime is illustrated in figure 3.

The management of the acquired multimedia data is done using the Hypercard. Hypercard is primarily a software development package that lets us control the way the text, graphics, audio and video are used on the Macintosh. The Hypercard environment consists of objects knitted together by a message passing hierarchy and the language Hypertalk through which they communicate. To make use of the QuickTime library of software to manipulate the data acquired, we make use of QuickTime's external commands (XCMDs) which are a small set of external commands that allow Hypercard users access to the features of the QuickTime library of software. The four XCMDs provided are the QTMovie XCMD, QTRecordMovie XCMD, QTEditMovie XCMD and QTPict XCMD.

There are two methods by which movies can be displayed and controlled in hypercard:

- Displaying the movies in external windows called XWindows.
- Having the movie played directly on the card window.

The first method allows the user to create a window to which the Hypercard will send the appropriate events as mouse clicks. The movies in the window may be controlled by sending messages to the window or through a graphical controller. The second method of displaying where the control movie is played directly on the card window returns a movie ID when the window is opened and the subsequent commands are sent along with this movie ID.

4. System design

The design of MIS has been split into three phases namely:

- The data acquisition phase and archival phase,
- The data editing phase, and
- The data display and interfacing with the relational database phase.

The data acquisition phase deals with the capturing and storage of the media elements directly from the external source. These media elements also referred to as the primitive data elements are then combined into a compound object using the data editing phase, for example the addition of annotation to a captured video. This combination of the different media elements - primarily audio and video - at the editing phase is required only if the various media elements originate from different sources, for instance video being captured from a video tape and the audio being the annotation of a physician which is captured from a different source. If the different media originate from the same source they are captured together in the data acquisition phase itself, for instance both audio and video being captured from a video tape simultaneously. These compound objects are then finally linked together and controlled using the display phase. Hence, once the data has been collected and edited, the user would always interact with the third phase.

The data acquisition and archival phase has been further split into two modules, for:

- The data input via a video cassette recorder (VCR) and
- The data input via a laser disc player (LDP)

This demarcation arises because the laser disc player can be controlled by the computer whereas the video cassette recorder is not. This is primarily owing to the digital nature of the laser disc player and the analog nature of the video cassette recorder.

4.1. Data acquisition (VCR)

To acquire the data from the VCR the *movie recorder* (a part of the QuickTime package) is invoked. This results in the output of the VCR being displayed in a window on the computer screen. When the movie recorder is invoked, a number of parameters may be altered from the default value. These include the size of the window, the type and quality of compression, option to compress the recorded movie while in RAM or while being stored to the hard disc, the frame rate, the image parameters as brightness, hue control and the audio parameters as the audio level and the quality. Having set the options if desired, the video may be viewed on the screen as it is being played by the VCR. The recording is commenced by clicking the record button present on the screen, clicking the button again signals the end of the recording phase. All the data captured is stored in QuickTime format by default with the file extension “MooV”.

While the recording is being carried out two options exists for storing the captured data:

- the data may directly be stored on the hard disk,
- the data may be held in the RAM till the end of the recording.

The primary advantage of latter is that of the greater frame grabbing rate achieved, as the process of compressing the frames while storing them on the hard disk can be delayed till the end of recording. The limitation however is that the amount of the data that can be captured in the second case is limited by the size of RAM available. Hence in order to capture more data than the available RAM requires that the output from the external source and the recording process be paused periodically (whenever the RAM is full), the data compressed and then transferred from the RAM to the hard disk.

4.2. Data acquisition (Laser Disc Player)

Quicktime offers the ability to control the laser disc player and this facility can be exploited to a large extent to obtain a great addition of functionality. For instance, the process of pausing the external device when the data has to be transferred from the RAM to the hard disk can be programmed in such a manner such that periodically a signal is sent to the laser disk player to pause the output and simultaneously the recording be stopped. The data can then be transferred to the hard disk and the recording process resumed. It may be observed that the ability to control the laser disk player in this example not only makes the pausing operation transparent to the user but also ensures that no frames are lost or blank frames added, unlike the case when the user struggles to control both the VCR and the recording process simultaneously. Since the interface provided by QuickTime with the laser disc player and the VCR are quite different we have provided a separate *stack* - which is a collection of cards, which are generally but not necessarily related to each other and which constitute the basic units on which the Hypercard information appears - to acquire the data from a laser disc player and from a video cassette recorder.

To acquire data from the laser disc player the corresponding stack is invoked. This stack primarily makes use of the QTRecordMovie XCMD.The QTRecordMovie XCMD is used to display a window on the screen in which the video coming from the digitizer can be viewed. The syntax of the command to open a window is:

QTRecordMovie windowName, windowType, windowRect, growable, connectToAudio,-videoStandard, videoFormat.

After the window has been opened, messages may be sent to the window. The messages fall under two categories namely:

- Messages to set the *properties* of the window.

Examples of these include the `frameRate, windowSize, brightness, appendGrab, pictureQuality`. The command for setting the properties is

set <property> of window <window name> to <value>.

Hence the command

```
set frameRate of window “Test Video” to 15
```

would set the rate at which the frames are displayed in the window opened with the name “Live Video” to 15 frames per second.

- Messages to send *commands* to the window.

Examples of this include `DoLiveGrab, GrabOneFrame, AudioOn, EndSingleGrabMovie, LiveGrabPrep`. The command for sending such messages is

send <message> to window <window name>.

For instance to grab a movie that is being displayed in the window, the command `LiveGrabPrep` followed by `DoLiveGrab` is sent to the window. Clicking the mouse again would signal the end of recording. If we desire to grab each frame that is being played from the laser disc the command `GrabOneFrame` and `EndSingleGrabMovie` would be used.

A number of other functions have been provided by this interface. These include the ability to view the incoming data frame by frame, ability to go backward or forward either in a fast motion or by skipping a number of frames, ability to capture one or a multiple sequence of frames, playing a particular segment (specified by the starting and the ending frames) in a loop fashion. The functionalities that have been provided are too many to be described here, so we have restricted the description of those relevant for the MIS project.

One of the desired functionalities of the project is the ability to view the previous or the next cycle in memory without requiring the user to explicitly rewind and then fast forward. This facility can be provided by creating a buffer that stores certain amount of history of the video being played and to display its contents if desired by the user. This method however fails to provide the desired amount of interactiveness due to certain features of QuickTime. The other strategy is to skip back a certain number of frames and then back to the original position when desired by the user. It may be noted that this feature is however not possible if the input is from a VCR.

Having described the data acquisition phases we now give a description of the data editing and the data display and management phases.

4.3. Data editing

The interface concerned with data editing primarily makes use of the QTEditMovie XCMD. The QTEditMovie is used to perform a variety of movie editing functions including the addition of new sound tracks - either by capturing sound from a digitizer, copying from a sound resource or copying from some other source - deletion of tracks, time shifting and the grouping of a number of tracks. For instance to add a new audio track to an

existing movie the `GrabAudioSoon` and the `GrabAudioNow` commands are send as follows:

```
send GrabAudioSoon to window <window Name>
send GrabAudioNow to window <window Name>
```

As another example to view the tracks in a movie the `DisplayTrack`s property may be set to true. The interface of the EditWindow is very much similar to the QTMovie window which we shall describe in Section 4.4.

The data editing phase primarily has two major functions, namely the grouping of a number of audio tracks with the same video track and the addition of certain amount of text along with the video data. Note that after this editing is performed, the added data or the text actually becomes a part of the movie file and is stored together.

For the addition of one or more sound tracks to an existing video file, the two files - one to which the addition has to be done, and the one from which the audio has to be copied - are opened and the process of adding an audio track just involves the selecting of the file from which the audio has to be copied and then selecting the file to which the audio has to be added. It is left to our discretion whether we wish to add a new audio track or to extend the existing audio track with the new audio data. Viewing this edited file and selecting the audio to be played is extremely simple - merely involving the selection of the appropriate audio track which are displayed on the screen. This functionality provides the ability for different users to view the same video data but with different audio data. Note that this is highly efficient in terms of the storage requirement considering the fact that video data requires a tremendous amount of space and by providing multiple audio tracks with the same video track we are avoiding the replication of the video data with each associated audio data.

The other functionality of the addition of text to the video image is of great significance - this in appearance is very similar to subtitles in a movie. The importance of providing the facility to display or view text while playing a movie file may not be quite intuitive. However, the fact that the viewer of the movie has to be provided with some indication to the effect that certain text, audio, video or graphic data is associated with the segment of video which is currently being displayed and may be viewed if desired, requires a mechanism by which this indication may be done. This added text may be made to overlap on the image or may be placed just adjacent to the image being displayed. A provision is also present by which the user may determine if any such text is present in a file and if so the position in the file, by simply inspecting the file. This may be of considerable use while dealing with large files when the user wishes to directly access the points in the file where some extra information has been associated with the video data, rather than browsing through the entire file.

Some additional features are also present for example setting a default frame which should be displayed each time the file is opened; but since these are not very crucial functionalities we refrain from discussing further. This editing interface though complete in terms of functionality has to be refined to make it more transparent and friendly to the user.

4.4. Data display

The data display phase makes use of the third type of XCMD namely the QTMovie. A movie can be opened by issuing the following command:

QTMovie, OpenMovie, windowType, <filename>, location [optional parameters]

The windowType basically refers to whether we intend to play the movie directly on the card window - as discussed earlier in section 3.0 - which is specified with the keyword `Direct` or have a separate Xwindow which is specified by an integer corresponding to a window definition ID. Similar to the commands of QTRecordMovie, in order to send a command the syntax is:

send <command> to window <window name>

Few of the commands that can be send to the window include `Play, Pause, GoNextKeyFrame, StepFwd, CopyFrame`.

4.5. Display interface

Our primary objective after having acquired and edited the various data - both continuous and discrete media - is to be able to view the data in a *controlled* fashion and to provide the user the ability to navigate through the data in a very flexible and easy manner. By viewing the data in a controlled fashion we basically refer to the ability to not only “play” the movie file but to also pause, scan forward or backward, view the file in small increments - ideally each frame, resizing the viewing window and some other functionalities.

To provide a feel for the application of the above said navigation consider the case of a physician viewing an image file. At a particular image the physician may wish to refer to certain information concerning that patient. This information may be a part of the routine information that is stored for all the patients or may be particular to the patient. If particular to the patient, the information whether such additional information is present would appear as text on the image - as discussed in Section 4.3.

Note the difference between the textual information associated with an image and the text that appears on the image. The text on the image provides the information to the user that

some information be it visual, audio or textual is associated with the image and may be accessed by going to the appropriate interface. Capability to access this additional information, whether it be routine or particular to the patient, along with the current image being displayed should be provided. Example of the stored textual information may be an article regarding a particular symptom that was encountered in another patient or cited in the literature which is stored in an archival text file. Audio information may include annotations or comments regarding a particular image by the physicians, for a file to be viewed by the students. The viewer should have the option to listen to that audio file or if desired read the text file or view some other image - probably taken from a different angle, if available. This is a part of the desired functionality of interactive learning. It should also be possible to view a number of different images or different instances of the same image simultaneously on the screen.

5. Discussion

Our prototype provides the above mentioned functionalities. The interface is composed of six stacks namely audio, visual, graphics, textual, main menu and the additional info stack. Invoking the display interface leads to the main menu shown in figure 4 where the file to be viewed for a particular patient may be selected. Present in the window are various menus specifying the options to view the file in loop or a palindrome fashion, control the rate of display of frames, the level of audio, the position on the screen where the file should be opened to view etc. When the user wishes to access the additional audio, visual, textual or graphical information (routine or particular to the patient), he may select the appropriate icon on the main menu, which invokes the stack of the corresponding media of the patient. The movie file that was being displayed is not closed - though can be if desired - in this process. The new stack (interface) consists of icons of the file that are present for the patient. Choosing any of the icon leads to the display of that file. If the user wishes to access an archive file or a file of a different patient he may enter the file name by choosing the icon “Enter file”. As pointed out earlier multiple number of files may be displayed in any interface (as shown in figure 5). An obvious advantage of this is the ability of the physician two compare two or more images (still or motion). This particularly may be the case when it is desired to observe the effect of some treatment on a certain patient or to compare a patients case with another similar case.

From the design of the system we observe that our system, provides a large number of additional features for viewing a file, in addition to providing all the features that a VCR provides for the display of a image (movie) - play, stop, pause, fast forward. These include provision for viewing the file frame by frame, playing a selected segment over and over again, ability to determine where in a file certain text (used to communicate the presence of additional information) has been added without *playing* the file (figure 6) and other features.

The developed prototype provides the environment both for the physicians and the students. The physician would chiefly interact with the main menu and the only requirement for that is creating of QuickTime movies from the data stored on the VCR tape. This process takes only a few seconds (usually less than 10) and is hence very much suited considering the maximum delay that may be permissible from the time data is collected from the patient to the time it is presented to the physician.

An issue that can be looked into is the capturing of QuickTime movie directly from the probe used to capture images of from the patient, instead of storing the image first on a VCR tape. The advantage of this is twofold. Firstly, the reduction in the delay from the time the patient is tested to the time the physician has the images available to see, and secondly, a probable improvement in the quality of the images captured.

The present interface of the editing phase, though provides us with the desired functionalities, requires the user to have more knowledge than the minimum that could be needed to perform the desired function. By the nature of the medical application we observe that the time required to work and understand the interface, coupled with the user friendliness would play a major role in determining the magnitude of the acceptance of the system, hence we plan to further enhance the editing interface. It may however be noted that since the editing phase is intended to be operated by the non professional staff, the time factor is not so critical. The interface provided for the physicians that is the data display phase has been tuned to be very user friendly and hardly requires any learning process.

As has been widely discussed in the literature, compression is essential if audio, video, and images are to be used in digital multimedia applications [6]. A megabyte of space is filled with roughly 6 seconds of CD quality audio, a single 640 X 480 pixel color image, or a single frame of CIF video. In general, compression can be classified into lossless or lossy compression. In the lossless compression or entropy encoding as it is sometimes referred to, the original representation can be perfectly recovered. However the compression ratios are low. A figure of 15:1 is considered to be a good one. A lossy or noisy compression may add artifacts that may be perceived [7]. However, very high compression ratios are attainable. To obtain a feedback of the compression ratios that would be acceptable in medical application - it may be noted that echo cardiography is one of the most demanding areas in medicine in terms of the required image quality - we have conducted a series of experiments using the JPEG compression provided by QuickTime. The determination of the compression ratio is complicated by the fact that a number of parameters play a vital role in determining the acceptance of a compressed image. Ponderous among them being:

- the experience of the physician
- the quality of the image which varies dramatically from patient to patient

- the instrument used to capture the images from the patient, which largely influences the quality of image.

To arrive at a single compression figure which embraces all of these factors requires that a large collection of images be taken from varied sources. Around 400 patient data images were taken and were subject to various levels of JPEG compression. The maximum compression that was achieved was found to be dependent on the machine that was used to capture the patient data. For instance the maximum compression recorded for images captured by HP machine was 19.72 : 1 while the maximum compression recorded for images captured by Acuson machine was 15.83 : 1. The compressed images were inspected by physicians and statistics from their feedback obtained. It was observed that the inter-viewer agreement was very high for less compressed images and increased with the increase in the compression level. In addition to the compression of the images, the images were also clipped and the color levels decreased in a manner such that no clinical information was lost. This resulted in a further saving of memory by more than an order of magnitude. The maximum saving of memory that was recorded (considering all the factors mentioned above) was 210.33.

5.1 Real-Time Considerations

The role of real-time in multimedia applications though inevitable [14], depends on the application under consideration. Medical applications fall under the purview of applications which are the most demanding in terms of time criticality and other real-time issues. The correctness of the system hence depends not only on the logical consistency of the data but also on the fact that the deadlines of the tasks be met regularly.

Real-time issues which play a very important role in monolithic systems become even more critical in a distributed environment [14]. Examples where the need for real-time considerations is required in a monolithic environment include storage and retrieval of data, scheduling of processes, sharing of available bandwidth or other resources. In a distributed environment we observe the need for the operating system to cope up with multiple real-time network connections, the need for real-time protocols to control the jitter and delay bound on the communication channel and the need for inter-operability of a system with other real-time servers across the network. As is evident from the discussion above, the real-time issues are of great importance whenever the system is required to deal with continuous media, the multimedia system make this role even greater and perhaps more stringent.

Based on our experience in the development of the multimedia information system and the requirements of the medical school, we have been able to identify a number of user level functionalities that can be quite easily incorporated if the multimedia platform is designed keeping in mind the real-time mechanisms [14]. For instance a multimedia system derives

its great strength from the ability to simultaneously display multiple images. Hence it would be very beneficial to provide the user the ability to designate a image as the “master” image which should be refreshed at a regular rate or given a priority over other images if there is a contention for resources among the images. Another functionality that is also related to the notion of quality of service is to provide the user the ability to predict if an image can be rendered at a specific frame rate or if particular frame in an image would always be displayed (observe that in event of unavailability of adequate resources, frames from an image may be dropped to maintain synchrony among media streams). In that scenario, the image in discussion would always be assigned the necessary bandwidth and other resources to meet the requirement and only if there are some resources left over will the other images be assigned the resources.

Compression of media data, as has been discussed can lead to tremendous amount of storage savings and is inevitable in future multimedia storage systems. However dealing with compressed data would mandate the need for some “adaptive” real-time mechanisms. We observe that the magnitude of compression of an image depends to a very large extent on the image itself. This as we have found in our study too can vary very significantly. This difference in compression levels means a difference in the amount of data generated and hence a difference in the time taken to retrieve, send, store and display the data. And hence it is conceivable that a particular image be rendered at the display site in accordance with the specification while another image may fail to comply with the specifications under the same conditions. And this calls for some kind of an adaptive real-time mechanism to be incorporated in the multimedia systems.

6. Known Limitations

One of the issues that we have briefly addressed earlier also is that of the interface with the VCR. From the experiences with Hypercard, we do not envision any mechanism by which we could probably control the VCR from the computer. This restriction probably is owing to the analog nature of the VCR. If an interface could be provided as with the Laser Disc Player almost all the functionalities that have been incorporated in the acquisition phase of the Laser Disc player could probably be incorporated into the VCRs acquisition phase. However, certain issues such as dealing with the tracks on the laser disc would have to be dealt with differently. For instance instead of specifying the number of tracks to rewind, we would probably specify it in terms of certain time units.

The issue of bandwidth has posed certain restrictions. Firstly the number of images that can be simultaneously displayed on the screen is highly dependent on the bandwidth. Opening too many windows makes the motion of the movies too erratic. Also, the number of frames per second that are displayed is dependent on the bandwidth. As a result, a window that is smaller in size displays more frames per second compared to a larger window.

Finally, QuickTime is a centralized authoring environment, a single application providing built in support for a number of media. As has been pointed out elsewhere in the literature [5], a centralized authoring environment provides built in support for a limited set of media that have been designed specifically for that purpose. This has the demerit that the support for one medium is generally weaker than an application solely devoted to that medium. Also, these environments are not flexible enough to accommodate new authorship models. The obvious solution to this issue is to consider multimedia environments that are extensible and can accommodate diverse styles of authorship. The issue that has to be however kept in mind is that not too many such softwares are yet commercially available to choose from.

7. Conclusion

We have designed and implemented a multimedia prototype system on the Quadra 950 using QuickTime ver 1.5 which is to serve as the basis for the design of the final version of the multimedia system which would be an integral part of the MIS. The prototype developed fulfills most of the desired requirements of the MIS project and has provided insight to a number of new applications for which the multimedia system can be used. We have studied the impact of compression on image quality to determine the compression ratio that would be acceptable for echocardiographic applications. We have also identified a number of functionalities that should be incorporated by a multimedia system which require the incorporation of real-time mechanisms in the multimedia platform. Finally we discuss the challenges posed by the compression of media data and the incorporation of some adaptive real-time mechanism to combat the same.

One of the issues that would be vital for determining the success of the project in real world application is related to networking. For instance in the medical school, once the images have been grabbed we would like them to be stored in a central or distributed system such that each user could access the images without having to have a copy of those images in his hard disk. This brings in all the advantages that go along with a networked environment, particularly easy access and data sharing.

Even though the present prototype has been developed on the Macintosh, we plan to implement the final version on a different platform. The primary reason for this being that the database for the MIS which is in a very advanced stage of development has been designed using Ingress, and hence we plan to implement the multimedia system on a platform that supports Ingress. For this reason, we have been looking into a number of platforms that have commercially available multimedia software. This exploration is however is its nascent stage and we are looking into the capabilities of several softwares, including MAEstro that has been developed at Stanford University & Sun Microsystems and runs on Sun and NeXT workstations [5].

References:

[1] QuickTime Components 1.5 Developers kit, Developer Technical Publications, Apple Computer, Inc. 1992.

[2] QuickTime 1.5 Developer kit, Developer Technical Publications, Apple Computer, Inc. 1992.

[3] The QuickTime XCMDs, Ken Doyle, QuickTime Software Group, Apple Computer, 1992.

[4] R G Herrtwich, R Steinmetz, Towards Integrated Multimedia Systems:Why and How, IBM ENC Tech Report # 43.9101.

[5] G D Drapeau at al., MAEstro - A Distributed Authoring Environment, USENIX conference, Summer 1991.

[6] E A Fox, Advances in Interactive Digital Multimedia System, Computer Communications October 1991, Pages 9-21.

[7] H Ramapriyan, Proc Scientific Data Compression Workshop, NASA Conf Pub 3025.

[8] Sanjiv Kaul, et al : Proposal to the National Institute of Health, University of Virginia, March 1993.

[9] Michael Liebhold, Eric M. Hoffert, Towards an open Environment for digital video, Communications of the ACM, Volume 34, No.4, April 1991, Pages 104-112.

[10] John A. Ada, Applications, implication ; IEEE Spectrum March 1993, Pages 24-31.

[11] Ralf Steinmetz et al, Generic Support for Distributed Multimedia Applications, International Conference on Communication, April 16-19 , 1990.

[12] Gilder G, Life After Television, Whitle Direct Books, Knoxville, Tennesse 1990.

[13] Bernard J. Haan et al, IRIS Hypermedia Services, Communications of the ACM, Volume 35, No. 1, January 1992, Pages 36-51.

[14] Sang H. Son and Nipun Agarwal, Real-Time and Synchronization Issues in Multimedia Computing Systems, IEEE Workshop on the role of real-time in multimedia/intercative systems, Raleigh, N.C., Nov 30, 1993.

[15] Nipun Agarwal, Sang H. Son et. al., Model Multimedia Workstation for Echocardiography, 17th Annual Symposium on Computer Applications in Medical care, Washington D.C., Oct 30 - Nov 3 1993.

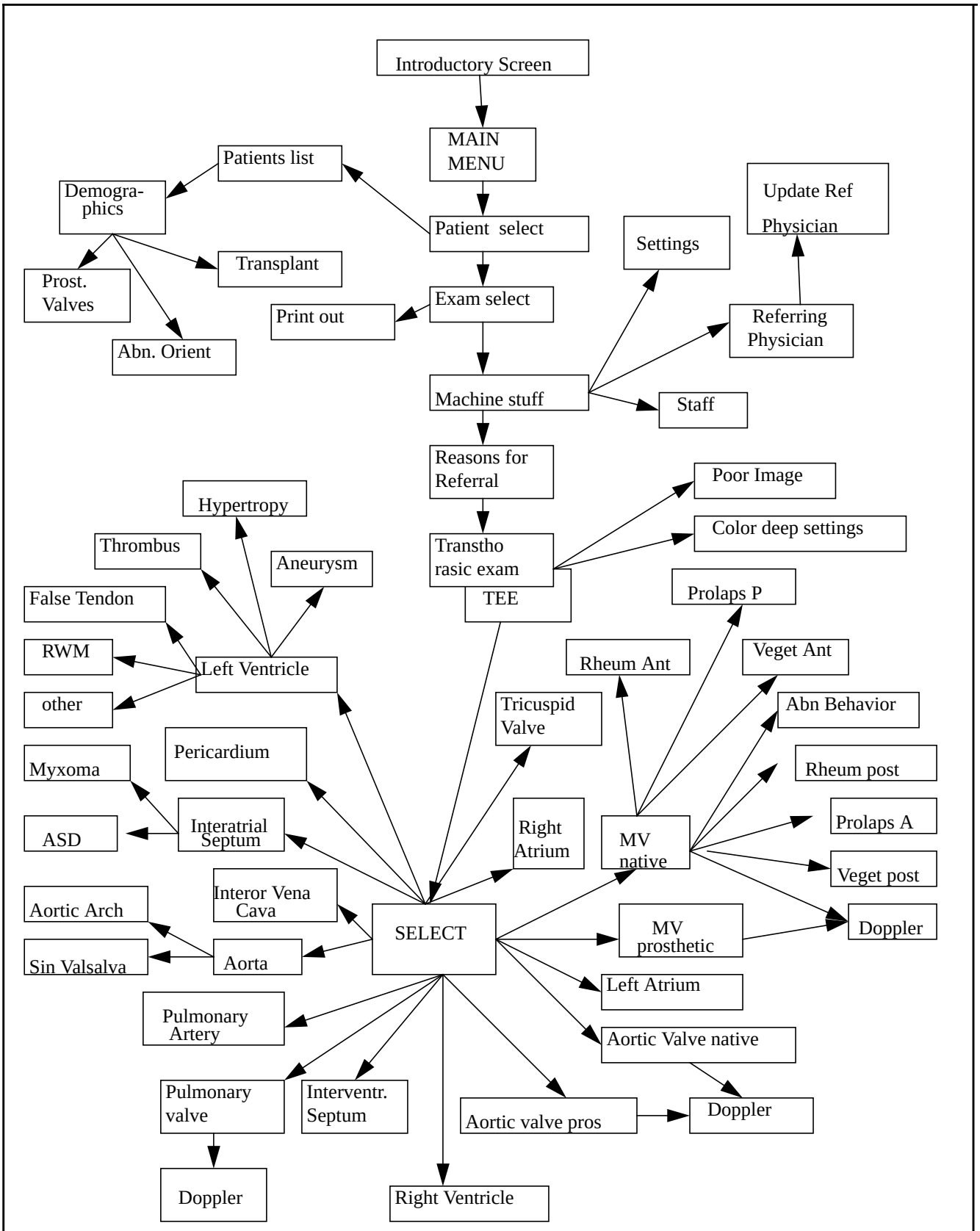

Fig. 1 Abridged view o f the multi-layered relational database design

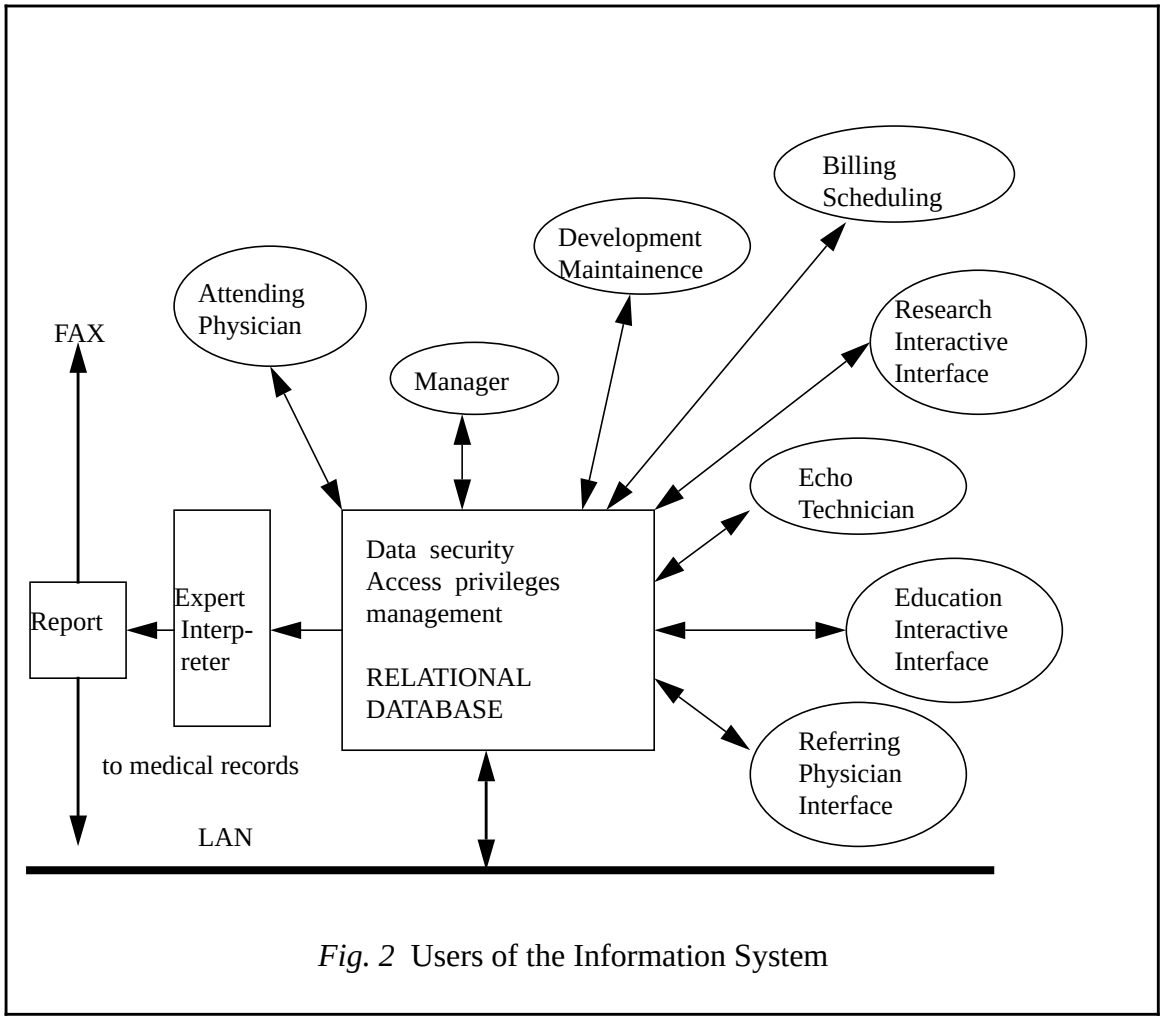

Fig. 2 Users of the Information System

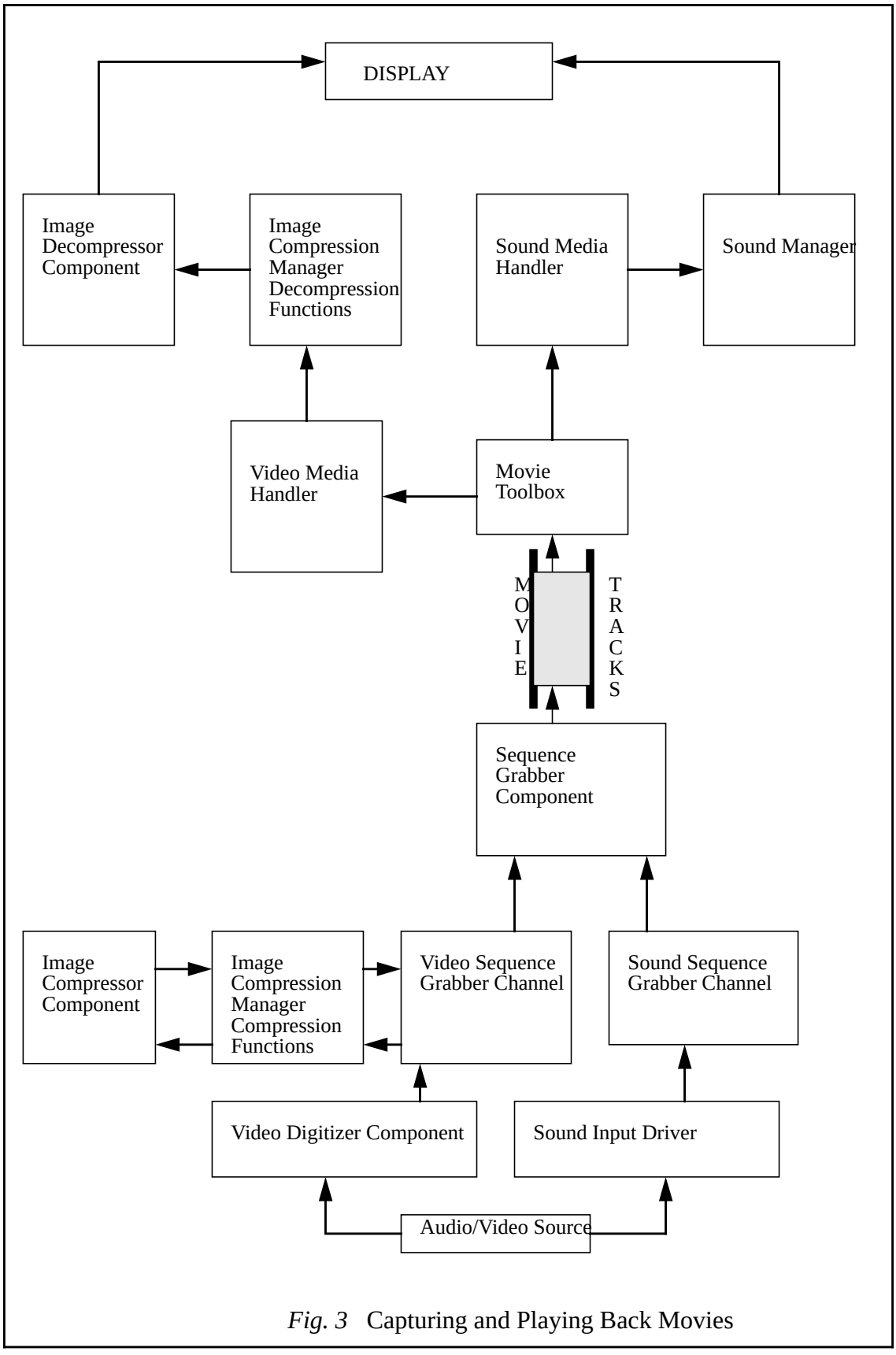

Fig. 3 Capturing and Playing Back Movies

Figures 4, 5, 6 and 7 have not been included due to the large space requirements. They may however be obtained from the authors.