A Hierarchical Approach for Content-Based Echocardiogram Video Indexing and Retrieval

Aditi Roy
School of Information Technology
IIT, Kharagpur, West Bengal, India
aditi.roy@sit.iitkgp.ernet.in

Shamik Sural
School of Information Technology
IIT, Kharagpur, West Bengal, India
shamik@cse.iitkgp.ernet.in

Jayanta Mukherjee
Dept. of Computer Science & Engg.
IIT, Kharagpur, West Bengal, India
jay@cse.iitkgp.ernet.in

A. K. Majumdar
Dept. of Computer Science & Engg.
IIT, Kharagpur, West Bengal, India
akmj@cse.iitkgp.ernet.in

ABSTRACT

Content-based video retrieval is one of the most challenging areas of research in information sciences. Here we present a new method for efficient indexing, storage and retrieval of echocardiogram video data. The architecture and implementation details of the system have been discussed. A hierarchical state based indexing scheme has been used for faster retrieval of information related to states of objects present in echo video. In video segmentation phase, views as well as states and sub-states of objects in a video are identified. Accordingly, schemas are populated. An object-relational database approach has been adopted to store object information into a backend RDBMS. The proposed video query language can be used to compose queries based on video contents using annotations, states, events, spatial or temporal relations. For ease of specifying query, a user friendly interface has been provided. Retrieved segments can be played back one at a time in a separate window.

Categories and Subject Descriptors

I.4.8 [**Image Processing and Computer Vision**]: Scene Analysis; I.4.9 [**Image Processing and Computer Vision**]: Applications.

General Terms

Algorithms.

Keywords

Echocardiogram video, Query language, Video retrieval.

1. INTRODUCTION

Cardiovascular disease is a frequent natural cause of death throughout the world. Total number of deaths due to cardiovascular diseases reach 17.5 million a year according to WHO report, "Preventing chronic diseases: a vital investment (2005)" (Source: WHO World Health Report 2004). It shows the importance of echocardiogram video which is mainly used for monitoring heart status. It is captured in the form of a video. With the rapid progress of technology, echocardiogram machines are now equipped with a number of built-in tools for better visual analysis of echocardiogram images. Several modern methodologies, such as estimation of volumes of cavities, estimation of wall and valve movements, delineation of cavities, measurement of blood flow, etc., are supported by these machines. However, acquired data are not preserved in digital format in the echocardiogram machines. The limitations associated with echocardiogram video storage are as follows [1].

- Searching a particular echo study of a patient from a large archive of videotapes is time consuming. The problem gets worse when several studies are stored on a single videotape.
- A full echocardiography study of one patient, which includes different modes and angle positions of transducer, could take up to one hour. Such a long time may not be affordable by the doctor.

These data can be useful for analysis of other features of echocardiogram data for better understanding of cardiovascular system, even though it may not be required for a particular patient. There is also a need for automated analysis of these videos. Medical researchers often need to retrieve specific frame/ segment of a particular patient or group of patients for analyzing various characteristics. This motivated us to design a framework for managing echo video based on their content. Our goal is to support an efficient and easy way for users to index, store and retrieve video data from database. Efficient storage and support for content-based retrieval of the echo video data have numerous applications like Hospital management systems, Telemedicine, Continuing medical education, Distance learning, etc.

The rest of the paper is organized as follows. In Section 2, we present a brief discussion about existing work and the user requirements that led us to design this method. Section 3 presents the system in detail. In Section 4, current status of the system implementation is described. Section 5 concludes the paper.

2. BACKGROUND

For successful content-based video retrieval (CBVR), there are two main factors. In the first stage, effective algorithms need to be developed to segment a raw video into meaningful components and to extract and index features describing the content of the video segments. The second issue is development of an effective and powerful query language for retrieving specific segments.

In recent past, advances have been made in content-based retrieval of medical images. Researches on echo video summarization, temporal segmentation for interpretation, storage and content-based retrieval of echo video based on views have been reported [2] [3]. But these methods are heavily dependent on the available domain knowledge, like, spatial structure of the echo video frames in terms of the 'Region of Interest' (ROI). Another approach for view classification is presented in [4]. But this approach is applicable for classifying apical two chamber and four chamber views only. On the other hand, an approach towards semantic content-based retrieval of video data using object state transition model has been put forward in [5]. In their work, echo videos are segmented based on states of the heart object. Ebadollahi et al. [6] have presented an approach for modeling the activity pattern of objects represented by a constellation of their parts.

One of the earliest approaches towards development of video database system is due to Oomoto and Tanaka [7]. They present an SQL-like query language called VideoSQL for retrieval of video objects by specifying some of the attribute values. Chu et al. [8] developed a medical multimedia distributed database system to support query by both image as well as alphanumeric content. User can formulate queries using conceptual and imprecise medical terms. But, this system does not support video querying. In contrast to this, in [9][10], a video query language is proposed that allows modeling of semantic content of video data as well as spatial properties of objects.

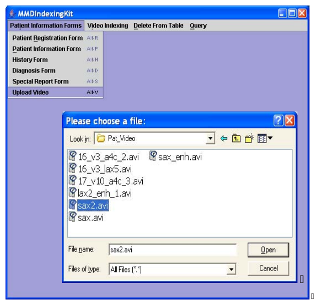
Chang et al. [11] introduced an object-oriented content-based video search engine, called VideoQ. It provides two methods for searching video clips. One is keyword-based and the other is based on visual content. Kuo and Chen [12] proposed a content-based video query language (CVQL) in which objects appearing in video data, as well as temporal and spatial relationship among objects, can be specified. But, it is not easy for user to specify queries relating events. Petkovic and Jonker [13] proposed COBRA (Content-Based RetrievAl) query language for bridging the gap between feature based model and semantic model. Hacid et al. [14] developed a rule-based constraint query language for querying both semantic and video image features like color, shape and texture. A more generic rule based video query language was proposed by Donderler et al. [15]. As specification of spatial queries using text or visual interfaces is not easy for novice users, this group has developed a natural language based interface for querying video database [16]. However, it is assumed that the video events, objects and the related properties are all available, which may not be true in the real world. Acharya et al. [17] proposed a video model for representing dynamic behaviors of objects. They reported an algebraic structure for query specification. Other video models and query languages have been proposed in [18, 19].

Most of the languages described above provide a fixed set of queries. Writing queries in database query language by non- IT specialists like doctors and medical researchers often turns out to be difficult. Keeping these facts in mind, we propose a video query language (VQL) suitable for retrieving video segments describing various states of the heart from dynamic behavior of objects present in an echocardiogram video. The language has been developed on top of an underlying state based video model [20]. A set of functions for specifying temporal, spatial as well as state-based queries is defined. In contrast to the work reported in [19], these functions facilitate query specification by non-expert users who may not be equipped to write queries in database query language like SQL. Thus, the proposed language will help doctors and medical research personnel to frame their queries and retrieve useful information from echocardiogram video database.

3. ARCHITECTURE AND DETAILED SYSTEM DESCRIPTION

The proposed system has an architecture that can be subdivided into two functional units - the first one is for database population and the second for database querying. Video indexing is done in two ways. View based indexing is done on the basis of the occurrence of objects and their orientation. State based indexing is done on the basis of the attribute of objects and their dynamic behavior. First, the echo videos are split into view segments by a view detector. The view detector basically segments the videos based on the different views. Then for each view, the state extractor identifies the states. Finally, sub-state extractor detects the sub-states. Shot, state and sub-state information is stored in a database hierarchically. When a query is put to the system, the query processor module interprets it and retrieves the video segments based on its semantic content. This technique leads to a simple and fast retrieval of the desired video segments. In the next sub-sections we describe the details of the system.

3.1 Uploading an Echocardiogram Video



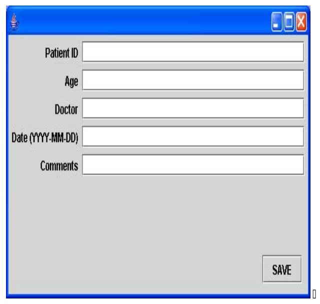
Look in: Pat_Video

16_v3_a4c_2.avi
sax_enh.avi
16_v3_lax5.avi
17_v10_a4c_3.avi
lax2_enh_1.avi
sax2.avi
sax.avi

File name: sax2.avi

Files of type: All Files (*.*)

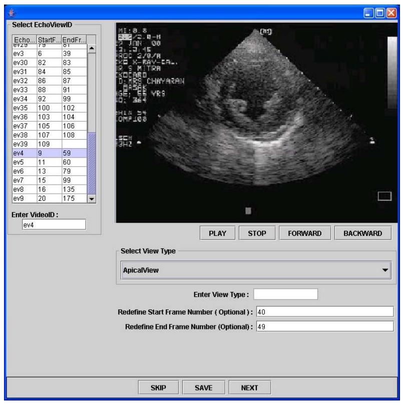
(a)



(b)

Figure 1: Interface for (a) uploading an echo video (b) patient information corresponding to the uploaded video

For uploading an echo video, user has to select the option "Upload Video" in the graphical interface shown in Figure 1(a). Then, a new selection wizard appears where he has to browse the location of echo video to be stored in the echo video database. After the selection is over, user has to put the information of the patient corresponding to that video


Select EchoViewID

Echo...	StartF...	EndFr...
ev29	79	81
ev3	6	39
ev30	82	83
ev31	84	85
ev32	86	87
ev33	88	91
ev34	92	99
ev35	100	102
ev36	103	104
ev37	105	106
ev38	107	108
ev39	109	
ev4	9	59
ev5	11	60
ev6	13	79
ev7	15	99
ev8	16	135
ev9	20	175

Enter VideoID : ev4

Select View Type: ApicalView

Enter View Type :

Redefine Start Frame Number (Optional) : 40

Redefine End Frame Number (Optional) : 49

Figure 2: Interface for displaying the result of view boundary detection and modification

in the text fields shown in Figure 1(b). When saving this information in the database, the system assigns a unique video ID to each echo video.

3.2 Echocardiogram Video Data Loading

Video segmentation is the most time consuming step of video database creation. Once an echo video is uploaded, the next step is to segment it into different views. More than one echo video can be recorded for the same patient at different times. User has to specify the video file he wants to segment.

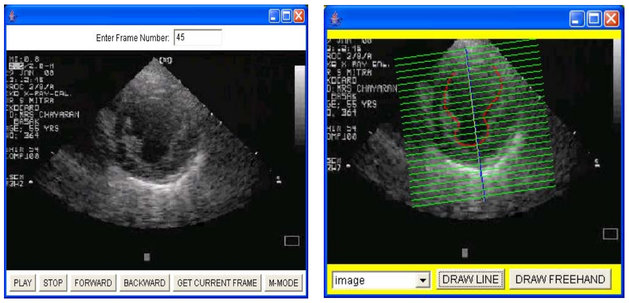
3.2.1 *View-based Segmentation*

After selecting the echo video, a new window appears for view detection. Here user needs to choose the shot detection procedure for starting view boundary detection After detecting views, a novel technique is applied for automatic view classification using some of the echocardiogram signal properties and their statistical variations. The fact that, for each view, different sets of cardiac chambers are visible, is used. The number of chambers present, their orientation and the presence of heart muscles in each view, give different patterns of histogram. Views are identified based on their unique histogram patterns. At first, some of the frames from the video are selected randomly. Then we extract the ROI (Region of Interest) from those frames to minimize the effect of noise. Next, normalized gray scale histograms are generated using 64 bins from the ROI image segment. This is treated as the 64-element input vector of a neural network. The neural network used here has one input layer (having 64 units), one hidden layer (having 80 units) and one output layer (having 4 units, one for each view). Once view detection and identification are done, the result is displayed in a different window (see Figure 2). User can play the video segments by selecting them one by one. Thus, he can verify if the boundaries are detected correctly or not. User has option to annotate the views manually from a drop-down list showing the valid views in an echo video (if the classification done by the classifier is not correct). Text fields are provided to redefine the view boundaries if they are not properly segmented.

3.2.2 *State-based Segmentation*

Once the video file has been selected, the next step is to detect the state segments for each of the views present in that specified video. Object states are detected with the help of synthetic M-mode images. In contrast to traditional single M-mode, a novel approach named as '*Sweep M-mode*' is used for state detection. For this, a graphical interface as shown in Figure 3(a) will appear. User can play the video file for obtaining primary information about how the states change in the echo video. Then, he has to enter any chosen frame number. Figure 3(b) shows a window displaying the selected frame that appears after entering the frame number. Next, user has to identify the cavity border freehand and draw one single straight line as shown in Figure 3(b). The system automatically generates a number of lines perpendicular to the user drawn line. The sweep M-modes will be generated along these horizontal broken straight lines. To compute M-mode image, this line is scanned for each frame of the video. The intensity value along the straight line is taken as ordinate and frame number is taken as abscissa.

Most echocardiograms have a relatively high noise level because of intrinsic limitation in the measurement device. Substantial noise reduction with minimal information loss is achieved by smoothing the image. First, LSMV filter is used to remove noise from M-mode images and then Sobel operator is applied for edge detection. Next, the cardiac borders are extracted from the M-modes by searching for optimal path along the time axis. For extracting state information from each M-mode, first the distance between the two endocardium borders is computed. This distance information is used to classify the echo video frames into two classes, namely, systole and diastole.

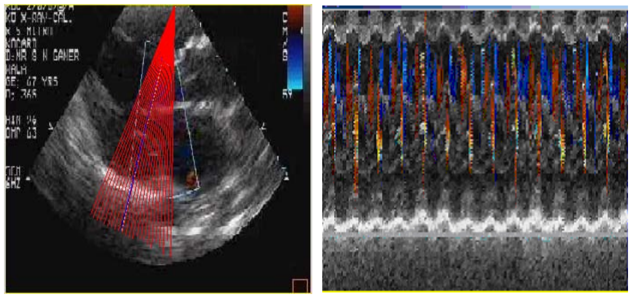

Enter Frame Number: 45

(a) (b)

Figure 3: (a) Interface for state detection (b) Interface for drawing lines to generate M-modes on a chosen frame of echo video

3.2.3 *Sub-state based segmentation*

Sub-states are extracted from radial color M-modes. Here, user draws one single straight line. Then, a number of straight lines are generated with the same origin but different slopes as shown in Figure 4(a). Color M-modes produced from radial M-mode contain red and blue color bars (see Figure 4(b)). The width of red color bar represents diastole state and the width of blue color bar represents systole state of the left ventricle. Color transition from red to blue bar indicates end of diastole and start of systole. Similarly, end of diastole and start of systole is indicated by color transition from blue to red. When blood flows from one side to another in the left ventricular cavity, there is a delay in color transition as we move from the aortic valve tip to mitral valve tip. This observation is used to identify sub-states



(a) (b)

Figure 4: (a) User drawn straight line on a chosen frame of video, (b) Corresponding color M-mode image

from the normal echocardiogram video. In absence of clear color transition, as expected in abnormal case, our method will not detect any state transitions. Thus, this method can also be used to identify the echo videos portraying an abnormal heart, such as mitral valve regurgitation.

3.3 Database Querying using Video Query Language

For successful content-based video retrieval, a query language that can formalize different kinds of queries based on users' need is important. We propose a new video query language (VQL) which is similar to SQL in structure. User can specify queries based on video contents such as annotations, structures, events, states, spatial relations, and temporal relations using this video query language.

The syntax of the proposed video query language is as follows.

select $< VideoSegmentVariable >, [Variable]$
from $< VideoDatabaseName >$
where $< Condition >$

The **select** clause specifies the result that user wants to retrieve. A query returns video segments (Video Segment Variable Name) with or without values of variables (Variable). Aggregate functions (sum, count, average) operating on video segments can also be used in the select clause. The database name on which user wants to execute the query is specified in the **from** clause. The **where** clause is used to specify query conditions. Conditions can consist of attribute / value pairs, comparison operators, video or aggregate functions. The video frame number also can be used in the conditions.

Operators: Our video query language supports a set of logical operators, namely 'and', 'or' and 'not'. For assignment and comparison, "=", "$<$", "$>$", "$\leq$", "$\geq$" and "!=" are used.

Video Functions: The functions that are supported can be classified into three categories, namely positional, temporal and state-based.

1. ***Spatial/ Positional Functions:*** The positional functions are Left, Right, Above, Below, LeftAbove, LeftBelow, RightAbove and RightBelow. These functions take two object variables as arguments and examine the relative position of the objects present in a video segment. For example, Left(o1, o2) describes whether o2 stays on the left of o1 in a video segment.

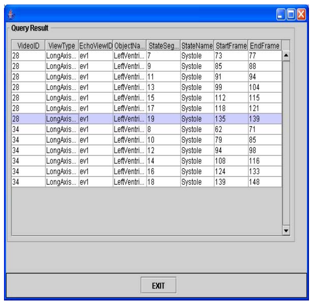
2. ***Temporal Functions:*** The temporal functions are as follows:

   - **Follows(k1, k2):** It returns the video segments (k1) which follow video segment k2. k1 follows k2 if k1.StartFrame >= k2.EndFrame. The complementary function FollowedBy(k1, k2) is also similarly defined.
   - **Contains(k1, k2):** It returns a video segment k2 for which k1.StartFrame $<= k2.StartFrame <= k2.EndFrame <=$ k1.EndFrame. ContainedIn(k1, k2) function is defined as the complementary function of Contains(k1, k2).

3. ***State-based Functions:*** For convenience of specifying complex queries, a number of state-based functions are developed. They can be used to evaluate some frequently needed guard conditions that can also be formed using general language syntax. Thus, complex queries can be expressed in simpler forms and non-expert users can use this language easily to express queries. The functions are discussed below.

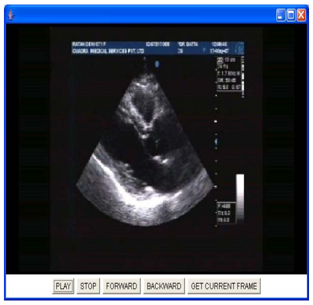
   - **Occurs(o.s, k):** examines whether the object 'o' in state/ sub-state 's' is present in video segment 'k'.
   - **OccursAfter(o.s, k, n):** determines the occurrence of the state/ sub-state 's' of the object 'o' in the video segment 'k' after the n^{th} frame of the video segment. OccursBefore(o.s, k, n) function is developed similarly as complement of OccursAfter.
   - **OccursFirst(o.s, k, n):** returns the first occurrence of the state/ sub-state 's' of object 'o' in video segment 'k' after n^{th} frame of the segment.
   - **ithOccurrenceOf(o.s, k, n):** returns the video segment which is the i^{th} occurrence of state/ sub-state 's' of an object 'o' within the video segment 'k'.
   - **Until(o.s, k, n):** returns all the frames that occur in a video segment before the n^{th} occurrence of the state/ sub-state 's' of object 'o'.
   - **Since(o.s, k, n):** returns all the frames that appear after the n^{th} occurrence of state/ sub-state 's' of object 'o'.
   - **Reachable(o1.s1, o2.s2):** examines whether the video segment containing state 's2' of the object 'o2' is reachable from the video segment containing state 's1' of the object 'o1' as a result of any number of valid transitions.
   - **Projection(k1, k2):** represents the projection operation on video segments. K2 is projected from k1 if k1.VideoID = k2.VideoID and k2.StartFrame >= k1.Startframe and k2.EndFrame <=k1.EndFrame.
   - **SegmentOccurence(O.s, k1, k2):** determines that there is a video segment k2 in video k1 and there is an occurrence of the state s of an object O in the video segment k2. Similarly, FirstSegmentOccurence(O.s, k1, k2), NthSegmentOccurence (O.s, k1, k2, n) is also developed.

Aggregate Functions: Our video query language supports three types of aggregate functions, namely, count, sum and average. These functions take a set of segments (intervals) as input. Count returns the total number of occurrence of the specified segments. Sum gives the total number of frames of all the video segments. Average computes the average number of frame count of all video segments taken as input. These functions are useful in getting statistical data about any patient in an echocardiogram video retrieval application.


Query Result

VideoID	ViewType	EchoViewID	ObjectNa...	StateSeg...	StateName	StartFrame	EndFrame
28	LongAxis...	ev1	LeftVentri...	7	Systole	73	77
28	LongAxis...	ev1	LeftVentri...	9	Systole	85	88
28	LongAxis...	ev1	LeftVentri...	11	Systole	91	94
28	LongAxis...	ev1	LeftVentri...	13	Systole	99	104
28	LongAxis...	ev1	LeftVentri...	15	Systole	112	115
28	LongAxis...	ev1	LeftVentri...	17	Systole	118	121
28	LongAxis...	ev1	LeftVentri...	19	Systole	135	139
34	LongAxis...	ev1	LeftVentri...	8	Systole	62	71
34	LongAxis...	ev1	LeftVentri...	10	Systole	79	85
34	LongAxis...	ev1	LeftVentri...	12	Systole	94	98
34	LongAxis...	ev1	LeftVentri...	14	Systole	108	116
34	LongAxis...	ev1	LeftVentri...	16	Systole	124	133
34	LongAxis...	ev1	LeftVentri...	18	Systole	139	148

(a)



(b)

Figure 5: (a) Query Results (b) Interface for displaying the retrieved video segment

3.3.1 *Types of Queries Supported by the System*

Here some basic query types are presented along with examples that the language supports. These queries can be combined to construct complex queries, which makes our language quite flexible, easy to use and powerful in expressing queries.

Object Query: This type of queries can be used to retrieve video segments containing specified objects. The query "Find the video segments where the object 'LeftVentricle' is present" can be expressed in VQL as follows.

select k1 from vdb where o1 in k1 and o1.ObjectName = LeftVentricle

Segment Query: In a segments query, conditions about segments are provided as predicates. Suppose the query is "Find the video segments which are in 'LongAxisView'". The VQL query is as follows.

select k1 from vdb where k1.ViewType = LongAxisView

State Query: This type of query can be used to retrieve state/ sub-state information about objects in video. Query like "Find the video segments where 'LeftVentricle' is in 'Systole' state" is expressed as follows.

select k1 from vdb where o1 in k1 and o1.ObjectName = LeftVentricle and o1.StateName = Systole

Event Query: Event query is specified using EventName as predicate. We can express the query "Find the video segments which start after frame number 52 and where 'LeftVentricle' is 'Contracting'" as follows.

select k1 from vdb where o1 in k1 and o1.ObjectName = LeftVentricle and o1.StateEvent = Contracting and k1.StartFrame > 52

Spatial/ Positional Query: This type of query is used to retrieve positional information about objects in video. A query such as "Find the video segments where 'LeftVentricle' is on the left of the 'RightVentricle'" is expressed as follows.

select k1 from vdb where o1, o2 in k1 and o1.ObjectName = LeftVentricle and o2.ObjectName = RightVentricle and Left(o1, o2)

Aggregate Query: In this type of query the above-mentioned aggregate functions can be used to retrieve statistical data about objects and states in Echocardiogram video. Suppose user wants to "Count the number of video segments which starts before frame number 100 and where the object 'LeftVentricle' is in 'Diastole' state". This can be expressed in VQL as follows.

select k1, Count(k1) from vdb where o1 in k1 and o1.ObjectName = LeftVentricle and OccursBefore(o1.Diastole, k1, 100)

User can specify a compound query using any combination of the above types of conditions in this video query language.

3.3.2 *Query Processing*

The main job of query processing module is to convert the queries composed in the proposed video query language into the query language provided by commercial database systems. The query processing is carried out in three phases, namely, query specification, query translation and query execution.

Query Specification:

Suppose the query is "Find all the video segments in 'longAxisView' which start after frame number 50 and end before frame number 150 and where 'LeftVentricle' is contracting". The corresponding VQL query is

select k1 from vdb where o1 in k1 and o1.ObjectName = LeftVentricle and o1.StateEvent = Contracting and k1.StartFrame > 50 and k1.EndFrame < 150 and k1.ViewType = LongAxisView

Query Translation: Once a query is submitted to the system, a lexer partitions the query into tokens and passes them to the parser. The latter then parses the VQL query and creates its parse tree. In the next phase, the parse tree is traversed for converting it into SQL query to be presented to the backend video database. The lexical analyzer and parser were implemented in JACK, a Java parser generator that works under Windows operating system. The above query is translated into SQL for a given database schema as follows.

*select * from EchoState where ObjectName = 'LeftVentricle' and StateName in (select StateName from ObjectState where ObjectName = 'LeftVentricle' and StateEvent = 'Contracting') and ViewType = 'longAxisView' and StartFrame > 50 and Endframe < 150*

Query Execution: The query interface is connected to the backend database using Open Database Connection (ODBC) layer. Retrieved datasets are presented on the interface that can be displayed when selected. Figure 5(a) shows a typical set of video segments retrieved as a result of the above query. These segments can be played one after another as shown in Figure 5(b).

4. IMPLEMENTATION DETAILS

We captured the test video data from the Quadra Diagnostic Center located in Kolkata, India. Videos were captured using GE VIVID4 ultrasound system with TruScan architecture. The **'5S'** probe was used with an operating frequency range of 2-5 MHz and multiple focal control technology. The audio/ video output from the machine was captured by a Dazzle Digital Video Converter 150 (DVC 150) card from Pinnacle Systems with frame rate of 30 frames per second.

The system was evaluated with the help of a cardiologist and a physician. The video processing routines were tested on twenty test videos. The echo videos were played at a reduced speed on a workstation to the experts for ease of identification. Experts evaluated the frames as a whole based on the content and annotated the frames with corresponding view, state and sub-state. The annotation was done independently by the two experts and cross-checked. This was used as the ground truth while computing the accuracy of our system. The view boundary detection and classification technique gives 97.19% accuracy. Misclassification error of the state detection routine is less than 13%.

5. CONCLUSIONS

We have proposed a method that comprises of several novelties as summarized here. Firstly, an efficient algorithm is used for automated detection and classification of view boundaries in an echo video. Secondly, novel state and sub-state based segmentation technique is used to index echo videos. It is the first approach in extracting sub-states from echo video to the best of our knowledge. This approach uses M-modes (color) for its high temporal resolution. This method can be used indirectly to detect the abnormalities of heart from the absence of proper color bars in the color M-mode. However, sub-state detection is done only from color flow doppler 2-D echocardiogram video. The work may be extended for detecting sub-states from general echo video. Thirdly, a new SQL like video query language has been proposed for retrieving video segments depending on users' query. This language is flexible enough in forming queries by any non-expert user.

All the analysis and modeling of echo video in current work (also in existing literature) involved extraction of visual features only. However, the audio associated with echocardiogram video also provides important information about patient status. So, audio may be used to extract information about dynamics of heart. Finding relationships between audio and video features is a promising avenue for research. We plan to include it in our future implementation.

6. ACKNOWLEDGEMENTS

The authors would like to thank Dr. P. P. Bhattacharya, Dr. D. Dasgupta and Dr. D. Goswami for helping us in capturing and annotating the echocardiogram videos. This work is supported by the project grant 1(23)/2006- ME and TMD, Dt. 07/03/2007 sponsored by the Ministry of Communication and Information Technology, Govt. of India.

7. REFERENCES

[1] S. Ebadollahi, S. F. Chang, H. Wu, and S. Takoma. Echocardiogram video summarization. In *SPIE MI 4325*, pages 492–501, 2001.

[2] S. Ebadollahi, S. F. Chang, and H. Wu. Echocardiogram videos: summarization, temporal segmentation and browsing. In *IEEE Int. Conf. on Image Process.*, 1:613-616, 2002.

[3] S. Ebadollahi, S. F. Chang, and H. Wu. Automatic view recognition in echocardiogram videos using parts-based representation. In *IEEE Comp. Society Conf. on Comp. Vision and Pattern Recog.*, 2:2-9, 2004.

[4] S. Kevin, J. H. Park, B. Georgescu, C. Simopoulos, J. Otsuki, and D. Comaniciu. Image-based multiclass boosting and echocardiographic view classification. In *IEEE Comp. Soci. Conf. on Comp. Vision and Pattern Recog.*, 2:1559-1565, 2006.

[5] B. Acharya, J. Mukherjee, and A. K. Majumdar. Modeling dynamic objects in databases: a logic based approach. *LNCS 2224, Springer Verlag*, pages 449-512, 2001.

[6] S. Ebadollahi, S. F. Chang, and H. Wu. Modeling the Activity Pattern of the Constellation of Cardiac Chambers in Echocardiogram Videos. In *Comp. Vision Appr.s to Med. Imag. Anal.*, (LNCS 4241, Springer Verlag, pages 202-213, 2006.

[7] E. Oomoto and K. Tanaka. OVID: Design and implementation of a video-object database system. *IEEE Trans. on Knowledge and Data Engg.*, 5(4):629-643, 1993.

[8] W. W. Chu, A. F. Cárdenas, and R. K. Taira. KMeD: A knowledge-based multimedia medical distributed database system. *Information Science*, 80(2):75-96, 1995.

[9] E. Hwang and V. Subrahmanian. Querying video libraries. *Journal of Visual Communication and Image Representation*, 7(1):44-60, 1996.

[10] M. Koprulu, N. K. Cicekli, and A. Yazici. Spatio-temporal querying in video databases. *Information Science*, 160(1-4):131-152, 2004.

[11] S. F. Chang, W. Chen, H. J. Meng, H. Sundaram, and D. Zhong. VideoQ: An automated content based video search system using visual cues. In *ACM Multimedia*, pages 313-324, 1997.

[12] T. Kuo and A. Chen. A Content-based query language for video databases. In *IEEE Int. Conf. on Multimedia Computing and Systems*, pages 209-214, 1996.

[13] M. Petkovic and W. Jonker. Content-based video retrieval: A database perspective. *Kluwer Academic Publishers*, 2003.

[14] M. S. Hacid, C. Decleir, and V. Kouloumdjian. A database approach for modeling and querying video data. *IEEE Trans. on Knowledge and Data Engg.*, 12(5):729-750, 2000.

[15] M. O. Donderler, O. Ulusoy, and U. Gudukbay. Rule-based spatiotemporal query processing for video databases. *Int. Journal on Very Large Data Bases*, 13(1):86-103, 2004.

[16] O. Kucuktunc, U. Gudukbay, and O. Ulusoy. A natural language-based interface for querying a video database. *IEEE Multimedia*, 14(1):83-89,2007.

[17] B. Acharya, A. K. Majumdar, and J. Mukherjee. Video model for dynamic objects. *Inf. Science*, 176(17):2567-2602, 2006.

[18] S. Shermann, M. Chan, L. Qing, Z. Wu, and Z. Zhuang. Accommodating hybrid retrieval in a comprehensive video database management system. *IEEE Trans. on Multimedia*, 4(2):146-159, 2002.

[19] C. G. M. Snoek and M. Worring. Multimedia event based video indexing using time intervals. *IEEE Trans. on Multimedia*, 7(4):638-647, 2005.

[20] A. Roy, S. Sural, J. Mukherjee, and A. K. Majumdar. State based modeling and object extraction from echocardiogram video. *IEEE Trans. on IT in Biomedicine*, 12(3):366-376, 2008.