Shiksha: A Novel Architecture for Tele-teaching Using Handwriting as a Perceptually Significant Temporal Media

Amit Singhal[1], Santanu Chaudhury[1], Sumantra Dutta Roy[1]

[1]Electrical Engineering Department, IIT Delhi, New Delhi, India – 110016
{amit.singhal, santanuc, sumantra}@ee.iitd.ac.in

Abstract. In this paper we present Shiksha − an integrated architecture which incorporates handwritten illustrations captured and rendered in a temporal fashion synchronized with audio and video data. The architecture of Shiksha permits non-linear growth in the form of multiple hierarchically organized play streams. We have developed an asynchronous multimedia conferencing application in which the users are provided with an authoring and rendering environment to record and view lectures. It also allows the users to ask and reply to doubts in the previously stored lectures making it a fully interactive but asynchronous system.

Keywords: Handwriting, Tele-teaching, Video Conferencing, Web Server.

1 Introduction

Use of integrated audio, video and text is a part of various experiential meeting, lecture capture and collaborative systems like [1], [2], [3], [4], [5], [6]. Beyond audio and video, some systems [3], [1], [7] have also captured the corresponding whiteboard data or slides. Among the handwriting enabled systems, [8] have designed and developed a virtual notepad which takes handwriting inputs from the user and stores it in the form of strokes without applying any recognition. Apoerri [9] has developed tools to annotate audio-video segments using handwritten data which is however stored as images. Schilit et al [10] have used a tablet and pen based system to enable a user to underline, highlight, or scribble comments on online documents. But none of the works has yet experimented with using handwriting as a temporal medium, and capturing and presenting the same in synchronization with the corresponding audio and video. Recognizing handwriting as a data type in itself, rather than limiting it by applying handwriting recognition, opens up a new and natural way of expression.

The major contribution in this paper is integrated document architecture, named as Shiksha, which integrates handwritten notes as a temporal medium synchronized in time with the corresponding audio and video data. It is an improvement over the architecture proposed in [11], specifically for the purpose of tele-teaching. Other unique features of Shiksha are as follows:

- Support for audio and video, along with handwritten illustrations.
- Integration of multiple play streams into a single logical document entity. This feature allows non-linear growth of the document, i.e. new play streams can be added and existing play streams can be modified.
- Integrated mechanisms to capture, transfer and render the media content in a mutually synchronized way, thus avoiding need to invoke separate applications or the use of attachments.

In the next section, we present the perceptual aspects in design. After that, we present the integrated document architecture for Shiksha. In the later sections, we describe an asynchronous video conferencing application in which we demonstrate the effective and immersive communication made possible by using Shiksha.

2 Perceptual Aspects in Design

Presentation without the presence of a person is not very effective as people tend to lose their interest. Hence, visual component becomes an integral part of the system. But then we also need to keep in mind the high bandwidth requirements for video-based systems. So, the main challenge is to hold the attention of the user, albeit the given bandwidth constraint. There has to be a judicial mixture of handwriting, audio and video. The resultant environment should give the user an experience similar to sitting in an actual classroom. For this purpose, a portion of the available bandwidth would be reserved for the continuous streaming of the audio and handwriting. The remaining bandwidth would be used for video. For small bandwidth, the video cannot be presented as a continuous stream. To solve this problem, there would be a small window (as shown in Fig. 1) which would show the video for small intervals along with the continuous stream of the audio and handwriting. These small intervals would be appropriately chosen on the basis of the lecture content. In between these intervals the video would disappear, thus saving on the bandwidth. The video would reappear at the next relevant instant in the lecture. This would use the given bandwidth in an optimal manner, and also provide an immersive experience to the user.

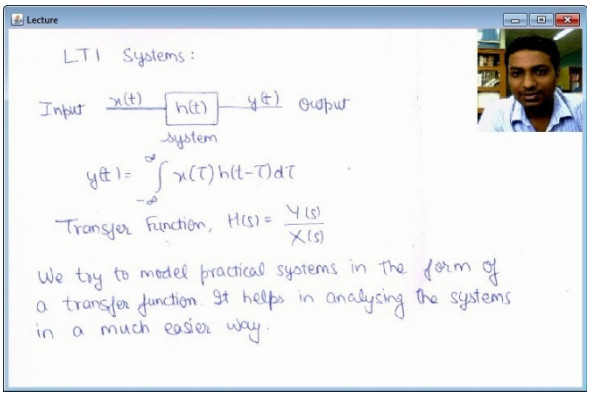

Fig. 1. Small video window on the handwriting panel

3 Document Architecture

Shiksha is a logical document entity which is an integration of multiple hierarchically organized play streams. A play stream can comprise multiple media streams which are rendered in synchronization using playback. In this section, we describe the encoding of the media streams particularly the handwritten data, synchronization of different media streams, viewing and rendering of Shiksha.

We have chosen 'QuickTime' as the video data-encoding format. For audio coding, we have used the 'wav' format. The handwriting stream is stored in the form a text file as a sequence of (x,y) coordinates of pen strokes. The current system design is flexible enough to allow change in the audio, video encoding formats. Change in the audio-video encoding formats does not affect the synchronization relations between the audio, video and the corresponding handwritten data.

Handwriting as a temporal medium is known as ink. It is more natural to draw, write, sketch using a pen rather than using a keyboard and mouse. The novel concept that has been employed in the design of this system is the preservation of temporal characteristics of handwritten data. This further enhances the communication experience, as the temporal characteristics of handwriting are captured and preserved while it is rendered back at the recipient end, giving the user the feel that the other person is writing in front of him. To facilitate use of ink data across devices and different system architectures, it is required to have a standardized representation. With the objective of having a universal standard for representing handwriting or ink, the Multi-modal Interaction working group at W3C has initiated work on one such standard named InkML, i.e. Ink Markup Language [12] which supports a complete and accurate representation of digital ink and allows temporal representation through the use of time-stamps. A typical ink document format is shown below.

```
<ink >
<captureDevice>
<channelDef name="T">
<representation type="integer" units="ms"
relativeTo="trace"/>
</channelDef>
</captureDevice>
<traceFormat id="default">
<regularChannels>
</ channel name="X" type="integer />
</ channel name="Y" type="integer />
</ channel name="T" type="integer />
</regularChannels>
</traceFormat>
<trace id="t001" timeOffset="5"> 10 20 100 11 20
120 13 20 215 19 21 290 </trace>
</ink >
```

It shows 3 channels, X, Y (which denote the *x*, *y* coordinates of each pixel of the stroke), and T which denotes the time-stamp when it was drawn. Each stroke of handwritten data is denoted as a trace and identified by a unique trace identifier.

The timeOffset contains the time-stamp when rendering of this trace is to be started. The relativeTo tag defines that the T values are relative to beginning time-stamps of individual traces. The T channel format is specifically defined in the beginning of the document using the channelDef tag. The tag regularChannels denotes that the channels embedded inside it will always have data for each pixel, i.e. all the three data values will always be present for each pixel. In our system design we have kept the different media streams in their respective containers. The advantage of this approach as against the option of having a single container for all the different media streams is that it makes the packaging independent of the audio video formats and avoids hassles of multiplexing the intermittent hand written data with the more continuous audio visual content.

3.1 Synchronization of Media Streams

To enable the temporal capture and synchronization of handwritten data with audio and video, we propose a time-stamp based synchronization mechanism for intra-media and inter-media synchronization of hand-written data. The synchronization between the consecutive data units of a temporal medium is termed as intra-media synchronization whereas that across such temporal media is termed as inter-media synchronization [13]. In our system, Java Media Framework has been used. The three media: Audio, Video, and Handwriting are transmitted over the network as two different streams (Audio-Video combined + Handwriting). We have focused on addressing the issues related to intra-media and inter-media synchronization of handwritten data since this task has not been addressed earlier. Every time the user starts authoring a document, the authoring environment starts a message clock which counts milliseconds from 0 onwards. When the user writes anything, the pen device generates events. The authoring environment responds to these events by noting down the pixel coordinates and the time-stamps derived from the message clock and stores this information in a text file.

When playing back the document, the renderer reads the stored text file containing the pixel coordinates and the time-stamps when they are to be rendered. The synchronization is done by buffering the received handwriting stream and not directly displaying on the handwriting panel and waiting for the client to receive the next 40ms time-stamped data of merged audio-video stream. Once the wait is over, the buffered handwriting data is displayed on the panel. Synchronizing the handwritten data to the message clock time rather than the time-stamps on audio-video helps make the system audio-video codec independent as well as independent of any effect of transcoding. Further the solution is applicable to different message types viz. audio-video, audio only, video only, etc.

The temporal arrangement of different play streams is achieved through synchronization based on virtual axis which is a subtype of axes based synchronization [13]. It supports the use of several virtual axes to create a virtual coordinate space. If each document path is assumed to be a virtual axis then its immediate child paths are arranged with respect to this axis, i.e. with respect to its parent stream time. This approach has a distinct advantage rather than using a global time axis to manage the document path hierarchy. Whenever a new path is added to the hierarchy, its start time is calculated with respect to the main path.

3.2 Viewing and Editing in Shiksha

Shiksha can have multiple play streams which can be organized in the form of a tree. The play stream which has got no parent is called as the root play stream. A user is presented with a hierarchical view of the different play streams in a Shiksha. The user can select the stream which he would like to watch. He can also use a tracker bar to locate a time point at which he would want to begin watching the selected play stream. The audio and video contents of the message first get downloaded on the local machine of the user, and it gets played from there. In addition, it supports multiple users to access the system at one time. The user can also select the child streams he is interested in watching rather than having to view them all. This provides instant access to the message segment the user is interested in viewing, rather than having to wait until the whole message is downloaded. During playback of a play stream the user is presented with the option of loading and viewing any concurrent child stream(s). The user may continue watching the parent stream or may select a child stream in which case the content of the child stream is played by the rendering engine.

Since Shiksha allows non-linear growth of the document, a new play stream can be linked to any point on an existing play stream. The added stream is considered as a child of the parent stream and it carries information about the ID of the parent stream and the time offset (from the start of the parent stream) for the point at which it is linked to the parent stream.

4 Deploying Shiksha for Asynchronous Video Conferencing System

We have developed an authoring and rendering engine for Shiksha. As an application of such integrated documents, we have developed a client-server based multimedia conferencing system in congruency with the paradigm of asynchronous communication. It basically acts as an ecosystem of offline lecture dissemination. A schematic diagram for the application is shown in Fig. 2.

4.1 Implementation Details

The implementation has been done in java. The various components involved are briefly described below.

GlassFish Server. The application is hosted on a GlassFish server. GlassFish is an open source application server project led by Sun Microsystems for the Java EE platform. It uses a derivative of Apache Tomcat as the servlet container for serving Web content, with an added component called Grizzly for scalability and speed.

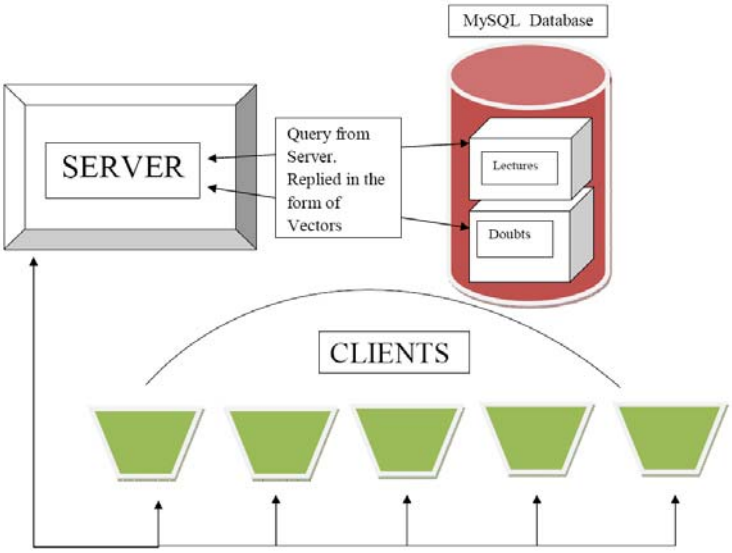

Fig. 2. Schematic Diagram

MySQL Database. The next step is creation of a Database. This is done using MySQL. MySQL is a relational database management system (RDBMS) that runs as a server providing multi-user access to a number of databases. The database has two tables: Lectures and Doubts as shown in Fig. 2. Lecture table stores all the details of the lecture and doubts stores the details of the doubts. These lectures and their corresponding doubts and comments are available for viewing to the clients and further additions can be made to them.

Java Servlets. There are five Java servlets which handle the calls from different clients and are responsible for separate functionalities. The call to these servlets is made in different points on the setup and playback timeline of the client application. Apart from handling the client requests, server provides the option of recording new lectures and adding clarifications in between the lecture in much the same way as the client adds doubt/comment to the lecture. This is achieved through two dedicated Java applications installed on the server.

The application can also be accessed through browser by entering the URL of the web server. The link of opening the required Java application is provided on the web page which when clicked by the user starts the properly guided lecture watching session on his/her terminal. This functionality is achieved by converting the executable JAR application file into a Java Web Start Application and deploying it onto the web server for direct access. The file available on the web page is essentially a JNLP (Java Network Launch Protocol) file. The web server is currently a local server at the multimedia lab in IIT Delhi which can be moved to a larger server space outside the IIT, if situation demands (increase in traffic/hits).

4.2 Functioning

View Lecture. When a client requests to view a lecture, then the server application sends a query to the database, which in turn refers to the lecture table and the available choice of the lectures is presented to the user. Since it is a repository of lectures and there will be more than one lecture, the user can chose which lecture he wants to view. A second choice is presented to the user, as to which doubt threads is he interested in viewing. Now the server accesses all the files – lectures, doubts, and comments related to that particular lecture thread and sends them to the client. All the files are stored in the client's cache, from where they are ready to be played back. Depending upon his choice, the lecture playback starts with the desired doubt threads popping in at their relevant timeline locations.

Raising a Doubt. During the lecture playback, the user might have questions or doubts in his mind. There is a system in place so that a user can conveniently ask his doubt. The user can ask a new doubt at any stage by clicking at the button 'Ask a new doubt', which would pause the playback of the lecture. His camera, microphone and the handwriting panel are activated and the query is sent to the server after the recording. The doubt also gets stored in the form of two different streams (Audio-Video combined + Handwriting). Once a doubt has been asked, there is an option to respond to it in the form of a new button 'Respond to Doubt'. On a future access of the particular lecture thread by any user, this recent doubt also appears in the playback options.

5 Conclusions

The use of Shiksha with its support for audio-visual and handwritten illustrations facilitates an experiential recording of events and near live interaction among the users. There are many schools and colleges in distant parts of India which do not have adequate teaching resources. By using this application, we can reach to those students and help them with their studies. By using a projector at the client end, the lecture experience seems almost as if a teacher is standing in front and delivering it. The option to ask and reply to doubts makes the system interactive also. In addition, the option to shut video also allows us to save bandwidth in real life situations. Since the system is asynchronous, the lecture can be viewed at any convenient time instead of following a fixed schedule.

References

1. D. Bargeron and J. Grudin. As users grow more savvy: Experiences with an asynchronous distance learning tool. In 37th Annual Hawaii International Conference on System Sciences, pages 96 – 105, Jan 2004.
2. K. Curran. A web-based collaboration teaching environment. IEEE Multimedia, 9(3):72 – 76, July 2002.

3. J. A. Brotherton, J. R. Bhalodia, and G. D. Abowd. Automated Capture, Integration and Visualization of Multiple Media Streams. In IEEE International Conference on Multimedia Computing and Systems, pages 54 – 63, 1998.
4. R. Jain, P. Kim, and Z. Li. Experiential Meeting System. In ACM SIGMM workshop on Experiential Telepresence, California USA, pages 1 – 12, Nov 2003.
5. C. K. Hess, D. Lin, and K. Nahrstedt. VistaMail: An Integrated Multimedia Mailing System. In IEEE Multimedia, pages 13 – 20, Oct-Dec 1998.
6. J. H. Watt, J. B. Walther, and K. L. Nowak. Asynchronous Videoconferencing: A Hybrid Communication Prototype. In International Conference on System Sciences, pages 13 – 21, Jan 2002.
7. S. LeeTiernan and J. Grudin. Fostering Engagement in Asynchronous learning through collaborative multimedia annotation. Technical Report Microsoft Research Technical Report MSR-TR-2000-91, 22, Microsoft Research Technical Report MSR-TR-2000-91, 22, September 2000.
8. I. Poupyrev, N. Tomokazu, and S. Weghorst. Virtual Notepad: Handwriting in Immersive VR. In IEEE Virtual Reality Annual International Symposium, pages 126 – 132, March 1998.
9. A. Apoerri. How to make audio/video an easy to use and share as text. In ASIST'02, Philadelphia, USA, 2002.
10. B. N. Schilit, G. Golovchimlq, and M. N. Price. Beyond paper: Supporting active reading with free form digital ink annotations. In ACM SIGCHI conference on Human factors computing systems Los Angeles, California, United States, pages 249 – 256, 1998.
11. G. Harit, V. Mankar, and S. Chaudhary. Patra: A novel document architecture for integrating handwriting with audio-visual information. In 9th International Conference on Document Analysis and Recognition (ICDAR), pages 699-703, Sep 2007.
12. I. I. M. Language. W3C Working Draft. http://www.w3.org/TR/InkML/.
13. G. Blakowski and R. Steinmetz. A media synchronization survey: Reference model, specification, and case studies. In IEEE Journal on Selected Areas Communication, volume 14, pages 5 – 35, Jan 1996.